Amortized Safe Active Learning for Real-Time Data Acquisition: Pretrained Neural Policies From Simulated Nonparametric Functions

Cen-You Li[1,2,◇] **Marc Toussaint**[1,3] **Barbara Rakitsch**[4,⋆] **Christoph Zimmer**[4,5,⋆]
[1]Technical University of Berlin, Germany [2]University of Helsinki, Finland
[3]Robotics Institute Germany [4]Bosch Center for Artificial Intelligence, Germany
[5]Baden-Württemberg Cooperative State University, Mannheim, Germany
[◇]Correspondence: cen-you.li@helsinki.fi [⋆]Equal contribution.

Abstract

Safe active learning (AL) is a sequential scheme for learning unknown systems while respecting safety constraints during data acquisition. Existing methods often rely on Gaussian processes (GPs) to model the task and safety constraints, requiring repeated GP updates and constrained acquisition optimization–incurring significant computations which are challenging for real-time decision-making. We propose amortized AL for regression and amortized safe AL, replacing expensive online computations with a pretrained neural policy. Inspired by recent advances in amortized Bayesian experimental design, we leverage GPs as pretraining simulators. We train our policy prior to the AL deployment on simulated nonparametric functions, using Fourier feature-based GP sampling and a differentiable acquisition objective that is safety-aware in the safe AL setting. At deployment, our policy selects informative and (if desired) safe queries via a single forward pass, eliminating GP inference and acquisition optimization. This leads to magnitudes of speed improvements while preserving learning quality. Our framework is modular and, without the safety component, yields fast unconstrained AL for time-sensitive tasks.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

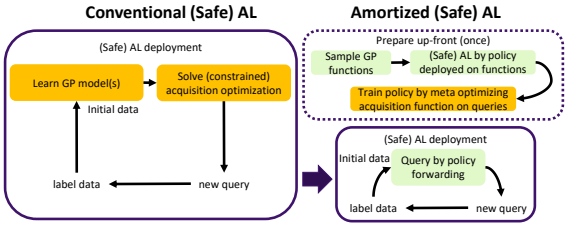

Figure 1: Conventional (safe) AL relies on computationally expensive (orange) GP fitting and (constrained) acquisition. Our amortized approach meta trains a safe learner up-front on synthetic data, allowing fast, real-time (green) deployment.

1 INTRODUCTION

Active learning (AL) is a sequential design of experiments, aiming to learn a task with reduced data labeling effort (Settles, 2010; Kumar and Gupta, 2020; Tharwat and Schenck, 2023). Each label is queried by optimizing an acquisition function, a function leveraging the current knowledge (typically model-based) to estimate the expected information gained from accessing new labels. AL is often discussed together with Bayesian optimization (BO), which aims to search global optima with limited evaluations (Srinivas et al., 2012; Brochu et al., 2010). The major difference is the acquisition function, where BO focuses only on candidate optima, while AL explores the complete space.

In many engineering (Zimmer et al., 2018; Berkenkamp et al., 2016) or chemical design problems (Griffiths and Hernández-Lobato, 2020), data evaluations can trigger safety concerns. The safety constraints, often real values, cannot be directly mapped to the input space, motivating the development of safe AL (Schreiter et al., 2015; Zimmer et al., 2018; Li et al., 2022) and safe BO (Sui et al., 2015, 2018; Berkenkamp et al., 2020). These approaches introduce additional model(s) to

quantify safety conditions and constrain the acquisition optimization (Figure 1 left). Gaussian processes (GPs, Rasmussen and Williams 2006) are widely used in this context due to their well-calibrated uncertainty estimates, suited for modeling safety confidence.

Safe learning methods are prominent, capable of learning functions adaptively and safely, needing no parametric structure. However, computation is heavy: (i) GPs scale cubically with the size of dataset and are updated repeatedly; (ii) each query solves an acquisition optimization. The cost poses a particular challenge for systems requiring real-time responses (Nguyen–Tuong and Peters, 2010; Andersson et al., 2017; Lederer et al., 2021). To alleviate this, efficient GP approximations have been explored (Titsias, 2009; Hensman et al., 2015a; Bitzer et al., 2023), including approaches designed for incremental data (Moss et al., 2023; Pescador-Barrios et al., 2024).

In this paper, we focus on AL for regressions, particularly under safety constraints (Schreiter et al., 2015; Zimmer et al., 2018), which are also relevant to safe BO, where some recent approaches separate safe space exploration from BO phase (Sui et al., 2018; Bottero et al., 2022). We aim to amortize the online data querying process. Inspired by recent amortized Bayesian experimental design (BED) literature (Foster et al., 2021; Ivanova et al., 2021), we propose to learn an AL policy offline using synthetic functions. The policy is a neural network (NN) which suggests a new query via a simple forward pass (Figure 1 right) hereby replacing both the GP modeling and the constrained optimization at deployment. Our paper first amortizes unconstrained AL on regressions, and then introduces safety awareness to the framework. Note that we use the term *model* to refer to the task-specific model being actively learned, while the NN policy guides the data collections, feeding the data to the model.

In a nutshell, our approach (i) takes GPs as distributions of general nonparametric functions, (ii) utilizes a scalable Fourier feature technique (Rahimi and Recht, 2007; Wilson et al., 2020) to generate functions in large scale, (iii) solves and meta learns AL decisions on those functions, and then (iv) zero-shot generalizes to real-world problems.

Contributions Our contributions are:

- We propose NN policies for safe AL and, as an intermediate contribution, for unconstrained AL, that suggest new queries based on recorded data, hereby completely replacing the costly GP modeling and acquisition optimization at deployment;
- This leads to a tremendous speed gain, allowing for real-time data acquisition with modeling performance comparable to traditional AL methods, while maintaining safety when required;
- Our policy is trained up-front exclusively on synthetic data of a broad class of functions sampled via GP Fourier features, enabling generalization across systems;
- To train our safe AL policy, we introduce a closed-form, differentiable, safety-aware objective, which may itself be a novel acquisition criterion.

Related Works Safe learning often employs GPs. Gelbart et al. (2014) proposed constrained BO, discounting acquisition scores by a GP constraint confidence. Schreiter et al. (2015); Sui et al. (2015) directly constrained the acquisition optimization, leading to safe AL and safe BO with probabilistic safety guarantees. Sui et al. (2018) introduced a stagewise safe BO that separates safe space exploration from BO phase, enabling AL methods to be applied specifically for safety (Bottero et al., 2022). Safe AL and safe BO were extended to systems with multiple safety constraints (Berkenkamp et al., 2020), time-series modeling (Zimmer et al., 2018), multi-task learning (Li et al., 2022) and transfer learning (Li et al., 2025). High-dimensional safe BO has also been explored via approximations and multi-stage procedures (Kirschner et al., 2019; Bottero et al., 2022). Safe learning methods employ significant computations, impeding their deployment in real-time problems.

Meta-learning has been explored to streamline sequential learning. Rothfuss et al. (2021) meta-learned GP priors from existing tasks, simplifying GP modeling in safe BO. Liu et al. (2020b); Bitzer et al. (2023) pretrained on synthetic data to infer the GP parameters. Neural processes (NPs, Garnelo et al. 2018; Foong et al. 2020), in particular transformer-based NPs (TNPs, Nguyen and Grover 2022), replace GPs by learning the posterior estimates. Müller et al. (2022) developed the prior-fitted networks (PFNs), TNPs trained on synthetic data, resulting in recent tabular foundation models (e.g. TabPFN, Hollmann et al. 2023, 2025). NPs and PFNs demonstrate amortized modeling costs and have been extended for unconstrained BO applications (Müller et al., 2023). These methods require existing meta tasks or do not consider (constrained) acquisition optimization, while our method simply queries end-to-end on constrained AL.

To streamline the entire data selection cycle, Chen et al. (2017) proposed an NN optimizer for BO tasks, which bypassed modeling and optimization by directly inferring the next query from evaluated data. Chang et al. (2025) proposed predicting the global optimum

for BO by taking a prior over the optimum as input to an NN. Foster et al. (2021); Ivanova et al. (2021) proposed the deep adaptive design (DAD), inferring queries for unconstrained AL of Bayesian *parametric* functions (which requires a known parametric structure, in contrast to our method). Huang et al. (2024) extended such amortized BED to account for model-based decision-making. DAD trains its NN policies on synthetic data, but the necessity of parametric structures limits training flexibility.

Our approach uses GPs as generic simulators to pre-train end-to-end policies for general, nonparametric (safe) AL, enabling real-time deployment on novel tasks. The closest related work is ALINE (Huang et al., 2025), a recently proposed framework that jointly amortizes posterior estimation and experimental design. However, ALINE cannot accommodate constrained learning and relies on a discretized search pool. In contrast, our method addresses standard and safe AL, and is developed so that the NN proposes queries directly on a continuous space.

2 PROBLEM STATEMENT

We are interested in a regression task of an unknown function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subseteq \mathbb{R}^D$ is a D-dimensional input space. We have another unknown safety function $q : \mathcal{X} \to \mathbb{R}$. As one normally focuses only on a domain of interest, we assume $\mathcal{X}$ is bounded, w.l.o.g., we may say $\mathcal{X} = [0, 1]^D$.

Our observations are noisy: a labeled data point comprises an input $\boldsymbol{x} \in \mathcal{X}$, its corresponding output observation $y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon$, and its safety measurement $z(\boldsymbol{x}) = q(\boldsymbol{x}) + \epsilon_q$, where ϵ, ϵ_q are unknown noise values. For clarity later, let $\mathcal{Y} \subseteq \mathbb{R}, \mathcal{Z} \subseteq \mathbb{R}$ denote the output space and the safety measurement space, respectively. $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ is a dataset, and $space(\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}) \coloneqq \{\mathcal{D} | \mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}\}$. We write $y_{\text{subscript}}$, $z_{\text{subscript}}$ as evaluated data at $\boldsymbol{x}_{\text{subscript}}$.

We follow a safe AL setting: a small labeled dataset $\mathcal{D}_0 \coloneqq \{\boldsymbol{x}_{\text{init},i}, y_{\text{init},i}, z_{\text{init},i}\}_{i=1}^{N_{\text{init}}}$ is given, and we have budget to label T more data points $\boldsymbol{x}_1, ..., \boldsymbol{x}_T$. The expensive evaluations will give us $y_1, z_1, ..., y_T, z_T$. It is safety critical if for any $t \in \{1, ..., T\}$, $z_t \geq 0$ is violated. Safe AL aims to select $\boldsymbol{x}_1, ..., \boldsymbol{x}_T$, such that $z_1 \geq 0, ..., z_T \geq 0$ with high probability, and that $y_1, ..., y_T$ are informative, i.e. $\{\boldsymbol{x}_{\text{init},i}, y_{\text{init},i}\}_{i=1}^{N_{\text{init}}} \cup \{\boldsymbol{x}_t, y_t\}_{t=1}^{T}$ helps us construct a good model of f. In this paper, we write $\boldsymbol{x}_{1:T} \coloneqq \{\boldsymbol{x}_t\}_{t=1}^{T}$, $y_{1:T} \coloneqq \{y_t\}_{t=1}^{T}$, $z_{1:T} \coloneqq \{z_t\}_{t=1}^{T}$, $\boldsymbol{X}_{\text{init}} \coloneqq \{\boldsymbol{x}_{\text{init},i}\}_{i=1}^{N_{\text{init}}}$, $Y_{\text{init}} \coloneqq \{y_{\text{init},i}\}_{i=1}^{N_{\text{init}}}$, $Z_{\text{init}} \coloneqq \{z_{\text{init},i}\}_{i=1}^{N_{\text{init}}}$.

Conventional safe AL (Figure 1 left and Schreiter et al. (2015); Zimmer et al. (2018)) obtains each query at step $t+1$ by solving

$$\begin{aligned} &\text{argmax}_{\boldsymbol{x} \in \mathcal{X}} \; a(\boldsymbol{x} | Y_{\text{init}}, y_{1:t}) \\ &\qquad s.t. \; p(z(\boldsymbol{x}) \geq 0 | Z_{\text{init}}, z_{1:t}) \geq 1 - \gamma, \end{aligned} \tag{1}$$

where $p(z(\boldsymbol{x}) \geq 0 | Z_{\text{init}}, z_{1:t})$ is the predictive safety distribution, a is an acquisition function, $\gamma \in [0, 1]$ is a probability tolerance of being unsafe, and the space of safe $\boldsymbol{x}$ is the safe set. The estimated safe set adapts to each new safety measurement (Sui et al., 2015, 2018). The acquisition function and safety distribution are computed from GPs. It is expensive to iteratively update GPs and solve the constrained optimization.

Goal We aim to have an AL policy up-front so that, at deployment, each query is produced by a single forward pass–replacing the per-query modeling and constrained acquisition optimization (Figure 1 right; Appendix F.2). Formally, the policy takes as inputs a budget variable and a flexible size of observed data, and returns the next query proposal, i.e. $\phi : \mathbb{N} \times space(\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}) \to \mathcal{X}$. We will describe in Section 3.1 why a budget variable is included into the policy input. We assume no additional real data are available for the policy training. Our contributions are twofold: (i) We amortize unconstrained AL ($\phi : \mathbb{N} \times space(\mathcal{X} \times \mathcal{Y}) \to \mathcal{X}$); (ii) we extend to safe AL by incorporating safety-awareness. Concretely, our paper investigates (i) how to simulate AL tasks at large scale for training, and (ii) how to design an effective training objective. Next, we state the necessary modeling assumptions on the unknown functions f and q.

Assumptions We assume the tasks are normalized to zero mean and unit variance. Furthermore, as existing safe learning methods (e.g. Zimmer et al. 2018; Berkenkamp et al. 2020), we assume the unknown functions f and q have GP priors (Rasmussen and Williams, 2006). A GP is a distribution over functions, characterized by a mean (e.g., $\mathbb{E}[f]$) and a kernel specifying covariance between function values at two inputs, $\boldsymbol{x}$ and $\boldsymbol{x}'$ (e.g., $\text{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')]$). A kernel, typically parameterized, encodes the function amplitude and smoothness. W.l.o.g., the prior mean is usually assumed zero, which holds true when the observation values are normalized. For the safety function, one could as well assume a zero mean prior, but a well-selected prior mean can sometimes be beneficial. For example, if the domain of interest $\mathcal{X}$ is chosen such that the center area is safe, then we may assign a prior mean which remains reasonably positive at the center but decreases to negative values at the boundary. Given GP priors, any finite number of output values (or of the safety values) are jointly Gaussian. The assumption is formulated below, while closed-form GP distributions are detailed in Appendix A.

Assumption 2.1. The unknown functions follow GP priors: $f \sim \mathcal{GP}(\mathbf{0}, k_\theta), q \sim \mathcal{GP}_{\theta_q}(\mu_q, k_q)$, with kernels $k_\theta, k_q : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a mean $\mu_q : \mathcal{X} \to \mathbb{R}$. k_θ is parameterized by θ; μ_q and k_q are jointly parameterized by θ_q. The output and safety observations at $\boldsymbol{x}$ are $y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon$, $z(\boldsymbol{x}) = q(\boldsymbol{x}) + \epsilon_q$, where $\epsilon \underset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2), \epsilon_q \underset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_q^2)$. We assume $k_\theta(\boldsymbol{x}, \boldsymbol{x}'), k_q(\boldsymbol{x}, \boldsymbol{x}') \leq 1$, as the data are normalized.

3 PRETRAIN POLICY TO REPLACE GP AND CONSTRAINED ACQUISITION

Our goal here is to train a policy ϕ to deploy AL on novel tasks. In other words, we construct our preparation block illustrated in Figure 1. Here we take inspiration from Chen et al. (2017) and DAD (Foster et al., 2021; Ivanova et al., 2021). The idea is to exploit the GP priors (Assumption 2.1) before AL deployments. We use $p(f), p(q)$ and the Gaussian likelihoods $p(y|\boldsymbol{x}, f) = \mathcal{N}\left(y|f(\boldsymbol{x}), \sigma^2\right)$, $p(z|\boldsymbol{x}, q) = \mathcal{N}\left(z|q(\boldsymbol{x}), \sigma_q^2\right)$ to construct a simulator. Notably, we go beyond DAD by considering the safety function q and nonparametric priors on f and q. This allows us to sample functions, simulate policy-based (safe) AL and, crucially, *meta optimize on broad classes of functions*, with an objective encoding an acquisition criterion.

The major challenges are: (i) our scheme requires differentiable objectives, where applying conventional formulations is difficult; (ii) sampling f, q is not trivial, especially with a constraint on q. We summarize our approach in Algorithm 1, and give details next.

3.1 Training Objective

Assume, we are given a batch of GP functions f, each coupled with a safety GP function q. Initial evaluations $\mathcal{D}_0 = \{\boldsymbol{X}_{\text{init}}, Y_{\text{init}}, Z_{\text{init}}\}$ are given, noises are denoted by $Y_{\text{init}} = f(\boldsymbol{X}_{\text{init}}) + \mathcal{E}_{\text{init}}, Z_{\text{init}} = q(\boldsymbol{X}_{\text{init}}) + \mathcal{E}_{q,\text{init}}$. We run our policy on each (f, q) pair for $T_{\text{sim}} \leq T$ iterations to obtain $\boldsymbol{x}_{\phi,1:T_{\text{sim}}}$, evaluations $y_{\phi,1:T_{\text{sim}}}$ and $z_{\phi,1:T_{\text{sim}}}$, $y_{\phi,t} = f(\boldsymbol{x}_{\phi,t}) + \epsilon_t$ and $z_{\phi,t} = q(\boldsymbol{x}_{\phi,t}) + \epsilon_{q,t}$ for $t = 1, ..., T_{\text{sim}}$ (L.5-11 of Algorithm 1). In this section, we introduce our training objectives (L.13 of Algorithm 1). The details of f, q and $\mathcal{D}_0$ sampling will be described in Section 3.2 (L.1,2,4 of Algorithm 1).

Unconstrained AL–Simulated Acquisition Assume for now that safety conditions are ignored. We are then dealing with an unconstrained AL: we collect $y_{\phi,1:T_{\text{sim}}}$ informative for f. The GP prior $f \sim \mathcal{GP}(\mathbf{0}, k_\theta)$ further turns this into an active GP learning problem. This means common acquisition functions (Seo et al., 2000; Guestrin et al., 2005; Krause et al., 2008) are valid to guide our queries to high information. Note, however, that we now optimize w.r.t. the policy where gradient is propagated from *all queries jointly*. A good choice here is the entropy (Seo et al., 2000; Krause and Guestrin, 2007), which (i) has closed forms, (ii) is differentiable, and (iii) is one of the gold standard acquisition criteria for GPs.

As opposed to an AL deployment, the sampled $y_{\phi,1:T_{\text{sim}}}$ are available when we optimize the queries $\boldsymbol{x}_{\phi,1:T_{\text{sim}}}$. For each AL instance, we take the following acquisition

$$\begin{aligned} h(y_{\phi,1:T_{\text{sim}}}, Y_{\text{init}}) &= -\log p(y_{\phi,1:T_{\text{sim}}}, Y_{\text{init}}) \\ \propto h(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}}) &= -\log p(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}}), \end{aligned} \tag{2}$$

where $\mathbb{E}_{f,T_{\text{sim}},\mathcal{E}_{\text{init}},\epsilon_{1:T_{\text{sim}}}}[h(y_{\phi,1:T_{\text{sim}}}, Y_{\text{init}})]$, an average of various instances, represents the policy's entropy defined by Krause and Guestrin (2007) (originally for conventional AL). $p(\cdot)$ is a GP likelihood (Appendix A). The proportionality symbol here indicates equivalency which holds by applying Bayes rule and removing the part that has no gradient w.r.t. the policy. The effect of T_{sim} will be described shortly after, let us say $T_{\text{sim}} = T$ is fixed here. Maximizing this acquisition criterion means the policy selects points that are the most distinctive to each other. We will later see that this choice has the advantage of explainability in conjunction with safety constraints. In our Appendix C (Figure S.C.2), we illustrate the acquisition value with $T = 1$.

The ideas until here are similar to Chen et al. (2017); Foster et al. (2021); Ivanova et al. (2021), i.e. turning the acquisition criteria we would have optimized sequentially into trainable objectives in an a priori simulated learning.

We further propose the regularized entropy criterion:

$$\begin{aligned} I(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}}) = &-\log p(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}}) \\ &+\log p(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}}, Y_{\text{grid}}), \end{aligned} \tag{3}$$

where Y_{grid} are noisy evaluations at randomly sampled $\boldsymbol{X}_{\text{grid}} \subseteq \mathcal{X}$ (with $|\boldsymbol{X}_{\text{grid}}| \gg T$). This $I(\cdot)$ is adapted from the mutual information AL criterion (Guestrin et al., 2005; Krause et al., 2008). It encourages the policy to look inside the input space and avoids overemphasizing the border of $\mathcal{X}$, a problem of entropy. We put details into Appendix C.

Unconstrained AL Objective, beyond Fixed Length & Fixed Priors One remark of the previous objectives is that they are nonmyopic[1] (Krause and Guestrin, 2007; Foster et al., 2021), assuming a pre-defined budget T. The resulting queries can be

[1]Nonmyopic exploration allocates multiple points jointly, e.g. query $1/3, 2/3$ in $[0, 1]$ instead of $1/2, 1/4$.

Algorithm 1 Safe AL Policy Training

Require: Assumption 2.1, T, N_{init}
draw a batch of $(\theta, \sigma^2, \theta_q, \sigma_q^2)$
draw a batch of (f, q) pairs (Algorithm 2)
for each f, q **do**
  given $\mathcal{D}_0$ per Algorithm 2
  draw $T_{\text{sim}} \sim \text{Uniform}[1, T]$
  for $t = 1, ..., T_{\text{sim}}$ **do**
    $\boldsymbol{x}_{\phi,t} = \phi(T_{\text{sim}} - t + 1, \mathcal{D}_{t-1})$
    draw $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, $y_{\phi,t} = f(\boldsymbol{x}_{\phi,t}) + \epsilon_t$
    draw $\epsilon_{q,t} \sim \mathcal{N}(0, \sigma_q^2)$, $z_{\phi,t} = q(\boldsymbol{x}_{\phi,t}) + \epsilon_{q,t}$
    $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{\boldsymbol{x}_{\phi,t}, y_{\phi,t}, z_{\phi,t}\}$
  end for
end for
compute loss per Eq. (7), update ϕ

suboptimal if we deploy for fewer steps. In practice, it is often unrealistic to know the precise number of queries T in advance, especially if we wish to deploy the trained policy on multiple problems. To this end, we assign random AL budget to each f, i.e. $T_{\text{sim}} \sim \text{Uniform}[1, T]$. Then, the exploration scores are normalized by the sequence length (AL sims indicate random $f, T_{\text{sim}}, \mathcal{E}_{\text{init}}, \epsilon_{1:T_{\text{sim}}}$):

$$\mathcal{H}(\phi) = \mathbb{E}_{\theta,\sigma^2}\mathbb{E}_{\text{AL sims}}\left[\frac{h(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}})}{N_{\text{init}} + T_{\text{sim}}}\right], \quad (4)$$

$$\mathcal{I}(\phi) = \mathbb{E}_{\theta,\sigma^2}\mathbb{E}_{\text{AL sims}}\left[\frac{I(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}})}{N_{\text{init}} + T_{\text{sim}}}\right]. \quad (5)$$

Crucially, the NN needs a budget variable as input to encode the number of queries. Without this budget variable, $T_{\text{sim}} = T$ must be fixed. Due to the space limit, the NN structure is described in Appendix B. Eqs. (4), (5) generalize over diverse functions by sampling the GP hyperparameters θ, σ^2.

Safe AL Objective We have obtained an entropy objective for unconstrained AL. Now we take $z_{\phi,1:T_{\text{sim}}}$ into consideration and we wish to follow the same intuition to get a safe AL objective. Recall that a conventional safe AL (Schreiter et al., 2015; Zimmer et al., 2018) solves Eq. (1) for each query. This constrained acquisition criterion has proven its effectiveness, and we next translate it into a differentiable objective compatible with our simulated environment.

If we query with our unconstrained entropy objective step-wise, the corresponding acquisition function is $a(\boldsymbol{x}) = -\log p(\hat{y}(\boldsymbol{x})|y_{\phi,1:t}, Y_{\text{init}})$, where $\hat{y}(\boldsymbol{x})$ is the simulated noisy function.[2] Plugging this $a(\cdot)$ into Eq. (1), we see that at each step $t+1$, the conventional safe AL objective optimizes $-\log p(\hat{y}(\boldsymbol{x})|y_{\phi,1:t}, Y_{\text{init}})$, constrained to $p(z(\boldsymbol{x}) \geq 0|z_{\phi,1:t}, Z_{\text{init}}) \geq 1-\gamma$, $z(\boldsymbol{x})$ is a prediction. In a Lagrange perspective, a constrained optimization may be transformed to a problem where we optimize the main term regularized by a factor of the constraint term (Nocedal and Wright, 2006; Gramacy et al., 2015). The factor, i.e. Lagrange multiplier, typically needs to be optimized as well, but we take the density interpretation to fix it. We propose to augment the problem and twist the Lagrangian form:

1. $p(z(\boldsymbol{x}) \geq 0|z_{\phi,1:t}, Z_{\text{init}}) \geq 1-\gamma$ is equivalent to $\log p(z(\boldsymbol{x}) < 0|z_{\phi,1:t}, Z_{\text{init}}) \leq \log(\gamma)$.

2. We consider a Lagrangian form

$$\begin{aligned}&\text{argmax}_{\boldsymbol{x}}\{-\log p(\hat{y}(\boldsymbol{x})|y_{\phi,1:t}, Y_{\text{init}})\\ &-\lambda \log p(z(\boldsymbol{x}) < 0|z_{\phi,1:t}, Z_{\text{init}}) + \lambda \log \gamma\},\end{aligned}$$

   where λ is a Lagrange multiplier. We fix $\lambda = 1$ as the likelihood terms can be interpreted as a joint likelihood of events $\hat{y}$ and unsafe z. Note in addition that $\log \gamma$ is a constant w.r.t. $\boldsymbol{x}$, which can be omitted. This form has thus become

$$\begin{aligned}&\text{argmax}_{\boldsymbol{x}}\{-\log p(\hat{y}(\boldsymbol{x})|y_{\phi,1:t}, Y_{\text{init}})\\ &\quad -\log p(z(\boldsymbol{x}) < 0|z_{\phi,1:t}, Z_{\text{init}})\},\end{aligned}$$

3. Importantly, we wish to preserve the parameter γ as it defines the desired safety level which helps to balance between exploration and safety. We thus adjust the previous form into

$$\begin{aligned}&\text{argmax}_{\boldsymbol{x}}\{-\log p(\hat{y}(\boldsymbol{x})|y_{\phi,1:t}, Y_{\text{init}}) \quad (6)\\ &\quad -\log \max(\gamma, p(z(\boldsymbol{x}) < 0|z_{\phi,1:t}, Z_{\text{init}}))\},\end{aligned}$$

   where γ disregards the exact safety level when the likelihood of being unsafe is small enough.

4. Similar to an unconstrained AL, we sum up the step-wise acquisition scores to get a joint safe AL objective which is differentiable w.r.t. the policy.

Our safe AL training objective is thus

$$\begin{aligned}&\mathcal{S}_{\mathcal{H}}(\phi) = \mathbb{E}\left[\frac{S(\mathcal{D}_{T_{\text{sim}}})}{N_{\text{init}} + T_{\text{sim}}}\right], \text{with} \quad (7)\\ &S(\mathcal{D}_{T_{\text{sim}}}) = -\log p(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}})\\ &-\sum_{t=0}^{T_{\text{sim}}-1} \log \max(\gamma, p(z(\boldsymbol{x}_{\phi,t+1}) < 0|z_{\phi,1:t}, Z_{\text{init}})).\end{aligned}$$

The expectation is over GP hyperparameters and AL instances, and $z_{\phi,1:0} = \emptyset$. $z(\boldsymbol{x}_{\phi,t+1})$ is a prediction

[2] $\hat{y}(\boldsymbol{x}_{\phi,t}) = y_{\phi,t}$. A standard sequential learning would use a random variable $y(\boldsymbol{x})$ to forecast the future query, leveraging $a(\boldsymbol{x}) = \mathbb{H}(y(\boldsymbol{x})|y_{\phi,1:t}, Y_{\text{init}})$.

here, because a realization $z_{\phi,t+1}$ cannot have a likelihood of $z_{\phi,t+1} < 0$.

Maximizing $\mathcal{S}_{\mathcal{H}}(\phi)$ corresponds to maximizing the exploration score of $y_{\phi,1:T_{\text{sim}}}$, while minimizing the likelihood of queries appearing unsafe. In conventional safe AL, $\gamma \in [0, 1]$ (but usually < 0.5) allows us to specify the level of safety criticality. In our setting, this parameter balances safety and AL exploration, offering a more intuitive approach than optimizing a Lagrange multiplier. Setting $\gamma \to 0$ means that the safety term $\log p(z(\boldsymbol{x}_{\phi,t+1}) < 0 | z_{\phi,1:t}, Z_{\text{init}})$ is optimized towards $-\infty$, causing safety to dominate the loss and overriding exploration (see ablation studies in Appendix G.5). We set $\gamma = 0.05$ similar to many conventional safe AL (Li et al., 2025), resulting in a safe yet explorative policy. In Appendix C.3, we provide an alternative objective maximizing exploration directly alongside the likelihood of queries being safe, i.e. $\log p(z(\boldsymbol{x}_{\phi,t+1}) \geq 0 | z_{\phi,1:t}, Z_{\text{init}})$. This loss cannot accommodate safety criticality control via γ due to the concavity of $\log(\cdot)$; Appendix G.6 (ablation studies) shows that this alternative is less safe than our primary loss. Appendix C.3 further illustrates the objectives for $T_{\text{sim}} = 1$ (Figure S.C.3). This illustration demonstrates that our main acquisition loss (Eqs. (6) and (7)) features a safe plateau, and optimizing this loss tends to select points within this region.

Our objective function $\mathcal{S}_{\mathcal{H}}(\phi)$ can vary with the order of queried data, as the order directly impacts the safety score. This inherits the conventional property that safety confidence adapts to every new observation. Note that the Eq. (6) can itself be a safety-aware yet unconstrained differentiable acquisition criterion for conventional safe AL setups (see ablation studies in Appendix G.6).

3.2 Function Sampling

In the previous section, we introduced an algorithm based on generative GP functions f, q, and an initial dataset $\mathcal{D}_0$. These functions were used to simulate safe AL deployments for policy training as summarized in Algorithm 1. Here, we detail the generative process (sampling steps: lines 1–4, 8, 9), focusing on: (i) sampling the initial dataset $\mathcal{D}_0$ with a suitable prior mean for the safety function q, and (ii) efficient GP function sampling for f and q. The first point is critical for ensuring realistic synthetic data, the second one for efficient and stable training.

Safety Function q and Initial Data $\mathcal{D}_0$ The first challenge is to ensure that the simulated q and $\mathcal{D}_0$ are both realistic and representative. In safe exploration problems (Sui et al., 2015; Zimmer et al., 2018; Bottero et al., 2022), the initial data are usually provided by a domain expert at the centric area of a safe set and the algorithms gradually explore towards the safe set border. Following this principle, our generative algorithm samples initial data $\mathcal{D}_0$ in a pre-defined safe set, and we assume w.l.o.g. that the safe set lies at the center of $\mathcal{X}$, denoted by $\mathcal{C} \subseteq \mathcal{X}$. This approach ensures that $\mathcal{D}_0$ closely resembles the initial datasets encountered in the deployment stage.

For this, we construct a GP prior $q \sim \mathcal{GP}_{\theta_q}(\mu_q, k_q)$ to increase the likelihood of a safe initial dataset $\mathcal{D}_0$. Specifically, we design a prior mean μ_q that ensures safety around the center, e.g. $\mathcal{C} = [0.4, 0.6]^D$ for $\mathcal{X} = [0, 1]^D$. As a result, $q(\mathcal{C})$ may likely have a safe region to initiate a safe AL simulation, while still allowing for a diverse set of safety functions by introducing variability within the GP. In Appendix D, we provide the exact expression for our mean function μ_q, which is designed to allow for a safe set of varying shape and size, while maintaining consistency with the deployment problem's normalization assumptions. Examples of q are illustrated in Figure S.D.4.

When we sample the initial dataset $\mathcal{D}_0$ in Algorithm 2, we sample safe data from $\mathcal{C}$, but once a maximum number of iteration is reached, the training algorithm proceeds regardless of whether all sampled points are safe.

Fourier Feature Functions - Efficient and Decoupled The final step is to devise an efficient sampling strategy for the noisy GP values $y_{\phi,1:T_{\text{sim}}}$ and $z_{\phi,1:T_{\text{sim}}}$. This is not trivial because the observations are sampled iteratively, e.g. $\boldsymbol{x}_{\phi,t} = \phi(\mathcal{D}_{t-1})$, meaning $y_{\phi,1:t-1}$, $z_{\phi,1:t-1}$ are sampled before we know $\boldsymbol{x}_{\phi,t}, ..., \boldsymbol{x}_{\phi,T}$. One can make standard GP posterior sampling conditioned on preceding samples, e.g. $y_{\phi,t} \sim p(y(\boldsymbol{x}_{\phi,t}) | \mathcal{D}_{t-1}, k_\theta, \sigma^2)$. However, when performed naively, this posterior sampling approach is bound to the GP cubic complexity for each data point

Algorithm 2 Function and Initial Sampling

Require: Assumption 2.1, center $\mathcal{C} \subseteq \mathcal{X}$, N_{init}, $\mathcal{D} = \emptyset$, $\mathcal{GP}(0, k_\theta), \mathcal{GP}_{\theta_q}(\mu_q, k_q), \sigma^2, \sigma_q^2$, max_iter= 50

1: draw $f_{\text{raw}} \sim \mathcal{GP}(0, k_\theta)$, $q_{\text{raw}} \sim \mathcal{GP}(0, k_q)$
2: $f(\cdot) = f_{\text{raw}}(\cdot) - \mathbb{E}_{\boldsymbol{x} \in \mathcal{X}}[f_{\text{raw}}(\boldsymbol{x})]$
3: $q(\cdot) = \mu_q(\cdot) + q_{\text{raw}}(\cdot) - \mathbb{E}_{\boldsymbol{x} \in \mathcal{X}}[q_{\text{raw}}(\boldsymbol{x})]$
4: **for** i=1,..., max_iter **do**
5: draw $\boldsymbol{X} \sim \text{Uniform}[\mathcal{C}], |\boldsymbol{X}| = N_{\text{init}}$
6: draw $\mathcal{E}_{\text{init}} \sim \mathcal{N}(0, \sigma^2 I), Y = f(\boldsymbol{X}) + \mathcal{E}_{\text{init}}$
7: draw $\mathcal{E}_{q,\text{init}} \sim \mathcal{N}(0, \sigma_q^2 I), Z = q(\boldsymbol{X}) + \mathcal{E}_{q,\text{init}}$
8: take those safe: $\mathcal{D} \leftarrow \mathcal{D} \cup [\boldsymbol{X}, Y, Z]_{z \geq 0}$
9: **if** i=max_iter **then** $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{X}, Y, Z)$
10: **if** $|\mathcal{D}| \geq N_{\text{init}}$ **then** take first N_{init} and break
11: **end for**
12: **return** $f, q, \mathcal{D}_0 = \mathcal{D}$ $(|\mathcal{D}| = N_{\text{init}})$

Table 1: **RMSE on standard AL tasks**. Performance on the test set. Our method shows competitive results while being orders of magnitude faster (Figure 2). Table S.G.10 show results of smaller T.

$N_{\text{init}}+T$	Sinus (1+20)	Airline (1+20)	Branin (1+30)	LGBB (1+30)
Our AAL	0.14±0.004	0.41±0.022	0.21±0.020	0.17±0.013
ALINE	0.38±0.034	0.43±0.011	0.27±0.013	0.17±0.009
DAD	0.49±0.065	0.43±0.018	0.99±0.128	0.48±0.063
PFN_AL	0.85±0.088	0.44±0.041	0.38±0.019	0.19±0.008
TabPFN_AL	1.04±0.033	0.43±0.023	0.22±0.008	0.16±0.010
AGP_AL	0.14±0.007	0.48±0.020	0.23±0.025	0.22±0.006
GP_AL	0.13±0.009	0.43±0.038	0.19±0.011	0.17±0.010
SVGP_AL	0.12±0.004	0.42±0.012	0.34±0.003	0.17±0.005
MGP_AL	0.12±0.007	0.39±0.020	0.17±0.006	0.15±0.008
Random	0.33±0.055	0.41±0.023	0.28±0.052	0.21±0.050

acquisition $y_{\phi,t}$, which is prohibitive. The runtime can be further reduced by using low-rank updates (Seeger, 2004). For specific objectives such as for our $\mathcal{H}$ or $\mathcal{S}_{\mathcal{H}}$, the GP posteriors can be then computed as intermediate results with minimal additional cost (see Eqs. (6) and (7)). However, low-rank updates can lead to additional issues, as they make the computational graph unnecessarily complex for the loss function differentiation. To overcome these challenges, we propose a decoupled approach, which is scalable by itself and enables flexible integration of any objectives, e.g. potentially cheap approximated objectives, for future work.

The core is a decoupled function sampling technique (Rahimi and Recht, 2007; Wilson et al., 2020), where each GP function is approximated by a linear combination of Fourier features. The function can be evaluated later at any $\boldsymbol{x} \in \mathcal{X}$ in linear time (line 8-9 of Algorithm 1). One requirement, however, is that the kernels k_θ, k_q need to have Fourier transforms (e.g. stationary kernels, see Bochner's theorem in Rasmussen and Williams 2006). For $q \sim \mathcal{GP}(\mu_q, k_q)$, one may sample $q_{\text{raw}} \sim \mathcal{GP}(0, k_q)$ and set $q = \mu_q + q_{\text{raw}}$. Note that a sampled function does not necessarily average to zero on $\mathcal{X}$, but we may compute a domain specific average analytically with usually negligible complexity (Appendix D.1). This is preferred because we consider normalized deployment problems, and such an optional mean shift improves the performance.

While not explored in this paper, we point out that alternative approaches might exist for efficient sampling, such as linear system solvers (Lin et al., 2024) or different random features (Tripp et al., 2023).

4 EXPERIMENTS

In this section, we empirically evaluate our method. We first present unconstrained AL results, demonstrating the effectiveness of our approach in terms of AL exploration. Next, we incorporate safety constraints and show that our approach remains effective in exploration while maintaining safety requirements. All experiments confirm that our methods are well suited for AL tasks requiring real-time querying. Our code is available at https://github.com/cenyou/ASAL.

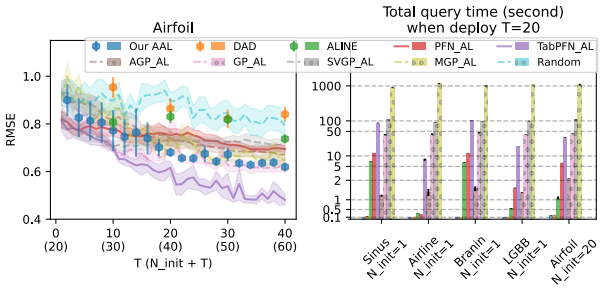

Figure 2: **Empirical results on standard AL.** Left: RMSE on airfoil dataset vs number of queries T. Our trained policy is deployed at $T = 2, 4, \ldots, 40$. DAD and ALINE train separate policies for each T (shown for $T = 10, 20, 30, 40$). Right: total query time at $T = 20$ across datasets. Our approach is significantly faster, requiring only a single NN forward pass for each data acquisition.

Experimental Setup We prepare the experiments by training our neural network policy utilizing Algorithm 1, corresponding to the up-front preparation block in Figure 1. We train one NN policy for each dimensionality D and for each experimental setting (unconstrained, safe). The policies are then applied to all benchmark problems of their respective configuration, corresponding to the deployment block in Figure 1. Our NN policy returns points on continuous space $\mathcal{X} \subseteq \mathbb{R}^D$. On benchmark functions, the query $\boldsymbol{x}_t = \phi(\cdot)$ is used directly for data acquisition. For testing on discrete dataset, we approximate the query by selecting the nearest point from the pool using the $L2$-norm. Since our high-level goal is to model a regression task, we use the final collected dataset to train a GP model and assess its performance. All experiments report modeling performance, measured with Root Mean Squared Error (RMSE), and AL deployment time. For safe AL, we additionally study if the safety threshold is satisfied. Each AL task is repeated five times with different random seeds and initial data. We provide more experimental details in Appendix F, including the training time (Table S.F.4).

4.1 Amortized Unconstrained AL (AAL)

We first study our AAL on standard unconstrained AL tasks by training on the objective $\mathcal{I}$ (Eq. (5)) and removing the safety samples from the generative process

in Algorithm 1 (see Algorithm S.1 for a cleaner version of unconstrained training). We compare AAL against (i) ALINE (Huang et al., 2025); (ii) DAD: amortized BED (Foster et al., 2021); (iii) PFN AL: AL where the model is a PFN (Müller et al., 2022) pretrained on our GP data; (iv) TabPFN AL: AL where the model is the TabPFN foundation model (Hollmann et al., 2025); (v) AGP AL: AL with amortized GP (AGP, Bitzer et al. 2023), where GP hyperparameters are not trained but inferred; (vi) GP AL: AL with a vanilla GP (trained on Type II maximum likelihood); (vii) SVGP AL: AL with sparse variational GP (Titsias, 2009; Hensman et al., 2013, 2015b), where the distribution is approximated with 15 inducing variables (ivs, pseudo data); (viii) MGP AL: AL with mixture of vanilla GPs (Riis et al., 2023); (ix) Random: random selection criterion. We provide details and deployment complexities in Appendices F.3 and F.4. We run experiments on 2 benchmark functions and 3 real-world datasets: Sin function (1D), Branin function (2D), Airline dataset (1D), Langley Glide-Back Booster dataset (LGBB, 2D), and Airfoil dataset (5D).

We report the mean and standard errors on the test dataset in Table 1 and Figure 2, training loss values in Appendix G. AL methods generally outperform random sampling, with the exception of the Airline dataset. Among the AL methods, we observe that the performance of AGP sometimes and of DAD often deteriorates (DAD is particularly bad when $D \geq 2$). AGP (Bitzer et al., 2023) infers hyperparameters to approximate Type II maximum likelihood, which might not fully align with the exploration needs of AL tasks. DAD is not designed for nonparametric modeling (see Appendix C.2). SVGP approximates vanilla GP with 15 ivs. However, training a SVGP requires more iterations than a vanilla GP while the computation reduction of each iteration is not obvious in our data sizes–resulting in a higher time cost. MGP method provides the strongest modeling results in most of the tasks, but requires multiple GP models and is around 3000x slower than our AAL. ALINE, PFN and TabPFN have the same deployment complexity up to a constant factor (Appendix F.5 and Table S.F.5). ALINE and PFN show comparable learning outcomes and TabPFN learns particularly good on the Airfoil dataset; however, they are slower than our approach in all scenarios due to attention to the pool. For 20 queries, our AAL spends < 0.3 (s), around 10x-3000x faster than AGP's < 3 (s), vanilla GP's < 50 (s), SVGP's < 120 (s), and MGP's 16 (min). When compared to other amortized approaches (ALINE, PFN, and TabPFN), AAL is at least 5x faster, except on the small Airline dataset (less than 200 points split into observation, validation, and pool sets) where it is roughly 2.5x faster. Additionally, our method offers a significant advantage in training time compared to ALINE; because ALINE relies on a discretized search pool, its training time is an order of magnitude longer (Table S.F.4).

In summary, our results clearly highlight the advantages of our method on time-sensitive tasks: AAL operates at least an order of magnitude faster, while remaining highly effective in exploration.

4.2 Amortized Safe AL (ASAL)

In this section, we study safe AL by training on the objective $\mathcal{S}_{\mathcal{H}}$ (Eq. (7)) with Algorithm 1. We compare our ASAL against conventional model-based safe AL baselines, i.e. solving Eq. (1) for each query decision: (i) Safe PFN AL which leverages PFNs for the main f and the safety function q (PFN pretrained on our GP data); (ii) Safe TabPFN AL where f and q are modeled by the TabPFN foundation model (Hollmann et al., 2025); (iii) Safe AGP AL where the models are AGPs (Bitzer et al., 2023); (iv) Safe GP AL (Schreiter et al., 2015; Zimmer et al., 2018; Li et al., 2022) with vanilla GPs (Type II maximum likelihood); (v) Safe SVGP AL with SVGPs, where the number of ivs are 20 for the Engine and Fluid System and 15 for other tasks (see tasks below); (vi) Safe MGP AL where MGPs Riis et al. (2023) are the models; (vii) Safe Random, where base acquisition scores are random values and the safety values are estimated by a vanilla GP. Except for Safe Random, all the baselines employ constrained entropy as the acquisition criteria. All methods are deployed with a safety level of $\gamma = 0.05$ (Eq. (1)).

We deploy safe AL on two constrained benchmark functions and two real-world datasets and a higher-dimensional Fluid System task: Simionescu function (2D), Townsend function (2D), LGBB dataset (2D), Engine measurements (3D), high-pressure Fluid System (7D). Our method builds on the standard safe AL criteria (Eq. (1)) with a focus on real-time deployment. Prior work typically focuses on low dimensions ($D \leq 3$ or 4, Table S.F.6) because safe acquisition optimization becomes increasingly burdensome as D grows. Our experiments on the Fluid System extend beyond the typical dimension. For this task, the training lengthscale of our GP prior is centered on a plausible operating scale for the system but kept sufficiently broad to remain generic (all other training and deployment settings are unchanged). Note that PFN and TabPFN cannot be deployed on the Fluid System. We learn the Fluid System actively from a million trajectories; this size of dataset is far beyond what PFNs and TabPFN are designed for. PFNs and TabPFN need to process all candidate queries, consuming memory exceeding the limit of our machine.

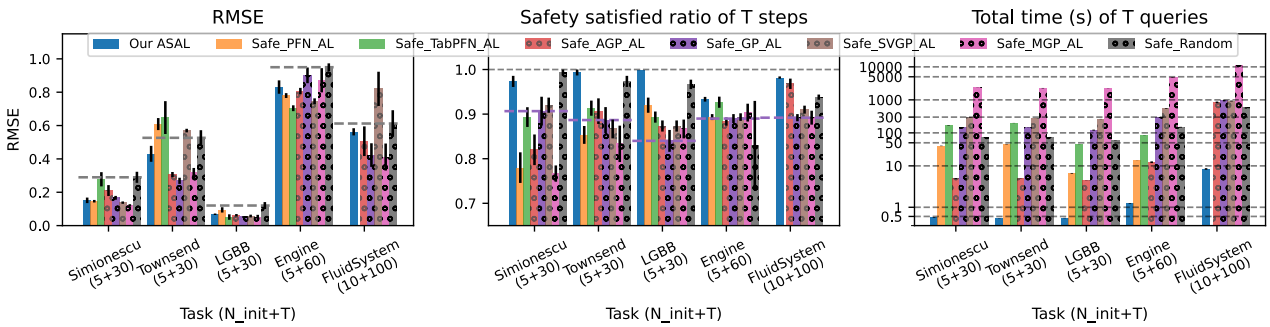

Figure 3: **Results on safe AL.** Left: Our method (blue) achieves competitive RMSE (on safe test data) and outperforms Safe Random across all tasks. Middle: The proportion of safe queries out of T queries confirm that our method queries highly safe data. Right: Our approach is significantly faster, as we avoid online GP modeling and acquisition optimization. Safe Random is slow due to the GP-based safe set estimation. PFN and TabPFN are not feasible on the Fluid System due to the large pool.

Our main results are shown in Figure 3, with further analyses presented in Appendix G, including an ablation on the effect of γ during policy training, and an ablation of conventional safe AL equipped with our novel acquisition criterion (Eq. (6)). ASAL achieves comparable model performance to Safe (A)GP AL on all datasets, except for a slight disadvantage on Townsend and the 7D Fluid System, and outperforms Safe Random across the board. It ensures safety awareness–the proportion of safe queries exceeds $1-\gamma$ on all tasks. The PFN and TabPFN methods have mixed results in terms of performance and safety awareness – while the methods are comparable to the GP baselines on LGBB and Engine, on Simionescu and Townsend either the performance is bad or the queries are unsafe (Safe GP AL as a reference). Importantly, ASAL is at least an order of magnitude faster than all PFN-based and GP-based methods (10x-600x), including Safe Random which relies on GPs for safe set estimation. On the 7D Fluid System, where the complexity of acquisition optimization is significant, the typically fast AGP baseline takes around 8.4 s/query, whereas our ASAL remains below 0.1 s/query, at least 80x faster. These results highlight the practical advantages of ASAL in real-time data acquisitions.

5 CONCLUSION AND DISCUSSION

We propose AAL and ASAL approaches that leverage GPs to pretrain AL and safe AL policies. In particular, we introduce a novel safety-aware yet unconstrained, differentiable loss and an efficient sampling scheme via GP Fourier features to simulate data acquisitions. The trained policy can zero-shot generalize to novel problems, enabling deployment that (i) queries informative and (if desired) safe data, and (ii) eliminates the need for online GP modeling and (constrained) acquisition optimization, resulting in a significant speed-up that facilitates real-time data acquisition.

Many existing safe learning methods (Sui et al., 2015, 2018; Zimmer et al., 2018; Lederer et al., 2021) offer theoretical safety guarantees, which our method does not provide. However, these guarantees typically rely on the assumption of well-chosen GP hyperparameters a priori–a strong assumption that is rarely met in real-world AL applications.

Our training relies on standard GP-based AL approaches. Therefore, it inherits limitations from established GP methods and also benefits from progress in GP modeling research. A future direction here is to extend to higher dimension (10D or higher, which remains challenging in general), for example by incorporating dimension reduction methods or scalable GPs. Our approach leverages similar idea of amortized BED (Foster et al., 2021; Ivanova et al., 2021; Huang et al., 2024), where sequential acquisitions are compressed into an offline training objective. This formulation complicates the integration of multi-stage optimizations, as often required e.g. to incorporate approximate GPs (Titsias, 2009; Hensman et al., 2015a) (even with known GP hyperparameter samples, approximate posteriors and acquisition criteria would be optimized with distinct objectives) and higher-dimensional AL (Zhang et al., 2016) and safe learning (Kirschner et al., 2019; Bottero et al., 2022).

Acknowledgments

This work was supported by Bosch Center for Artificial Intelligence, which provided financial support and computers, as well as by the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG). The Bosch Group is carbon neutral. Administration, manufacturing and research activities no longer leave a carbon

footprint.

During the course of this project, Cen-You Li transitioned to a position supported by the Research Council of Finland (Flagship programme: Finnish Center for Artificial Intelligence, FCAI), and Christoph Zimmer transitioned to Baden-Württemberg Cooperative State University.

References

Andersson, O., Wzorek, M., and Doherty, P. (2017). Deep learning quadcopter control via risk-aware active learning. *AAAI Conference on Artificial Intelligence*.

Berkenkamp, F., Krause, A., and Schoellig, A. P. (2020). Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics. *Machine Learning*.

Berkenkamp, F., Schoellig, A. P., and Krause, A. (2016). Safe controller optimization for quadrotors with Gaussian processes. *International Conference on Robotics and Automation*.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. (2018). Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*.

Bitzer, M., Meister, M., and Zimmer, C. (2023). Amortized inference for Gaussian process hyperparameters of structured kernels. *Conference on Uncertainty in Artificial Intelligence*.

Bottero, A., Luis, C., Vinogradska, J., Berkenkamp, F., and Peters, J. R. (2022). Information-theoretic safe exploration with Gaussian processes. *Conference on Neural Information Processing Systems*.

Brochu, E., Cora, V. M., and de Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv*.

Chang, P. E., Loka, N., Huang, D., Remes, U., Kaski, S., and Acerbi, L. (2025). Amortized probabilistic conditioning for optimization, simulation and inference. *International Conference on Artificial Intelligence and Statistics*.

Chen, Y., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P., Botvinick, M., and de Freitas, N. (2017). Learning to learn without gradient descent by gradient descent. *International Conference on Machine Learning*.

Foong, A. Y., Bruinsma, W. P., Gordon, J., Dubois, Y., Requeima, J., and Turner, R. E. (2020). Meta-learning stationary stochastic process prediction with convolutional neural processes. *Conference on Neural Information Processing Systems*.

Foster, A., Ivanova, D. R., Malik, I., and Rainforth, T. (2021). Deep Adaptive Design: Amortizing sequential Bayesian experimental design. *International Conference on Machine Learning*.

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. A. (2018). Conditional neural processes. *International Conference on Machine Learning*.

Gelbart, M. A., Snoek, J., and Adams, R. P. (2014). Bayesian optimization with unknown constraints. *Conference on Uncertainty in Artificial Intelligence*.

Gramacy, R. B., Gray, G. A., Digabel, S. L., Lee, H. K. H., Ranjan, P., Wells, G., and Wild, S. M. (2015). Modeling an augmented lagrangian for blackbox constrained optimization. *arXiv*.

Griffiths, R.-R. and Hernández-Lobato, J. M. (2020). Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*.

Guestrin, C., Krause, A., and Singh, A. P. (2005). Near-optimal sensor placements in Gaussian processes. *International Conference on Machine Learning*.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. *Conference on Uncertainty in Artificial Intelligence*.

Hensman, J., Matthews, A., and Ghahramani, Z. (2015a). Scalable variational Gaussian process classification. *International Conference on Artificial Intelligence and Statistics*.

Hensman, J., Matthews, A. G., Filippone, M., and Ghahramani, Z. (2015b). Mcmc for variationally sparse Gaussian processes. *Conference on Neural Information Processing Systems*.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. (2023). TabPFN: A transformer that solves small tabular classification problems in a second. *International Conference on Learning Representations*.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. (2025). Accurate predictions on small data with a tabular foundation model. *Nature*.

Huang, D., Guo, Y., Acerbi, L., and Kaski, S. (2024). Amortized Bayesian experimental design for decision-making. *Conference on Neural Information Processing Systems*.

Huang, D., Wen, X., Bharti, A., Kaski, S., and Acerbi, L. (2025). ALINE: Joint amortization for Bayesian

inference and active data acquisition. *Conference on Neural Information Processing Systems*.

Ivanova, D. R., Foster, A., Kleinegesse, S., Gutmann, M. U., and Rainforth, T. (2021). Implicit Deep Adaptive Design: Policy-based experimental design without likelihoods. *Conference on Neural Information Processing Systems*.

Kirschner, J., Mutny, M., Hiller, N., Ischebeck, R., and Krause, A. (2019). Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. *International Conference on Machine Learning*.

Krause, A. and Guestrin, C. (2007). Nonmyopic active learning of Gaussian processes: An exploration-exploitation approach. *International Conference on Machine Learning*.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*.

Kumar, P. and Gupta, A. (2020). Active learning query strategies for classification, regression, and clustering: A survey. *Journal of Computer Science and Technology*.

Lederer, A., Conejo, A. J. O., Maier, K. A., Xiao, W., Umlauft, J., and Hirche, S. (2021). Gaussian process-based real-time learning for safety critical applications. *International Conference on Machine Learning*.

Li, C.-Y., Dünnbier, O., Toussaint, M., Rakitsch, B., and Zimmer, C. (2025). Global safe sequential learning via efficient knowledge transfer. *Transactions on Machine Learning Research*.

Li, C.-Y., Rakitsch, B., and Zimmer, C. (2022). Safe active learning for multi-output Gaussian processes. *International Conference on Artificial Intelligence and Statistics*.

Lin, J. A., Padhy, S., Mlodozeniec, B. K., Antoran, J., and Hernández-Lobato, J. M. (2024). Improving linear system solvers for hyperparameter optimisation in iterative gaussian processes. *Conference on Neural Information Processing Systems*.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. (2020a). On the variance of the adaptive learning rate and beyond. *International Conference on Learning Representations*.

Liu, S., Sun, X., Ramadge, P. J., and Adams, R. P. (2020b). Task-agnostic amortized inference of Gaussian process hyperparameters. *Conference on Neural Information Processing Systems*.

Moss, H. B., Ober, S. W., and Picheny, V. (2023). Inducing point allocation for sparse Gaussian processes in high-throughput Bayesian optimisation. *International Conference on Artificial Intelligence and Statistics*.

Müller, S., Feurer, M., Hollmann, N., and Hutter, F. (2023). PFNs4BO: In-context learning for Bayesian optimization. *International Conference on Machine Learning*.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. (2022). Transformers can do Bayesian inference. *International Conference on Learning Representations*.

Nguyen, T. and Grover, A. (2022). Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. *International Conference on Machine Learning*.

Nguyen–Tuong, D. and Peters, J. (2010). Incremental sparsification for real-time online model learning. *International Conference on Artificial Intelligence and Statistics*.

Nocedal, J. and Wright, S. J. (2006). Numerical optimization. *Springer*.

Pamadi, B. N., Covell, P. F., Tartabini, P. V., and Murphy, K. J. (2004). Aerodynamic characteristics and glide-back performance of langley glide-back booster. *Applied Aerodynamics Conference and Exhibit*.

Pescador-Barrios, G., Filippi, S., and van der Wilk, M. (2024). "how big is big enough?" adjusting model size in continual Gaussian processes. *arXiv*.

Pu, M., Songhao, W., Wang, H., and Ng, S. H. (2025). Weighted euclidean distance matrices over mixed continuous and categorical inputs for Gaussian process models. *International Conference on Artificial Intelligence and Statistics*.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. *Conference on Neural Information Processing Systems*.

Rainforth, T., Foster, A., Ivanova, D. R., and Smith, F. B. (2024). Modern Bayesian experimental design. *Statistical Science*.

Rasmussen, C. and Williams, C. (2006). Gaussian processes for machine learning. *MIT Press*.

Riis, C., Antunes, F., Hüttel, F. B., Azevedo, C. L., and Pereira, F. C. (2022). Bayesian active learning with fully Bayesian Gaussian processes. *Advances in Neural Information Processing Systems*.

Riis, C., Rodrigues, F., and Pereira, F. C. (2023). Bayesian active learning with fully Bayesian Gaussian processes. *TechRxiv*.

Rogers, S. E., Aftosmis, M. J., Pandya, S. A., Chaderjian, N. M., T., E. T., and Ahmad, J. U. (2003). Au-

tomated cfd parameter studies on distributed parallel computers. *AIAA Computational Fluid Dynamics Conference*.

Rothfuss, J., Fortuin, V., Josifoski, M., and Krause, A. (2021). Pacoh: Bayes-optimal meta-learning with pac-guarantees. *International Conference on Machine Learning*.

Schreiter, J., Nguyen-Tuong, D., Eberts, M., Bischoff, B., Markert, H., and Toussaint, M. (2015). Safe exploration for active learning with Gaussian processes. *Machine Learning and Knowledge Discovery in Databases*.

Seeger, M. (2004). Low rank updates for the cholesky decomposition.

Seo, S., Wallat, M., Graepel, T., and Obermayer, K. (2000). Gaussian process regression: active data selection and test point rejection. *International Joint Conference on Neural Networks*.

Settles, B. (2010). Active learning literature survey. *University of Wisconsin-Madison*.

Simionescu, P. (2014). Computer-aided graphing and simulation tools for autocad users. *Computer-Aided Graphing and Simulation Tools for AutoCAD Users*.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. W. (2012). Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*.

Sui, Y., Gotovos, A., Burdick, J., and Krause, A. (2015). Safe exploration for optimization with Gaussian processes. *International Conference on Machine Learning*.

Sui, Y., Zhuang, V., Burdick, J. W., and Yue, Y. (2018). Stagewise safe Bayesian optimization with Gaussian processes. *International Conference on Machine Learning*, 80.

Tharwat, A. and Schenck, W. (2023). A survey on active learning: State-of-the-art, practical challenges and research directions. *Mathematics*.

Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. *International Conference on Artificial Intelligence and Statistics*.

Townsend, A. (2017). Constrained optimization in chebfun. *chebfun.org*.

Tripp, A., Bacallado, S., Singh, S., and Hernández-Lobato, J. M. (2023). Tanimoto random features for scalable molecular machine learning.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. *Conference on Neural Information Processing Systems*.

Viering, T. J., Adriaensen, S., Rakotoarison, H., Müller, S., Hvarfner, C., Hutter, F., and Bakshy, E. (2025). α-PFN: In-context learning entropy search. *Workshop on Frontiers in Probabilistic Inference: Learning meets Sampling at the International Conference on Learning Representations*.

Wilson, J. T., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2020). Efficiently sampling functions from Gaussian process posteriors. *International Conference on Machine Learning*.

Zhang, Y., Hoang, T. N., Low, K. H., and Kankanhalli, M. (2016). Near-optimal active learning of multi-output Gaussian processes. *AAAI Conference on Artificial Intelligence*.

Zimmer, C., Meister, M., and Nguyen-Tuong, D. (2018). Safe active learning for time-series modeling with Gaussian processes. *Conference on Neural Information Processing Systems. Clarification/Errata on arxiv: https://arxiv.org/abs/2402.06276*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes, Section 2, Section 3]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes, Appendix E, Appendix F.5]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes, https://github.com/cenyou/ASAL]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable, our work is mainly empirical while assumptions are stated in Section 2]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes, https://github.com/cenyou/ASAL; all datasets are publically available (described in Appendix F.6)]

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes, Appendix F.1]

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes, Table S.F.4 Appendix F]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. [Yes, we cite existing works we build upon]

(b) The license information of the assets, if applicable. [Yes]

(c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Materials

Table of Contents

A Gaussian Process: Distribution and Entropy

GP Distribution We first write down the GP predictive distribution. Details can be seen in Rasmussen and Williams (2006). We write down a general form with a non-zero prior mean $q \sim \mathcal{GP}(\mu_q, k_q)$. The GP hyperparameters are omitted for brevity. One can remove μ_q to get a zero-mean GP distribution, e.g. for y.

Given a set of N_{observe} data points $\mathcal{D} = \{\boldsymbol{X}, Z\} \subseteq \mathcal{X} \times \mathcal{Z}$, we wish to make inference at points $\boldsymbol{X}_{\text{test}} = \{\boldsymbol{x}_{\text{test},1}, ..., \boldsymbol{x}_{\text{test},N_{\text{test}}}\}$. We write $Z_{\text{test}} = (z(\boldsymbol{x}_{\text{test},1}), ..., z(\boldsymbol{x}_{\text{test},N_{\text{test}}}))$ for brevity. The joint distribution of Z and predictive Z_{test} is Gaussian:

$$p(Z, Z_{\text{test}}) = \mathcal{N}\left(\mu_q(\boldsymbol{X} \cup \boldsymbol{X}_{\text{test}}), k_q(\boldsymbol{X} \cup \boldsymbol{X}_{\text{test}}, \boldsymbol{X} \cup \boldsymbol{X}_{\text{test}}) + \sigma_q^2 I_{N_{\text{observe}}+N_{\text{test}}}\right) \tag{S.1}$$

where $k_q(\boldsymbol{X} \cup \boldsymbol{X}_{\text{test}}, \boldsymbol{X} \cup \boldsymbol{X}_{\text{test}})$ is a gram matrix with $[k_q(\boldsymbol{X} \cup \boldsymbol{X}_{\text{test}}, \boldsymbol{X} \cup \boldsymbol{X}_{\text{test}})]_{i,j} = k_q\left([\boldsymbol{X} \cup \boldsymbol{X}_{\text{test}}]_i, [\boldsymbol{X} \cup \boldsymbol{X}_{\text{test}}]_j\right)$.

This leads to the following predictive distribution (or GP posterior distribution)

$$\begin{aligned} p(Z_{\text{test}}|Z) &= \mathcal{N}\left(Z_{\text{test}}|\mu_{\mathcal{D}}(\boldsymbol{X}_{\text{test}}), cov_{\mathcal{D}}(\boldsymbol{X}_{\text{test}})\right), \\ \mu_{\mathcal{D}}(\boldsymbol{X}_{\text{test}}) &= k_q\left(\boldsymbol{X}_{\text{test}}, \boldsymbol{X}\right)\left[k_q\left(\boldsymbol{X}, \boldsymbol{X}\right) + \sigma_q^2 I_{N_{\text{observe}}}\right]^{-1}\left[Z - \mu_q\left(\boldsymbol{X}\right)\right] + \mu_q\left(\boldsymbol{X}_{\text{test}}\right), \\ cov_{\mathcal{D}}(\boldsymbol{X}_{\text{test}}) &= k_q\left(\boldsymbol{X}_{\text{test}}, \boldsymbol{X}_{\text{test}}\right) + \sigma_q^2 I_{N_{\text{test}}} \\ &\quad - k_q\left(\boldsymbol{X}_{\text{test}}, \boldsymbol{X}\right)\left[k_q\left(\boldsymbol{X}, \boldsymbol{X}\right) + \sigma_q^2 I_{N_{\text{observe}}}\right]^{-1} k_q\left(\boldsymbol{X}, \boldsymbol{X}_{\text{test}}\right). \end{aligned} \tag{S.2}$$

Elements of the predictive mean vector $\mu_{\mathcal{D}}(\boldsymbol{X}_{\text{test}})$ are the noise-free predictive values.

Inverting a $N_{\text{observe}} \times N_{\text{observe}}$ matrix $\left[k_q\left(\boldsymbol{X}, \boldsymbol{X}\right) + \sigma_q^2 I_{N_{\text{observe}}}\right]$ has complexity $\mathcal{O}(N_{\text{observe}}^3)$ in time.

When a GP is used to model the probability beyond a threshold, e.g. a predictive safety probability, at a test point $\boldsymbol{x}$ this is a cumulative distribution function

$$p(z(\boldsymbol{x}) \geq 0|Z) = 1 - p(z(\boldsymbol{x}) < 0|Z) = 1/2\left[1 + erf\left(\frac{\mu_{\mathcal{D}}(\boldsymbol{x})}{\sqrt{2 cov_{\mathcal{D}}(\boldsymbol{x})}}\right)\right]. \tag{S.3}$$

The log probability density function is

$$\begin{aligned} \log p(Z_{\text{test}}|Z) = &- N_{\text{test}}/2 \log(2\pi) - 1/2 \log\det(cov_{\mathcal{D}}(\boldsymbol{X}_{\text{test}})) \\ &- 1/2(Z_{\text{test}} - \mu_D(\boldsymbol{X}_{\text{test}}))^T \left[cov_{\mathcal{D}}(\boldsymbol{X}_{\text{test}})\right]^{-1}(Z_{\text{test}} - \mu_D(\boldsymbol{X}_{\text{test}})), \end{aligned} \tag{S.4}$$

inverting $cov_{\mathcal{D}}(\boldsymbol{X}_{\text{test}})$ or computing the determinant takes $\mathcal{O}(N_{\text{test}}^3)$ in time.

GP Entropy If we consider Z_{test} as a collection of N_{test} random variables, the entropy is

$$\mathbb{H}(Z_{\text{test}}|Z) = \mathbb{E}_{p(Z_{\text{test}}|Z)}[-\log p(Z_{\text{test}}|Z)] = N_{\text{test}}/2 \log(2\pi e) + 1/2 \log\det(cov_{\mathcal{D}}(\boldsymbol{X}_{\text{test}})). \tag{S.5}$$

The difference between $-\log p(\cdot)$ and $\mathbb{H}(\cdot)$ is marked blue, and this comparison will be described later when we look into different training objectives. Note further that if we plug predicted values $Z_{\text{test}} = \mu_D(\boldsymbol{X}_{\text{test}})$ into Eq. (S.4), then the blue term becomes zero and this becomes proportional to negative entropy.

GP Training When we deploy a conventional (safe) AL, or when we use the collected data after a deployment to model a function, we usually need to select the hyperparameters. One standard approach is to conduct a Type II maximum likelihood (vanilla GP):

$$\text{argmax}_{\theta_q, \sigma^2} \log p(Z|\boldsymbol{X}) = \text{argmax}_{\theta_q, \sigma^2} \log \mathcal{N}\left(Z|\mu_q(\boldsymbol{X}), k_q(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 I_{N_{\text{observe}}}\right),$$

where the free variables are hyperparameters of μ_q, k_q and the noise variance σ^2. The time complexity is $\mathcal{O}\left(N_{\text{observe}}^3\right)$ while the exact factor depends on the number of parameters, the optimizer, and the numerical stability. We keep the prior mean μ_q for consistent notation even though the only place when we train GPs is during deployment where we use zero mean GPs.

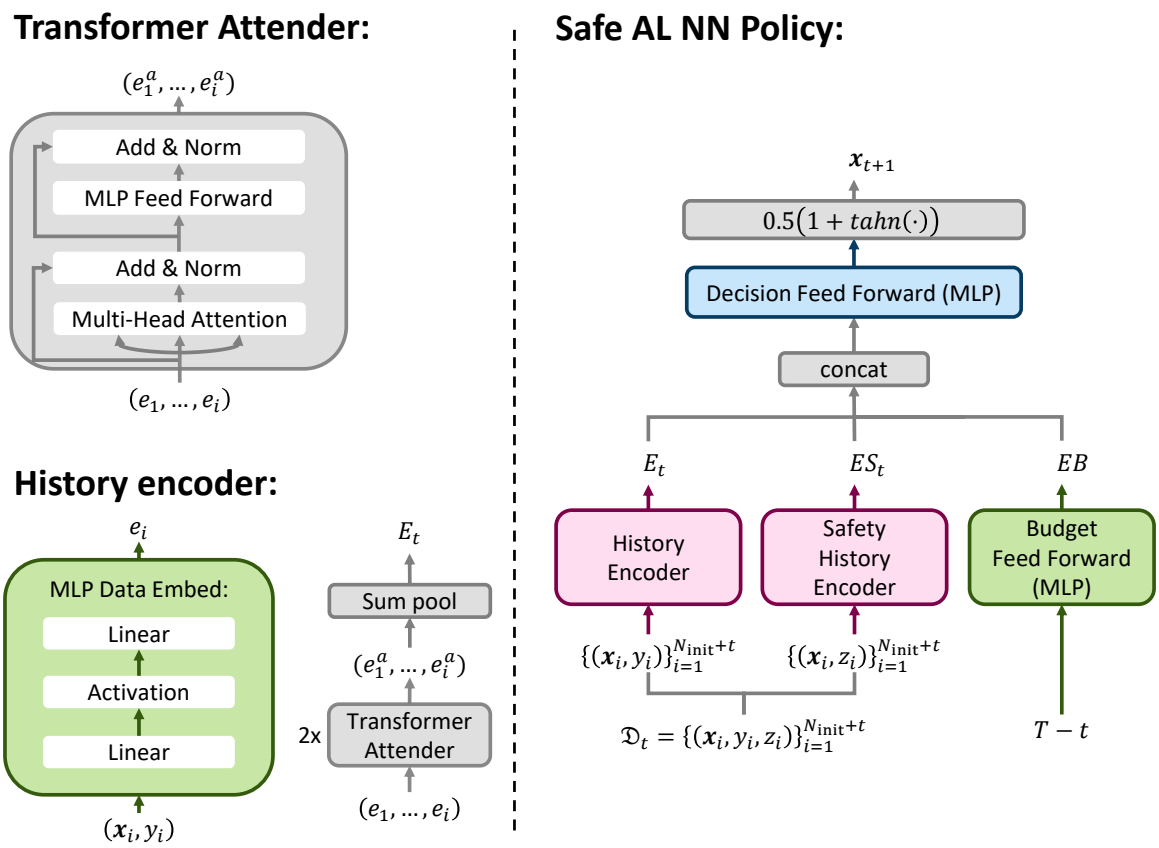

Figure S.B.1: **Our NN** $\phi(T-t, \mathcal{D}_t) = \boldsymbol{x}_{t+1}$. Each observation pair $(\boldsymbol{x}_t, y_t)$ (or $(\boldsymbol{x}_t, z_t)$) is first mapped by an MLP to an embedding e_t. Data of different t use the same MLP Data Embed module, i.e. one history encoder has only one MLP Data Embed. A sequence of data embeddings is then stacked and attended by a 2-layer transformer encoder (Vaswani et al., 2017), and then a final history embedding is obtained by summing up the attended per-data embeddings $\sum_t e_t^a$. The sum pool ensures an invariance of input data order. Details of the history encoder are described in Foster et al. (2021); Ivanova et al. (2021). The history encoder and safety history encoder are two modules of individual parameters. The budget variable is as well mapped by a separate MLP module to a budget embedding EB. The history embedding E_t, safety history embedding ES_t, and the budget embedding EB are stacked, mapped by another MLP module, and then refrained by a *tahn* function to a bounded input domain $[0, 1]^D$.

B Policy NN Structure

We build our NN upon the structure described in Foster et al. (2021); Ivanova et al. (2021). The final structure is sketched in Figure S.B.1. We summarize the comparison between our network and the one developed by Ivanova et al. (2021):

1. the history encoder ($\{(\boldsymbol{x}_i, y_i)\}_{i=1}^t \to E_t$) and the decision feed forward MLP (originally $E_t \to \boldsymbol{x}_{t+1}$) are taken from Ivanova et al. (2021);

2. we add a hyperbolic tangent function as the last layer to ensure the policy output is in our bounded $\mathcal{X}$, which was not needed in the original BED problems;

3. we add another history encoder to handle the safety data;

4. we add a budget encoder to handle the budget variable.

Note that the history encoder incorporates the inductive bias that observed data are order-invariant (see Foster et al. (2021); Ivanova et al. (2021) for details). This can be seen by noticing that a conventional AL computes the acquisition score conditioned on the past observations and the order of the past data does not matter. We leverage the same architecture to encode the safety embedding because our safety likelihoods follows the same invariance (future likelihood conditioned on the past, see Eq. (6)). We stack the history embedding, the safety history embedding, and the budget to reflect that the querying decision is made by balancing the three of them

Our NN structure is highly modularized. The safety history encoder can be removed to form a budget-aware unconstrained AL policy. If we intend to work on a policy under fixed budget, i.e. T_{sim} is fixed to T in Algorithm 1, we can remove the Budget Feed Forward module.

For the numerical settings, please see Appendix F.1.

C Training Objectives: Details, Illustrations, More Objectives

In this section, we provide

- an illustration of our entropy objective, another version of our entropy objective and an illustration (Appendix C.1),
- details of our regularized entropy (mutual information) objective, another version of our regularized entropy objective (Appendix C.1),
- details of the DAD baseline in our GP context (Appendix C.2),
- an illustration of our safe AL objective, an additional safe AL objective and an illustration (Appendix C.3).

This section inherits the notation of Section 3.1. The objectives are computed with simulated (safe) AL instances.

C.1 Objectives: Unconstrained AL

In this section, we provide an illustration of our main unconstrained AL entropy objective, details of our regularized entropy, and additional objectives.

Note first that our unconstrained AL training does not need the safety measurements: a NN may have no safety history encoder (Figure S.B.1), and the training algorithm can operate with neither safety measurements q and z nor safety center $\mathcal{C}$. Therefore, our main safe AL training (Algorithm 1) can be reduced to Algorithm S.1.

Algorithm S.1 Unconstrained AL Policy Training

Require: Assumption 2.1, T, N_{init}
draw a batch of (θ, σ^2)
draw a batch of $(f, \mathcal{D}_0)$ per Algorithm S.2
for each $(f, \mathcal{D}_0)$ **do**
  draw $T_{\text{sim}} \sim \text{Uniform}[1, T]$
  for $t = 1, ..., T_{\text{sim}}$ **do**
    $\boldsymbol{x}_{\phi,t} = \phi(T_{\text{sim}} - t + 1, \mathcal{D}_{t-1})$
    draw $\epsilon_t \sim \mathcal{N}(0, \sigma^2), y_{\phi,t} = f(\boldsymbol{x}_{\phi,t}) + \epsilon_t$
    $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{\boldsymbol{x}_{\phi,t}, y_{\phi,t}\}$
  end for
end for
compute AL loss (e.g. Eq. (5)), update ϕ

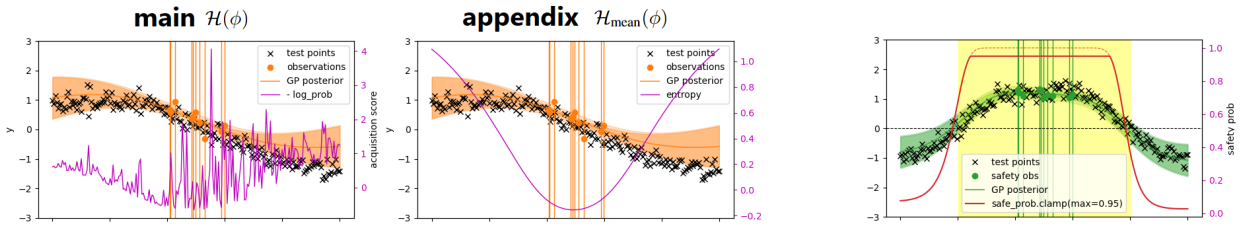

Figure S.C.2: **Simulated unconstrained AL objectives and safety probability.** $T_{\text{sim}} = 1$. Given observed data (y orange, z green), we plot GP posteriors (orange, Eq. (S.2)), compute AL objectives (pink) and the safety likelihood $p(z(\cdot) \geq 0 | z_{\text{observed}})$ (red, Eq. (S.3)) of the next queries (black). The yellow region is the true safe area. The objectives illustrated are meant to be maximized. Note that y, z are available because our training scheme simulates AL before utilizing the evaluations for objective computations. A conventional (safe) AL optimizes queries based on inferred future evaluations.

Algorithm S.2 Unconstrained Initial Sampling

Require: Assumption 2.1, N_{init}, $\mathcal{GP}(0, k_\theta), \sigma^2$
1: draw $f_{\text{raw}} \sim \mathcal{GP}(0, k_\theta)$
2: $f(\cdot) = f_{\text{raw}}(\cdot) - \mathbb{E}_{\boldsymbol{x} \in \mathcal{X}}[f_{\text{raw}}(\boldsymbol{x})]$
3: draw $\boldsymbol{X}_{\text{init}} \sim \text{Uniform}[\mathcal{X}], |\boldsymbol{X}_{\text{init}}| = N_{\text{init}}$
4: draw $\mathcal{E}_{\text{init}} \sim \mathcal{N}(0, \sigma^2 I)$, $Y_{\text{init}} = f(\boldsymbol{X}_{\text{init}}) + \mathcal{E}_{\text{init}}$
5: $\mathcal{D}_0 = \{\boldsymbol{X}_{\text{init}}, Y_{\text{init}}\}$
6: **return** $f, \mathcal{D}_0$

C.1.1 Objectives: Unconstrained AL - Entropy

Here, we illustrate our main entropy objective, and we introduce another version of the entropy objective.

In our main paper, we introduce the unconstrained AL meta objective

$$\mathcal{H}(\phi) \propto \mathbb{E}_{\theta,\sigma^2} \mathbb{E}_{f, T_{\text{sim}}, \mathcal{E}_{\text{init}}, \epsilon_{1:T_{\text{sim}}}} \left[\frac{-\log p(y_{\phi,1}, ..., y_{\phi,T_{\text{sim}}} | Y_{\text{init}})}{N_{\text{init}} + T_{\text{sim}}} \right] \text{ (Eqs. (2) and (4)).}$$

Here, we introduce another objective function which computes

$$\mathcal{H}_{\text{mean}}(\phi) = \mathbb{E}_{\theta,\sigma^2} \mathbb{E}_{f, T_{\text{sim}}, \mathcal{E}_{\text{init}}, \epsilon_{1:T_{\text{sim}}}} \left[\frac{\mathbb{H}(y(\boldsymbol{x}_{\phi,1}), ..., y(\boldsymbol{x}_{\phi,T_{\text{sim}}}) | Y_{\text{init}})}{N_{\text{init}} + T_{\text{sim}}} \right]. \tag{S.6}$$

Note that entropy $\mathbb{H}$ takes random variables, not the sampled $y_{\phi,1}, ..., y_{\phi,T_{\text{sim}}}$. Substituting Eqs. (S.4) and (S.5) into the objectives, we see that the key difference (marked blue) is indeed whether the observation values $y_{\phi,1}, ..., y_{\phi,T}$ are taken into account. Our main $\mathcal{H}$ aims for points $y_{\phi,1}, ..., y_{\phi,T}$ that are the most distinctive, while $\mathcal{H}_{\text{mean}}$ aims for $\boldsymbol{x}_{\phi,1}, ..., \boldsymbol{x}_{\phi,T}$ that have the most uncertainty jointly (no actual output values). We suspect that having $y_{\phi,1}, ..., y_{\phi,T}$ in the loss (our main $\mathcal{H}$, Eq. (4)) may help the policy adapt in an AL deployment. Please see Figure S.C.2 for illustrations. One can see that $\mathcal{H}$ has the evaluated $y_{\phi,1:T_{\text{sim}}}$ which encodes output values and noises.

The subscript name is "mean" because $\mathbb{H}$ computes the expectation of negative log likelihood.

Note that, if we would obtain each $\boldsymbol{x}_{\phi,t+1}$ step-wise, the objectives $\mathcal{H}, \mathcal{H}_{\text{mean}}$ correspond to the following acquisition function:

$$\begin{aligned} a_{\mathcal{H}}(\boldsymbol{x} | y_{\phi,1:t}, Y_{\text{init}}) &= -\log p(\hat{y}(\boldsymbol{x}) | y_{\phi,1:t}, Y_{\text{init}}), \\ a_{\mathcal{H}_{\text{mean}}}(\boldsymbol{x} | y_{\phi,1:t}, Y_{\text{init}}) &= \mathbb{E}_{y(\boldsymbol{x}) | y_{\phi,1:t}, Y_{\text{init}}} [-\log p(y(\boldsymbol{x}) | y_{\phi,1:t}, Y_{\text{init}})] \\ &= \mathbb{H}(y(\boldsymbol{x}) | y_{\phi,1:t}, Y_{\text{init}}). \end{aligned} \tag{S.7}$$

Here, $y(\boldsymbol{x})$ is a random variable forecasting into the next query, while $\hat{y}(\boldsymbol{x})$ is an estimated value. In a conventional scenario where the acquisition function measures the unqueried future point, one may take the GP predictive mean for $\hat{y}(\boldsymbol{x})$ resulting in $a_{\mathcal{H}}(\boldsymbol{x}|y_{\phi,1:t}, Y_{\text{init}}) = -1/2 \log e + \mathbb{H}(y(\boldsymbol{x})|y_{\phi,1:t}, Y_{\text{init}}) \propto \mathbb{H}(y(\boldsymbol{x})|y_{\phi,1:t}, Y_{\text{init}})$ (Eqs. (S.2), (S.4) and (S.5)). In our simulation, $\hat{y}(\boldsymbol{x})$ would be the simulated noisy realization on the entire domain (not just at $\boldsymbol{x}_{\phi,t}$ but all $\boldsymbol{x} \in \mathcal{X}$) where $\hat{y}(\boldsymbol{x}_{\phi,t+1}) = y_{\phi,t+1}$.

$\mathbb{H}(y(\boldsymbol{x})|y_{\phi,1:t}, Y_{\text{init}})$ is the standard predictive entropy (Seo et al., 2000; Krause and Guestrin, 2007).

We refer the readers to Krause and Guestrin (2007); Krause et al. (2008) for detailed discussions of conventional GP AL methods. In these papers, the authors discuss sequential step-wise querying and joint batch querying. Our training scheme can be considered as a special case where the policy performs sequential querying while the objectives compute joint batch querying but with true evaluations.

C.1.2 Objectives: Unconstrained AL - Regularized Entropy

Here, we discuss empirical issues of unconstrained entropy and our approach using mutual information objectives (Guestrin et al., 2005; Krause et al., 2008).

Maximizing an entropy objective favors a set of distinctive points, which naturally encourages points at the border, as they are the most scattered. Guestrin et al. (2005) argued that exploring the border may be suboptimal because we would rather explore inside the space. The authors further proposed a mutual information acquisition criterion to tackle this problem, at least in conventional AL settings. In our training scheme, the objectives $\mathcal{H}(\phi)$ and $\mathcal{H}_{\text{mean}}(\phi)$ sometimes overemphasize the border and completely ignore the inner region of $\mathcal{X}$, which is clearly not desired. Part of the reasons can be the tanh layer in our NN (Figure S.B.1), which, although helping us bound the NN outputs in $\mathcal{X}$, has vanishing gradients at the boundary. To tackle this issue, we wish to derive a mutual information objective suitable for our framework.

In active GP learning, Guestrin et al. (2005) define a mutual information criterion as the reduction of entropy in an unexplored region: $\mathbb{H}(y(\boldsymbol{x}_{\phi,1}), ..., y(\boldsymbol{x}_{\phi,T})|Y_{\text{init}}) - \mathbb{H}(y(\boldsymbol{x}_{\phi,1}), ..., y(\boldsymbol{x}_{\phi,T})|Y_{\text{init}}, \hat{y}(\mathcal{X} \setminus \boldsymbol{X}_{\phi}))$, $\hat{y}(\mathcal{X} \setminus \boldsymbol{X}_{\phi})$ means the output estimates corresponding to $\mathcal{X} \setminus \{\boldsymbol{x}_{\phi,1}, ..., \boldsymbol{x}_{\phi,T}\}$. The original paper considered small discrete $\mathcal{X}$ where $\hat{y}(\mathcal{X} \setminus \{\boldsymbol{x}_{\phi,1}, ..., \boldsymbol{x}_{\phi,T}\})$ is a computable set of variables. In our framework, $\mathcal{X}$ is a continuous space, and thus this is not well-defined. Even if $\mathcal{X}$ is discrete, conditioning on a large pool (fine discretization) is computationally heavy, i.e. GP cubic complexity $\mathcal{O}\left(|\mathcal{X}|^3\right)$ (Appendix A). A discrete pool also enforces a classifier-like policy ϕ, as the policy needs to select points from a pool, which prohibits us from utilizing the current NN structure developed on top of Foster et al. (2021); Ivanova et al. (2021).

To derive mutual information objectives suitable for our learning framework, note that entropy reduction is a regularized entropy objective. We propose a simple yet effective approach: compute the regularization term only on a sparse set of N_{grid} samples $(\boldsymbol{X}_{\text{grid}}, Y_{\text{grid}}) \subseteq \mathcal{X} \times \mathcal{Y}$. Then we turn the acquisition functions proposed by Guestrin et al. (2005); Krause and Guestrin (2007) into the following training objectives

$$\begin{aligned}
\mathcal{I}(\phi) =& \mathbb{E}\left[\frac{-\log p(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}}) + \log p(y_{\phi,1:T_{\text{sim}}}|Y_{\text{init}}, Y_{\text{grid}})}{N_{\text{init}} + T_{\text{sim}}}\right] \text{ (Eqs. (3) and (5))}, \\
\mathcal{I}_{\text{mean}}(\phi) =& \mathbb{E}\left[\frac{\mathbb{H}(y(\boldsymbol{x}_{\phi,1:T_{\text{sim}}})|Y_{\text{init}}) - \mathbb{H}(y(\boldsymbol{x}_{\phi,1:T_{\text{sim}}})|Y_{\text{init}}, Y_{\text{grid}})}{N_{\text{init}} + T_{\text{sim}}}\right].
\end{aligned} \tag{S.8}$$

The expectation is over GPs (θ, σ^2) and AL functions $(f, T_{\text{sim}}, \mathcal{E}_{\text{init}}, \epsilon_{1:T_{\text{sim}}})$. N_{grid} should be much larger than T. Maximizing these objectives encourages $\{\boldsymbol{x}_{\phi,1}, ..., \boldsymbol{x}_{\phi,T}\}$ to track subsets of $\boldsymbol{X}_{\text{grid}}$. The intuition is two-fold: (i) we can view them as entropy objectives regularized by an additional search space indicator, or (ii) we can view them as imitation objectives because a subset of grid points, if happens to have small joint likelihood or large joint entropy, maximizes the objective.

Note that, to keep the policy from overfitting those sparse grid samples, which are not necessarily optimal points, we re-sample $\boldsymbol{X}_{\text{grid}}$ in each training step. We sample $\boldsymbol{X}_{\text{grid}}$ with Beta$(0.5, 0.5)$, but a uniform distribution can also be used and we did not see obvious difference. Empirically, training with the two mutual information objectives are easier to converge than the entropy objectives. Entropy objectives often stick in border-only patterns, particularly when $D \geq 2$.

One drawback of such objectives is that we lack insight into the appropriate number of the grid points. When the dimension of $\mathcal{X}$ grows, N_{grid} might need to be increased, which can make our mutual information objectives

computationally impossible. We provide our exact N_{grid} in Table S.E.1.

Our ablation study in Appendix G demonstrate the results of $\mathcal{I}_{\text{mean}}$.

Note that this regularization is meaningless to safe AL, because the border of $\mathcal{X}$ is typically less safe and the safe AL objectives already reflect this (Figure S.C.3)

C.2 Objectives: DAD Baseline

Algorithm S.3 DAD Training with GP Prior

Require: Assumption 2.1, T, N_{init}
draw a batch of (θ, σ^2)
for each (θ, σ^2) **do**
  draw $f_{0,\text{raw}} \sim \mathcal{GP}(0, k_\theta)$
  $f_0(\cdot) = f_{0,\text{raw}}(\cdot) - \mathbb{E}_{\boldsymbol{x}\in\mathcal{X}}[f_{0,\text{raw}}(\boldsymbol{x})]$
  draw $\boldsymbol{X}_{\text{init}} \sim \text{Uniform}[\mathcal{X}], |\boldsymbol{X}_{\text{init}}| = N_{\text{init}}$
  draw $\mathcal{E}_{\text{init}} \sim \mathcal{N}(0, \sigma^2 I), Y_{\text{init}} = f_0(\boldsymbol{X}_{\text{init}}) + \mathcal{E}_{\text{init}}$
  $\mathcal{D}_0 = \{\boldsymbol{X}_{\text{init}}, Y_{\text{init}}\}$
  for $t = 1, ..., T$ **do**
    $\boldsymbol{x}_{\phi,t} = \phi(\mathcal{D}_{t-1})$
    draw $\epsilon_t \sim \mathcal{N}(0, \sigma^2), y_{\phi,t} = f_0(\boldsymbol{x}_{\phi,t}) + \epsilon_t$
    $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{\boldsymbol{x}_{\phi,t}, y_{\phi,t}\}$
  end for
  draw $f_{l,\text{raw}} \sim \mathcal{GP}(0, k_\theta), l = 1, ..., N_{f,q}$
  $f_l(\cdot) = f_{l,\text{raw}}(\cdot) - \mathbb{E}_{\boldsymbol{x}\in\mathcal{X}}[f_{l,\text{raw}}(\boldsymbol{x})], l = 1, ..., N_{f,q}$
end for
compute DAD loss (Eq. (S.9)), update ϕ

We describe the DAD baseline in our GP context (Foster et al., 2021). DAD is a scheme which samples tasks and learns to perform Bayesian optimal experimental design originally aiming at parametric models. Bayesian optimal experimental design aims to query data to learn about a posterior Bayesian model (Rainforth et al., 2024). In a GP learning problem, this means collecting $\boldsymbol{x}_{1:T}, y_{1:T}$ to make inference $p(f(\cdot)|\boldsymbol{X}_{\text{init}}, Y_{\text{init}}, \boldsymbol{x}_{1:T}, y_{1:T})$. DAD follows similar training procedure: we (i) sample tasks from a prior, and (ii) learn a data querying policy by optimizing the DAD objective. To train with DAD, one may use our main training algorithm because we generate more quantities than needed by the DAD objective. For clarity, however, we write down a clean version of DAD training in Algorithm S.3, which is the Algorithm 1 of Foster et al. (2021) combined with our GP prior. We summarize the comparison between DAD and our main safe AL training (Algorithm 1): (i) safety measurements are not present in DAD, (ii) the policy of DAD does not take the budget variable while $T_{\text{sim}} = T$ is fixed during the training, and (iii) functions $f_0, ..., f_{N_{f,q}}$ are sampled but the DAD objective simulates the observations $Y_{\text{init}}, y_{1:T}$ on f_0 only while $f_1, ..., f_{N_{f,q}}$ are contrastive samples for noise contrastive estimations. Note that in this paper, $N_{f,q}$ denotes the number of functions $|\{(f, q)\}|$ per set of GP hyperparameters. Here, we use $N_{f,q}$ to denote the number of contrastive functions.

The objective is

$$DAD(\phi) = \mathbb{E}_{\theta,\sigma^2}\left[\mathbb{E}_{f_0,...,f_{N_{f,q}},\mathcal{E}_{\text{init}},\epsilon_{1:T}}\left[\log \frac{p(Y_{\text{init}}, y_{\phi,1:T}|f_0)}{1/(N_{f,q}+1)\sum_{l=0}^{N_{f,q}} p(Y_{\text{init}}, y_{\phi,1:T}|f_l)}\right]\right]. \tag{S.9}$$

This objective is meant to be maximized.

This objective was derived for parametric models, i.e. $p(\text{model parameters}|\text{data})$ in contrast to our $p(f|\text{data})$. Please see Foster et al. (2021) for details.

Notice that a further work, Ivanova et al. (2021), extended DAD to more general scenarios, for example, allowing intractable $p(y|f)$. However, their proposed learning objectives require another NN mapping, $\{(\mathcal{D}, f)\} \to \mathbb{R}$ if described in our notation, to help estimate the involved intractable distributions. Such a mapping is not

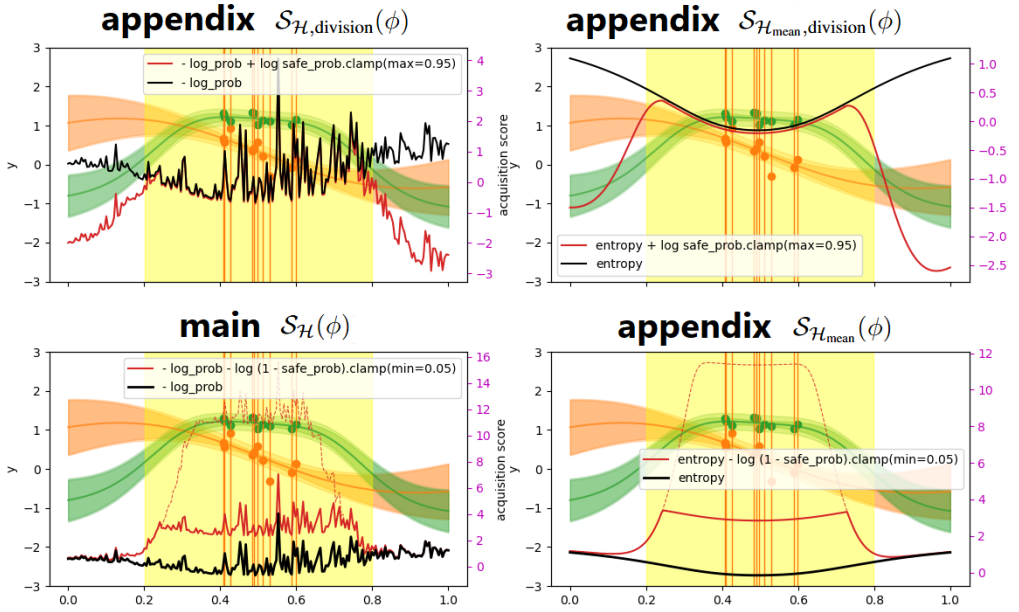

Figure S.C.3: **Simulated safe AL objectives.** $T_\text{sim} = 1$. As described in Figure S.C.2, orange and green are y, z and the corresponding GP predictions, while black curves are unconstrained objectives $\mathcal{H}, \mathcal{H}_\text{mean}$. Our safe AL objectives (red) decorate the unconstrained objectives with our main min unsafe log probability (bottom, $\mathcal{S}_\cdot$, Eq. (7)) or with the appendix max safe log probability (top, $\mathcal{S}_{\cdot,\text{division}}$, Eq. (S.10)). The objectives illustrated are meant to be maximized. Red dashed lines show the objectives if the safety probability is not clamped by γ. Note again that y, z are available because our training scheme simulates AL before utilizing the evaluations for objective computations.

applicable for nonparametric f, and the objectives in Ivanova et al. (2021) were again not designed for AL of nonparametric functions. This is a different direction to what we need in this paper.

C.3 Objectives: Safe AL

We illustrate our main safe AL objective, and we introduce more objectives.

In our main paper, we propose a safe AL objective (Eq. (7))

$$\mathcal{S}_{\mathcal{H}}(\phi) = \mathbb{E}\left[\frac{-\log p(y_{\phi,1}, ..., y_{\phi,T_\text{sim}}|Y_\text{init}) - \sum_{t=0}^{T_\text{sim}-1} \log\ \max(\gamma, p(z(\boldsymbol{x}_{\phi,t+1}) < 0|z_{\phi,1:t}, Z_\text{init}))}{N_\text{init} + T_\text{sim}}\right].$$

This objective is a combination of an unconstrained exploration objective and a safety regularization. We are able to exchange the exploration term $-\log p(y_{\phi,1}, ..., y_{\phi,T_\text{sim}}|Y_\text{init})$ by any other objectives, e.g. $\mathcal{S}_{\mathcal{H}_\text{mean}}(\phi)$ corresponds to safety decorated $\mathcal{H}_\text{mean}(\phi)$.

If we take a closer look into the safety term, the objective is maximized when $\log p(z(\boldsymbol{x}_{\phi,t+1}) < 0|z_{\phi,1:t}, Z_\text{init})$ is small. Ideally, the probability of being unsafe should be small (or equivalently the probability of being safe is large, Figure S.C.2). Minimizing the unsafe probability nevertheless makes the log value explode to negative infinity (if not clamped by a non-zero γ). Numerically, we suggest to add a small number to the probability to stabilize the computations; e.g., we add 10^{-5} for clamped and non-clamped versions for stability.

Nevertheless, another approach we consider is to change the safety term, so that we directly maximize the safety

probability

$$
\begin{aligned}
&- \log p(y_{\phi,1}, ..., y_{\phi,T_{\text{sim}}} | Y_{\text{init}}) + \sum_{t=0}^{T_{\text{sim}}-1} \log p(z(\boldsymbol{x}_{\phi,t+1}) \geq 0 | z_{\phi,1:t}, Z_{\text{init}}) \\
&= - \sum_{t=0}^{T_{\text{sim}}-1} \log \frac{p(y_{\phi,t+1} | y_{\phi,1:t}, Y_{\text{init}})}{p(z(\boldsymbol{x}_{\phi,t+1}) \geq 0 | z_{\phi,1:t}, Z_{\text{init}})},
\end{aligned}
$$

which corresponds to

$$
\mathcal{S}_{\mathcal{H},\text{division}}(\phi) = \mathbb{E}\left[\frac{- \log p(y_{\phi,1}, ..., y_{\phi,T_{\text{sim}}} | Y_{\text{init}}) + \sum_{t=0}^{T_{\text{sim}}-1} \log p(z(\boldsymbol{x}_{\phi,t+1}) \geq 0 | z_{\phi,1:t}, Z_{\text{init}})}{N_{\text{init}} + T_{\text{sim}}} \right]. \tag{S.10}
$$

Note that $z(\boldsymbol{x}_{\phi,t+1})$ is a prediction, not the evaluated $z_{\phi,t+1}$. Similarly, we may exchange the base exploration term to get e.g. $\mathcal{S}_{\mathcal{H}_{\text{mean}},\text{division}}(\phi)$. With this objective, we do not clamp the safety probability with γ because this has no visible effect (see Figure S.C.3). The reason is, if we would clamp the likelihood, i.e. $\min(1-\gamma, p(z(\boldsymbol{x}_{\phi,t+1}) \geq 0 | z_{\phi,1:t}, Z_{\text{init}})$ (maximizing the safety likelihood until $1-\gamma$), then the clamped term is optimized towards $p(z(\boldsymbol{x}_{\phi,t+1}) \geq 0 | z_{\phi,1:t}, Z_{\text{init}}) \geq 1-\gamma$ but for small γ, $\log(1-\gamma) \approx \log 1$.

Please see Figure S.C.3 for an illustration. Our main $\mathcal{S}_{\cdot}$ is more conservative at the safe set border while the appendix $\mathcal{S}_{\cdot,\text{division}}$ is more prone to space exploration.

In this paper, we avoid coupling $\mathcal{I}, \mathcal{I}_{\text{mean}}$ with our safety terms because (i) it is rather tricky to sample $\{\boldsymbol{X}_{\text{grid}}, Y_{\text{grid}}, Z_{\text{grid}}\}$ (Eqs. (5) and (S.8)) when a safety constraint is present, (ii) we do not see an obvious intuition behind such safe AL objectives, and (iii) we do not see empirical benefit in our preliminary experiments (not shown in the paper).

We illustrate the safe AL objectives in Figure S.C.3 for $T_{\text{sim}} = 1$.

D Training: GP Function Sampling

This section outlines the mathematical details of our GP function sampling, i.e. Algorithm 2.

We first introduce our GP kernel. In our paper, we always use an RBF kernel, including k_θ, k_q in Algorithm 2 and the GP kernel in benchmark experiments. An RBF kernel has $D+1$ variables: the variance v and a D dimension lengthscale vector $\boldsymbol{l} = (l_1, ..., l_D)$

$$
k(\boldsymbol{x}, \boldsymbol{x}') = v \exp\left(-1/2 \sum_{d=1}^{D} \left(\frac{[\boldsymbol{x} - \boldsymbol{x}']_d}{l_d} \right)^2 \right). \tag{S.11}
$$

D.1 Fourier Feature Functions

In Algorithm 2, the GP functions $f_{\text{raw}} \sim \mathcal{GP}(0, k_\theta), q_{\text{raw}} \sim \mathcal{GP}(0, k_q)$ are approximated by Fourier features, which means each function sample is a linear combination of cosine functions (Rahimi and Recht, 2007; Wilson et al., 2020):

$$
\text{e.g. } f_{\text{raw}}(\boldsymbol{x}) = \sum_{i=1}^{L} \omega_i \sqrt{2/L} \cos\left(\boldsymbol{a}_i^T \boldsymbol{x} + b_i \right),
$$

$\omega_i \sim \mathcal{N}(0, \sqrt{v}^2), \boldsymbol{a}_i \sim \mathcal{N}(0, \text{diag}\{\boldsymbol{l}\}^{-1})$ are kernel dependent and $b_i \sim \text{Uniform}(0, 2\pi)$. Each function has $L*(D+2)$ parameters. Larger L lead to better approximations. An error bound w.r.t. D and L can be seen in Rahimi and Recht (2007). We set $L = 100$.

Remark D.1 (running out of symbols)**.** We highlight that the parameters of our Fourier feature functions, $\omega_i, \boldsymbol{a}_i$ and b_i, are independent of our acquisition function notation $a(\cdot)$ or other constants. The Fourier feature function parameters (except for L, the number of features) are used only for this subsection and will not be referred in any other part of the paper.

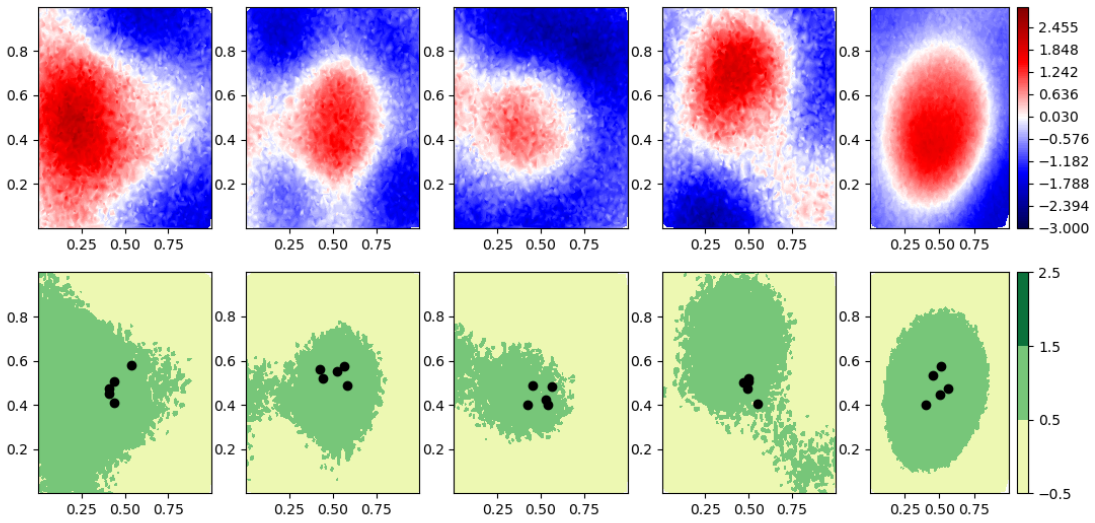

Figure S.D.4: **Examples of sampled** $q \sim \mathcal{GP}(\mu_q, k_q)$. We run Algorithm 2 to sample f (not shown) and q. The observations are blurred with noises. The top are five samples of $z\left([0,1]^2\right)$, the bottom safe and unsafe classifications $z\left([0,1]^2\right) \geq 0$. Black dots are the sampled initial data. The parameters of μ_q, k_q leverage samplers listed in Table S.F.3. Our algorithm samples diverse safety patterns while guaranteeing stable initial safe data sampling.

In our main paper, the analytical mean of window $\mathcal{X} = [0,1]^D$ is computed such that all functions can be shifted to zero mean in the particular domain. The analytical mean is the integral of $f_{\text{raw}}(\boldsymbol{x}), q_{\text{raw}}(\boldsymbol{x})$ divided by volume of $[0,1]^D$. We give examples for the one and two dimensional cases:

$$\sum_{i=1}^{L} \frac{1}{a_i} \omega_i \sqrt{2/L} \sin\left(a_i x + b_i\right) |_{x=0}^{1}, \text{ for } D = 1,$$

$$\sum_{i=1}^{L} \frac{-1}{[\boldsymbol{a}_i]_1 [\boldsymbol{a}_i]_2} \omega_i \sqrt{2/L} \cos\left([\boldsymbol{a}_i]_1 x_1 + [\boldsymbol{a}_i]_2 x_2 + b_i\right) |_{x_1=0}^{1} |_{x_2=0}^{1}, \text{ for } D = 2,$$

higher dimensions are analogous.

It may happen that at least one component of $\boldsymbol{a}_i$ is zero or is close to zero, which causes a problem in the division. In this case, we replace $|1/\prod_{d=1}^{D} [\boldsymbol{a}_i]_d|$ by 100000. The error is negligible, i.e. much smaller than noise level. We can replace $[0,1]^D$ by any windows of our interest.

The complexity of computing such mean is $\mathcal{O}\left(L 2^D\right)$, and this can be distributed to time or space. With common GP and RBF kernel dimensions, e.g. $D \leq 5$ or 10, this complexity is negligible.

D.2 Prior Mean of Safety Functions

As described in our main Section 3.2, we use a GP mean function μ_q to help us control the safety function sampling.

The overall goal is to sample functions $q \sim \mathcal{GP}_{\theta_q}(\mu_q, k_q)$, while guaranteeing a high probability of the existence of central safe data. We design μ_q such that it is safe at the center of the domain $\mathcal{X}$. We use the hyperbolic

Table S.E.1: **Batch sizes in training.**

loss functions	$\mathcal{I}$	DAD	$\mathcal{S}_{\mathcal{H}}$
$N_k = \lvert\{(\theta, \theta_q)\}\rvert$	10	10	10
$N_{f,q} = \lvert\{(f, q)\}\rvert$	5	200	5
$B = \lvert\{(\mathcal{E}_{\text{init}}, \epsilon_{1:T}, \mathcal{E}_{q,\text{init}}, \epsilon_{q,1:T})\}\rvert$	10	10	1
L =num of fourier features	100	100	100
N_{grid} (number of $\boldsymbol{X}_{\text{grid}}$)	100 (for NN $D \leq 2$) 500 (for NN $D = 5$)	N/A	N/A

secant function

$$\mu_q(\boldsymbol{x}) = 3.2c\left(-0.47 + \operatorname{sech}\left(\frac{1}{D}\sum_{d=1}^{D} w_d[\boldsymbol{u}]_d^2\right)\right),$$
$$\boldsymbol{u} = Q^T\left(\boldsymbol{x} - \begin{pmatrix}0.5\\ \vdots \\ 0.5\end{pmatrix}\right), \tag{S.12}$$

$Q \in \mathbb{R}^{D\times D}$ is an orthogonal matrix where columns of Q are orthonormal, each $w_d > 0$ is a shape parameter. We provide our design logic and the chosen parameters of this function step by step:

1. Consider $Q = I_D$, then $\frac{1}{D}\sum_{d=1}^{D} w_d[\boldsymbol{u}]_d^2 = \frac{1}{D}\sum_{d=1}^{D} w_d([\boldsymbol{x}]_d - 0.5)^2$ is an ellipsoid centering around $(0.5, ..., 0.5) \in \mathbb{R}^D$.

2. We can see that μ_q has the center area being a safe ellipsoid as long as $c > 0$, with shape and size controlled by w_d, and the orthogonal matrix Q allows us to rotate the ellipsoid around the center $(0.5, ..., 0.5)$. The orthogonal matrix Q is obtained by performing a QR-decomposition of a sampled $A \in \mathbb{R}^{D\times D}$ (each entity from Uniform$[-1, 1]$).

3. The above steps describe variables w_d, c, Q, and we then describe the constants.

4. If we consider $c = 1, w_d/D = 10, \forall d \leq D$ (e.g. $w_d = 20, D = 2$) and $Q = I_D$, then the central safe area is a ball and it takes about half of the space, i.e. the mean function μ_q brings half of the space safe and half unsafe. We will later sample the shape and the half-safe space is only for an initial design.

5. With the same c, w_d, Q, the constants 3.2 and -0.47 ensure zero mean and unit variance of this μ_q function, which aligns with our setup that the deployment problems are normalized, and this provides us an estimated variance of $\mu_q \approx c^2$.

When we sample $q = \mu_q + q_{\text{raw}} - \mathbb{E}_{\boldsymbol{x}\in\mathcal{X}}[q_{\text{raw}}(\boldsymbol{x})], q_{\text{raw}} \sim \mathcal{GP}(0, k_q)$ (see Algorithm 2), we aim for mean $\mathbb{E}[q]$ around 0 and variance of q around 1, as our deployment problems. We distribute the variance by making sure $c^2 + v + \sigma_q^2$ is around 1 (kernel variance v, observation noise level σ_q), in particular, we set $c^2 = 0.5$ while the kernel and the noise take the remaining half. The kernel and the noise are the main sources of functional stochasticity. The assumption of having additive variances is in fact not true for our μ_q, but this setting is enough for the NN to learn. One may also consider the overall function variance as an amplitude of the function. The shape parameters $\boldsymbol{w} = (w_1, ..., w_D)$ are sampled around value 20, such that μ_q has around 10% to 100% of the space above safety threshold 0. We will summarize the sampling setting later in Appendix F.

We illustrate examples of sampled q in Figure S.D.4.

E Training Complexity

Our loss functions take expectation over GP hyperparameters and functions. We summarize the batch sizes in Table S.E.1. Note that for each set of GP hyperparameters, the output realizations consist of noise-free f, q and noise realizations, denoted individually. The number of noise realizations, B, can be considered as the repetitions of simulated safe AL per function pair (f, q).

Table S.E.2: **Loss function complexities.** The batch sizes $B, N_k, N_{f,q}$ contribute linearly in time or space (but not both), and this is up to the implementation. We derive the worst case $T_{\text{sim}} = T$.

	time	space
$p(\cdot\|Y_{\text{init}})$ (Eq. (S.2), plug $y_{\phi,1:T}$ in later) $p(\cdot\|Y_{\text{init}}, Y_{\text{grid}})$ (Eq. (S.2)) $\log p(z(\boldsymbol{x}_{\phi,T}) < 0\|z_{\phi,1:T-1}, Z_{\text{init}})$ (Eqs. (S.2) and (S.3))	$\mathcal{O}\left(N_{\text{init}}^3\right)$ $\mathcal{O}\left((N_{\text{init}} + N_{\text{grid}})^3\right)$ $\mathcal{O}\left((N_{\text{init}} + T)^3\right)$	$\mathcal{O}\left(N_{\text{init}}^2\right)$ $\mathcal{O}\left((N_{\text{init}} + N_{\text{grid}})^2\right)$ $\mathcal{O}\left((N_{\text{init}} + T)^2\right)$
$\mathcal{H}$ (Eqs. (2) and (4))	$\mathcal{O}\left(N_{\text{init}}^3\right) + \mathcal{O}\left(T^3\right)$	$\mathcal{O}\left(N_{\text{init}}^2\right) + \mathcal{O}\left(T^2\right)$
$\mathcal{I}$ (Eqs. (3) and (5))	$\mathcal{O}\left((N_{\text{init}} + N_{\text{grid}})^3\right) + \mathcal{O}\left(T^3\right)$	$\mathcal{O}\left((N_{\text{init}} + N_{\text{grid}})^2\right) + \mathcal{O}\left(T^2\right)$
$\mathcal{S}_{\mathcal{H}}$ (Eq. (7))	$\mathcal{O}\left(N_{\text{init}}^3\right) + \mathcal{O}\left(T^3\right)$ $+ \mathcal{O}\left((N_{\text{init}} + T)^3\right)$	$\mathcal{O}\left(N_{\text{init}}^2\right) + \mathcal{O}\left(T^2\right)$ $+ \mathcal{O}\left((N_{\text{init}} + T)^2\right)$
DAD baseline (Eq. (S.9))	$\mathcal{O}\left(N_{\text{init}} + T\right)$ in time or space	

Table S.F.3: **Training samplers.**

samples	**distributions**
$f \sim \mathcal{GP}(0, k_\theta)$	k_θ : RBF kernel, Eq. (S.11) (parameters $\theta = (v, \boldsymbol{l})$, $\boldsymbol{l} = (l_1, ..., l_D)$)
	$v \sim$ Uniform$[0.9616, 1.0]$ $\forall d \leq D, (l_d - 0.2) \sim$ Gamma$(shape = 1, rate = 10)$
$\epsilon \sim \mathcal{N}(0, \sigma^2)$	$\sigma^2 = 1.0001 - v$, i.e. $0.01 \leq \sigma \leq 0.2$, signal-to-noise ratio $\frac{\sqrt{v}}{\sigma} \geq 5$
$q \sim \mathcal{GP}_{\theta_q}(\mu_q, k_q)$ $\theta_q = (c, \boldsymbol{w}, Q, v, \boldsymbol{l})$	μ_q: sech prior mean, Eq. (S.12) (parameters $c, \boldsymbol{w}, Q$, $\boldsymbol{w} = (w_1, ..., w_D)$)
	k_q: RBF kernel, Eq. (S.11) (parameters $v, \boldsymbol{l}$, $\boldsymbol{l} = (l_1, ..., l_D)$)
	Q: QR-decomposition of A, $[A]_{i,j} \sim$ Uniform$[-1, 1], \forall i, j = 1, ..., D$ $\forall d \leq D, w_d \sim$ Uniform$[5, 40]$ $c = \sqrt{0.5}$ $v \sim$ Uniform$[0.9616 - c^2, 1.0 - c^2]$ $\forall d \leq D, (l_d - 0.2) \sim$ Gamma$(shape = 1, rate = 10)$
$\epsilon_q \sim \mathcal{N}(0, \sigma_q^2)$	$\sigma_q^2 = 1.0001 - c^2 - v$, i.e. $0.01 \leq \sigma_q \leq 0.2$, $\frac{\sqrt{c^2+v}}{\sigma_q} \geq 5$
$\mathcal{I}$ (Eqs. (2) and (5))	$\boldsymbol{X}_{\text{grid}} \sim$ Beta$(0.5, 0.5)$, $Y_{\text{grid}} = f(\boldsymbol{X}_{\text{grid}}) +$ noise, noise $\sim \mathcal{N}(0, \sigma^2)$
Fluid System	kernels: $\boldsymbol{l}$ of k_θ, k_q sampled as Eq. (S.13) safety mean shape: $\boldsymbol{w}$ sampled as Eq. (S.13)

The training complexities are dominated by the NN forward passes and the loss computations.

The NN forward passes takes $\mathcal{O}\left(\sum_{t=1}^{T}(N_{\text{init}} + t - 1)^2\right)$ in time, as self attention has square complexity (Figure S.B.1, Vaswani et al. (2017)). The space complexity depends on the number of NN parameters.

Table S.E.2 summarizes the complexities of computing our loss functions. Note that the conditional probability are expressed by the posterior Gaussian mean and covariance (Eq. (S.2)). Then we use the posterior to compute the log likelihood. We use different colors for the GP posteriors to indicate the sources of the complexities. Our appendix objectives $(\mathcal{H}_{\text{mean}}, \mathcal{I}_{\text{mean}})$ have the same complexities as the main objectives $(\mathcal{H}, \mathcal{I})$.

The safe AL objective $\mathcal{S}_{\mathcal{H}}$ is a combination of $\mathcal{H}$ and safety score. Note that $\log p(z(\boldsymbol{x}_{\phi,t+1}) < 0|z_{\phi,1:t}, Z_{\text{init}})$ are all intermediate results of $\log p(z(\boldsymbol{x}_{\phi,T}) < 0|z_{\phi,1:T-1}, Z_{\text{init}})$, which creates negligible additional complexities. The appendix safety score has the same complexity as the main, i.e. $\mathcal{S}_{\mathcal{H},\text{division}}$ and $\mathcal{S}_{\mathcal{H}}$ have the same complexities.

F Experiment Details

F.1 Offline Training & Training Time

In our current implementation, the data dimension and input bound need to be pre-defined. We fix $\mathcal{X} = [0, 1]^D$, and we map all test problems to this domain. For each number of dimensions, D, we train one policy to deploy on various AL problems. We summarize all of our sampling distributions in Table S.F.3. For GP function sampling, see Appendix D for mathematical details. Our numerical setting utilizes the assumption that deployment problems are normalized to zero mean and unit variance. An implicit assumption we make here is that the functions do have information to be extracted, not just noises (so we set $\sigma \leq 0.2, \sigma_q \leq 0.2$). The numerical setting can be adapted according to different applications. Our Fluid System is a $D = 7$ task, beyond typical deployment dimension of conventional methods ($D \geq 3$ or 4 is challenging for safe AL/BO methods, Kirschner et al. 2019; Bottero et al. 2022). We tailor the kernel lengthscale and mean shape parameters to a broad range

Table S.F.4: **Policy training time.** All training jobs were on a single Intel i5-12600K CPU and an NVIDIA RTX 3080 GPU. A combined comparison of training and runtime against baselines is provided in our ablation study (Appendix G and Table S.G.7).

loss functions training steps	$\mathcal{I}$ $10k$	DAD baseline $20k$	ALINE baseline $40k$	$\mathcal{S}_{\mathcal{H}}$ $10k$
NN 1D, 2D $N_{\text{init}} = 1, T_{\text{sim}} \leq T = 30$	$\sim$ 1 hours ($N_{\text{grid}} = 100$)			
NN 5D $N_{\text{init}} = 20, T_{\text{sim}} \leq T = 40$	$\sim$ 2 hours ($N_{\text{grid}} = 500$)			
NN 1D $N_{\text{init}} = 1, T = [10, 20]$		$\sim$ [2, 2.5] hours	$\sim$ [6, 64] hours	
NN 2D $N_{\text{init}} = 1, T = [10, 20, 30]$		$\sim$ [2, 2.5, 3.5] hours	$\sim$ [6, 77, 93.5] hours	
NN 5D $N_{\text{init}} = 20, T = [10, 20, 30, 40]$		$\sim$ [3, 4, 6, 9] hours Appendix F.1 explains why slow	$\sim$ [14, 72, 200, 350] hours	
NN 2D $N_{\text{init}} = 5, T_{\text{sim}} \leq T = 40$				$\sim$ 2.5 hours
NN 3D $N_{\text{init}} = 5, T_{\text{sim}} \leq T = 60$				$\sim$ 4 hours
NN 7D $N_{\text{init}} = 10, T_{\text{sim}} \leq T = 100$				$\sim$ 12 hours

of plausible operating values:

$$\begin{aligned} l_1 &\sim \text{Uniform}\,[0.24, 0.72], w_1 \sim \text{Uniform}\,[10.42, 41.66]\,, \\ l_2 &\sim \text{Uniform}\,[1.59, 4.77], w_2 \sim \text{Uniform}\,[1.57, 6.29]\,, \\ l_3 &\sim \text{Uniform}\,[0.415, 1.245], w_3 \sim \text{Uniform}\,[6.02, 24.1]\,, \\ l_4 &\sim \text{Uniform}\,[0.3, 0.9], w_4 \sim \text{Uniform}\,[8.33, 33.34]\,, \\ l_5 &\sim \text{Uniform}\,[0.15, 0.45], w_5 \sim \text{Uniform}\,[16.66, 66.67]\,, \\ l_6 &\sim \text{Uniform}\,[1.7, 5.1], w_6 \sim \text{Uniform}\,[1.47, 5.89]\,, \\ l_7 &\sim \text{Uniform}\,[0.22, 0.66], w_7 \sim \text{Uniform}\,[11.36, 45.45]\,. \end{aligned} \tag{S.13}$$

The gradient of all our loss functions can be computed by PyTorch autograd. This is because the GP i.i.d. noise assumption enables all our expectations to separate f, q and noises, which means a derivative w.r.t. $y_{\phi,1:T}, z_{\phi,1:T}$ propagates automatically through f, q to $\boldsymbol{x}_{\phi,1:T}$ which are direct outputs of the NN policies.

The batch sizes we set are listed in Table S.E.1. For each training pipeline, we train with a few different seeds. The optimizer is RAdam (Liu et al., 2020a). We set a lr scheduler to discount the lr by 2% every 50 training steps. We usually train with $200 * 50 = 10000$ steps, but we give the DAD baseline more steps to converge (still not competitive on GP functions when $D \geq 2$). The ALINE baseline utilizes even more steps as it learns posterior estimates and AL decisions together (Huang et al., 2025). The exact traininig steps and times are summarized in Table S.F.4. All training jobs were on a single Intel i5-12600K CPU and an NVIDIA RTX 3080 GPU.

NN Architectural Hyperparameters The NN configurations are taken largely from Ivanova et al. (2021): The transformer block is taken as it is; the number of hidden neurons in the MLPs (Data Embed, Budget Feed Forward, and Decision Feed Forward) are all set to 512. The remaining parameter is the embedding dimension (the dimension of e_t, E_t, ES_t, EB, and the input dimension of the Decision Feed Forward), which determines the size of the transformer layers and plays a crucial role in the NN size. In our main paper, we set this dimension to 128, but we provide an ablation study on this dimension in Appendix G.

In our unconstrained AL experiments, our policy NNs were trained without safety measurements: an NN does not have a safety history encoder (Figure S.B.1), and the training algorithm has neither safety measurements q and z nor safety center $\mathcal{C}$, i.e. the training Algorithm 1 is reduced to Algorithm S.1.

Training of Baseline Approaches For the DAD baseline (described in Appendix C.2, Algorithm S.3), we take the code from Foster et al. (2021); Ivanova et al. (2021). The DAD implementation utilizes Pyro (Bingham

et al., 2018) to condition the same sequence of queries on different functions, which however involve CPUs for each loss computation. In comparison, our loss functions are computed completely with PyTorch which requires little CPU-GPU interaction. This is why the training time per step of DAD is not necessarily faster than our objectives (Table S.F.4), despite DAD time complexity much smaller (Table S.E.2).

For ALINE (Huang et al., 2025), we use the original code to train the method. As ALINE optimizes the AL decisions for a predefined T, we train individual ALINE for different T. Similar to our NN policies (Figure S.B.1), we use 2 layers of transformer attender.

For PFN baselines, we train PFN models (Müller et al., 2022) on GP data (Table S.F.3). The training was adapted from the implementation of Chang et al. (2025). PFNs are amortized Bayesian models which can condition on observed data and output predictive distributions of unseen data. We use 2 layers of transformer attender to ensure the models have similar sizes (around 300K parameters) to our NN policies (Figure S.B.1). Each PFN is limited to the specified dimension D; we train individual PFNs for each dimension of problems. Each training uses 1.28M GP functions and around 3 hours (320k steps).

F.2 Online Policy Deployment

We deploy our Amortized AL (AAL) policy with Algorithm S.4 and Amortized safe AL (ASAL) policy with Algorithm S.5.

Algorithm S.4 Unconstrained AL with NN Policy

Require: $\mathcal{D}_0 \subseteq \mathcal{X} \times \mathcal{Y}$, AL policy ϕ, T
1: **for** $t = 1, ..., T$ **do**
2: $\quad \boldsymbol{x}_t = \phi(T - t + 1, \mathcal{D}_{t-1})$ **if** ϕ is budget aware **else** $\boldsymbol{x}_t = \phi(\mathcal{D}_{t-1})$
3: $\quad$ Query at $\boldsymbol{x}_t$ to get y_t
4: $\quad \mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{\boldsymbol{x}_t, y_t\}$
5: **end for**
6: **return** $\mathcal{D}_T$ suitable to model f

Algorithm S.5 Safe AL with NN Policy

Require: $\mathcal{D}_0 \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, Safe AL policy ϕ, T
1: **for** $t = 1, ..., T$ **do**
2: $\quad \boldsymbol{x}_t = \phi(T - t + 1, \mathcal{D}_{t-1})$
3: $\quad$ Query at $\boldsymbol{x}_t$ to get y_t, z_t
4: $\quad \mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{\boldsymbol{x}_t, y_t, z_t\}$
5: **end for**
6: **return** $\mathcal{D}_T$ suitable to model f

F.3 Baseline Methods

The DAD baseline has to be trained up-front (Appendices C.2 and F.1). In this case, we further take the budget encoder out of the NN structure (Figure S.B.1), as this is useless for DAD (and $T_{\text{sim}} = T$ is fixed). The DAD policies are deployed with Algorithm S.4.

The ALINE baseline is as it is in Huang et al. (2025), except that (i) the input domain is adapted to our $\mathcal{X}$ and (ii) the GP hyperparameters for training are as Table S.F.3.

Algorithm S.6 is the conventional unconstrained AL (Settles, 2010; Kumar and Gupta, 2020; Tharwat and Schenck, 2023).

Algorithm S.6 Conventional AL

Require: $\mathcal{D}_0 \subseteq \mathcal{X} \times \mathcal{Y}$, acquisition function a, T
for $t = 1, ..., T$ **do**
  Model $\mathcal{M}_{t-1}$ with $\mathcal{D}_{t-1}$
  $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x} \in \mathcal{X}} a(x | \mathcal{M}_{t-1}, \mathcal{D}_{t-1})$
  Query at $\boldsymbol{x}_t$ to get y_t
  $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{\boldsymbol{x}_t, y_t\}$
end for
return $\mathcal{D}_T$

Then we write down the safe AL (Schreiter et al., 2015; Zimmer et al., 2018; Li et al., 2022).

Algorithm S.7 Conventional Safe AL

Require: $\mathcal{D}_0 \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, acquisition function a, safety threshold 0, confidence tolerance γ, T
for $t = 1, ..., T$ **do**
  Model $\mathcal{M}_{t-1}$, $\mathcal{M}_{\text{safety},t-1}$ with $\mathcal{D}_{t-1}$
  $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x} \in \mathcal{X}} a(x | \mathcal{M}_{t-1}, \mathcal{D}_{t-1})$ subject to $p(z(\boldsymbol{x}) \geq 0 | \mathcal{M}_{\text{safety},t-1}, \mathcal{D}_{t-1}) \geq 1 - \gamma$
  Query at $\boldsymbol{x}_t$ to get y_t, z_t
  $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{\boldsymbol{x}_t, y_t, z_t\}$
end for
return $\mathcal{D}_T$

We fix a safety tolerance of $\gamma = 0.05$, if not particularly described.

The base acquisition function a is the predictive entropy $\mathbb{H}(y(\boldsymbol{x})|\mathcal{D}_{t-1})$ (Eq. (S.5)). The models $\mathcal{M}_{t-1}, \mathcal{M}_{\text{safety},t-1}$ are GPs or PFNs in this paper. The models are described in detail later in Appendix F.4.

The benchmark problems consist of continuous functions and discrete datasets.

For the datasets, the (constrained) acquisition optimizations are solved by an exhaustive search, as done in safe learning literature (Sui et al., 2015; Berkenkamp et al., 2020; Li et al., 2022). In other words, we compute the acquisition scores and the safety distributions on the entire pool of unseen data, and we use the values to solve the (constrained) acquisition optimization.

For function problems, since a fine discretization with exhaustive search is computationally possible, we discretize the space densely by randomly sampling 5000 input points. Then we solve the (constrained) acquisition optimization problems as if these are pools. Please be aware that the discretization is inherited from conventional safe learning methods (Sui et al., 2015; Berkenkamp et al., 2020; Li et al., 2022)) to solve the constrained acquisition optimization problem. In the main paper, our policies propose queries on continuous space, which requires no discretization.

F.4 Online Evaluations and Posterior Estimates

In our experiments, we compute posteriors for either RMSE evaluations (using posterior means) or model-based AL and safe AL baselines.

GP Posteriors The first estimation approach is via GPs, leveraged in the following scenarios. We always use a GP of zero mean and RBF kernel.

1. Our **AAL**, our **ASAL**, **DAD**, Random: we deploy our amortized (safe) AL (Algorithms S.4 and S.5) or the DAD, Random baselines to collect data. After the specified T data points are collected, we use the initial and queried data to fit a vanilla GP model with Type II maximum likelihood (optimization: L-BFGS-B algorithm).

2. **GP AL**, **Safe GP AL**, **Safe Random**: We deploy conventional AL and safe AL (Algorithms S.6 and S.7)

with vanilla GPs. Each iteration updates GPs with Type II maximum likelihood (optimization: L-BFGS-B algorithm).

3. **AGP AL**, **Safe AGP AL**: We deploy conventional AL and safe AL (Algorithms S.6 and S.7) with AGPs. AGP is an amortized GP method developed by Liu et al. (2020b); Bitzer et al. (2023) (AGP). Such a method sampled GP data and trained a transformer model to approximate the Type II maximum likelihood. The AGP is a model with a transformer module. Whenever an observation dataset is given, the transformer module infers the kernel structure (which we fix to an RBF) together with the kernel hyperparameters. Afterward, one can apply Eq. (S.2) for computing the predictive distribution. Bitzer et al. (2023) provides a trained model attached to their code.

4. **SVGP AL**, **Safe SVGP AL**: We deploy conventional AL and safe AL (Algorithms S.6 and S.7) with sparse variational GPs (SVGPs, Hensman et al. 2013, 2015b). A SVGP uses a set of inducing variables (ivs, pseudo data points) to approximate the distribution conditioned on the full dataset. The number of ivs are smaller than the number of observed data, resulting in a cheaper decomposition of the gram matrix. The ivs are selected via k-means and the GP models are optimized via the L-BFGS-B algorithm, as done in the literature. We do not perform mini-batching (thus equivalent to Titsias 2009). For Engine and Fluid System problems, we use 20 ivs, and 15 for all other problems. The ivs are fixed to the observed data if given less data than the desired number of ivs (i.e., a SVGP reduces to full GP with the variational inference objective).

5. **MGP AL, Safe MGP AL**: We deploy conventional AL and safe AL (Algorithms S.6 and S.7) with mixture of GPs (MGPs, Riis et al. 2023), an ensembling GP method fine-tuning multiple vanilla GPs for a fixed number of steps, each with an individual set of initial model hyperparameters. We use 30 GPs and fine-tune each with Adam optimizer (learning rate 0.01) for 30 steps (Riis et al., 2023).

We want to point out that the cubic complexity is inevitable as long as we compute a GP distribution. The difference of GP and AGP is that training a GP computes the gram matrix multiple times while AGP computes the gram matrix only once. Our Amortized (safe) AL methods take GP computation completely out of the deployment cycle.

Other Posterior Estimates In addition, the following baselines have posterior estimates available as the queries are acquired.

1. **ALINE** (Huang et al., 2025): the method uses NN forward passes for AL decisions and posterior modeling jointly. The posterior estimates predictive density conditioned on previous observations.

2. **PFN/TabPFN AL, Safe PFN/TabPFN AL**: PFNs and TabPFN are amortized Bayesian models which can condition on observed data and output predictive distributions of unseen data. PFNs are trained on our GP data while TabPFN is a foundation model published by Hollmann et al. (2025) (we use the TabPFNv2 regressor). The output posterior estimates can be used to compute the entropy (Viering et al., 2025) and safety likelihood, which are then used for (constrained) acquisition optimizations. In other words, we deploy Algorithms S.6 and S.7 by using a PFN or TabPFN model.

F.5 Deployment Hardware & Complexities

All of our (safe) AL deployments are run on the same personal computer without a GPU.

The (safe) AL deployment complexities are summarized in Table S.F.5 and detailed below.

- Our **AAL**, our **ASAL**, **DAD** (Algorithms S.4 and S.5): NN forward passes take $\mathcal{O}\left((N_{\text{init}}+t-1)^2\right)$ at each $t=1,...,T$, dominated by the transformer attender layers (Figure S.B.1, Vaswani et al. (2017)).

- **ALINE** (Huang et al., 2025): this method uses NN forward passes for AL decisions and posterior modeling jointly. The complexity is $\mathcal{O}\left(N_{\text{pool}}(N_{\text{init}}+t-1)^2\right)$ as the method uses a transformer to process the pool as well as the observed data.

Table S.F.5: **Deployment complexities.** We deploy (safe) AL for $t = 1, ..., T$, and provide the complexities for each t. Details are given in Appendix F.5.

method	time complexity
our AAL (Algorithm S.4) our ASAL (Algorithm S.5) DAD (Algorithm S.4)	$\mathcal{O}\left((N_{\text{init}} + t - 1)^2\right)$
ALINE (Huang et al., 2025) PFN AL, TabPFN AL (Algorithm S.6) Safe PFN AL, Safe TabPFN AL (Algorithm S.7)	$\mathcal{O}\left(N_{\text{pool}}(N_{\text{init}} + t - 1)^2\right)$
GP AL, AGP AL (Algorithm S.6) Safe GP AL, Safe AGP AL (Algorithm S.7) Safe Random (Algorithm S.7)	$\mathcal{O}\left((N_{\text{init}} + t - 1)^3\right) + \mathcal{O}\left(N_{\text{pool}}(N_{\text{init}} + t - 1)^2\right)$
SVGP AL (Algorithm S.6) Safe SVGP AL (Algorithm S.7)	$\mathcal{O}\left((N_{\text{init}} + t - 1)N_{\text{iv}}^2\right) + \mathcal{O}\left(N_{\text{pool}}N_{\text{iv}}^2\right)$
MGP AL	$\mathcal{O}\left(N_{\text{ensemble}}(N_{\text{init}} + t - 1)^3\right) + \mathcal{O}\left(N_{\text{pool}}N_{\text{ensemble}}(N_{\text{init}} + t - 1)^2\right)$

- **PFN/TabPFN AL, Safe PFN/TabPFN AL** (Algorithms S.6 and S.7): PFN and TabPFN use transformers to process the observed data and the target pool. The observed data are self attended while the target pool cross attends to the observed data. The complexity is $\mathcal{O}\left((N_{\text{init}} + t - 1)^2\right)$ to produce the posterior estimates of each data point. A linear burden is necessary for the (constrained) acquisition optimization, resulting in the final complexity $\mathcal{O}\left(N_{\text{pool}}(N_{\text{init}} + t - 1)^2\right)$.
- **(A)GP AL**, **Safe (A)GP AL** (Algorithms S.6 and S.7):
  1. Model fitting: at each $t = 1, ..., T$, GP modeling takes $\mathcal{O}\left((N_{\text{init}} + t - 1)^3\right)$ in time, exact factor depends on GP training methods; note that AGP methods have much smaller factor because the GP hyperparameters are inferred and the gram matrix is computed only once.
  2. Acquisition optimization: at each $t = 1, ..., T$, (constrained) acquisition optimization via exhaustive search takes $\mathcal{O}\left(N_{\text{pool}}(N_{\text{init}} + t - 1)^2\right)$ due to GP inferences, where N_{pool} is the size of the search pool.
- **Safe Random**: A vanilla GP is used to model the safety confidence, resulting in the same complexities as **Safe (A)GP AL** (up to a constant scaling).
- **SVGP AL**, **Safe SVGP AL** (Algorithms S.6 and S.7):
  1. Model fitting: at each $t = 1, ..., T$, GP modeling takes $\mathcal{O}\left((N_{\text{init}} + t - 1)N_{\text{iv}}^2\right)$ in time, N_{iv} is the number of inducing variables; the time of selecting the inducing variables via k-means is negligible; in comparison to vanilla GPs (GP AL, Safe GP AL), we observe more training iterations needed while the complexity reduction of each gram matrix decomposition is minor in our data set sizes, resulting in a worse empirical time performance.
  2. Acquisition optimization: at each $t = 1, ..., T$, (constrained) acquisition optimization via exhaustive search takes $\mathcal{O}\left(N_{\text{pool}}N_{\text{iv}}^2\right)$ due to GP inferences, where N_{pool} is the size of the search pool.
- **MGP AL** (Algorithm S.6):
  1. Model fitting: at each $t = 1, ..., T$, GP modeling takes $\mathcal{O}\left(N_{\text{ensemble}}(N_{\text{init}} + t - 1)^3\right)$ in time, N_{ensemble} is the number of GPs.
  2. Acquisition optimization: at each $t = 1, ..., T$, (constrained) acquisition optimization via exhaustive search takes $\mathcal{O}\left(N_{\text{pool}}N_{\text{ensemble}}(N_{\text{init}} + t - 1)^2\right)$ due to GP inferences, where N_{pool} is the size of the search pool.

F.6 Benchmark Problems

We run *AL* and *safe AL* experiments over the following benchmark problems. All problems are mapped to $\mathcal{X} = [0, 1]^D$ if they are not originally defined on such domain.

We select benchmark tasks in line with the GP, AL, and safe-AL literature, focusing on low- to moderate-dimensional regimes where these methods are proven most competitive (summarized in Table S.F.6). Importantly, our final Fluid System scales to 7D–exceeding the usual $D \leq 3$ or 4 in safe AL/BO–indicating that our approach remains effective on conventionally challenging problems.

Table S.F.6: **Benchmark dimension in the literature.**

	dimension D	area of impact
Riis et al. (2022)	$D \leq 6$	GP AL on benchmark problems
Pu et al. (2025)	$D \leq 3$, $C \leq 4$, C: number of categorical variables	GP modeling, BO on hyperparameter tuning
DAD (Foster et al., 2021; Ivanova et al., 2021)	$D \leq 2$	Amortized BED on scientific experiments
Zimmer et al. (2018)	time series of 2 free variables	Safe AL on engineering problems
Stage Opt (Sui et al., 2018)	$D \leq 2$	Safe BO on clinical experiments
Safe Opt (Berkenkamp et al., 2020)	$D \leq 2$	Safe BO on robotics
Bottero et al. (2022)	$D \leq 5$	Safe exploration on benchmark problems

AL, continuous - sin function: This is a one dimension problem $x \in [0, 1]$,

$$f(x) = \sin(20x).$$

In the experiments, we sample Gaussian noise $\epsilon \sim \mathcal{N}\left(0, 0.1^2\right)$.

We randomly sample 50 test points to evaluate the modeling RMSE.

AL, continuous - branin function: This function is defined over $(x_1, x_2) \in [-5, 10] \times [0, 15]$,

$$f_{a,b,c,r,s,t}\left((x_1, x_2)\right) = a(x_2 - bx_1^2 + cx_1 - r) + s(1 - t)\cos(x_1) + s,$$

where $(a, b, c, r, s, t) = (1, \frac{5.1}{4\pi^2}, \frac{5}{\pi}, 6, 10, \frac{1}{8\pi})$ are constants. We sample noise free data points and use the samples to normalize our output

$$f_{a,b,c,r,s,t}\left((x_1, x_2)\right)_{\text{normalize}} = \frac{f_{a,b,c,r,s,t}\left((x_1, x_2)\right) - mean(f_{a,b,c,r,s,t})}{std(f_{a,b,c,r,s,t})}.$$

std is a standard deviation. In the experiments, we sample Gaussian noise $\epsilon \sim \mathcal{N}\left(0, 0.1^2\right)$.

We randomly sample 200 test points to evaluate the modeling RMSE.

AL, pool - airline passenger dataset: This is a publically available time series dataset [3]. Each data point has a date input (year and month) and a number of passengers as output. We convert the input into real number as $year + (month - 1)/12$, and then rescale the entire input space to $[0, 1]$ (the earliest date becomes 0 while the latest becomes 1). The output data are again normalized to zero mean and unit variance.

This is a dataset of 144 measurements. Before the experiments, we randomly pick 50 points as test data to evaluate RMSE, N_{init} initial data, and the remaining forms the pool.

AL, pool - Langley Glide-Back Booster (LGBB) dataset: This is a two dimension dataset described in Rogers et al. (2003)[4]. The dataset has multiple outputs and we take the "lift" to run our experiments (after normalized to zero mean and unit variance). The inputs are x_1 (mach) and x_2 (alpha), which are normalized by

$$x_1 = mach/6,$$
$$x_2 = (alpha + 5)/35.$$

After doing this, the input space is $[0, 1]^2$.

This is a dataset of around 850 measurements. Before the experiments, we randomly pick 200 points as test data to evaluate RMSE, N_{init} initial data, and the remaining forms the pool.

AL, pool - Airfoil dataset: This is a five dimension NASA dataset available on UCI datasets[5]. The first five channels are taken as inputs, after mapped to $[0, 1]^5$. The last channel is an output which needs to be normalized to zero mean and unit variance. This is a dataset of around 1500 measurements. Before the experiments, we randomly pick 500 points as test data to evaluate RMSE, N_{init} initial data, and the remaining forms the pool.

[3] https://github.com/jbrownlee/Datasets/blob/master/airline-passengers.csv
[4] https://bobby.gramacy.com/surrogates/lgbb.tar.gz, lgbb_original.txt
[5] https://archive.ics.uci.edu/dataset/291/airfoil+self+noise

Safe AL, continuous - Simionescu function: This is a constrained problem (Simionescu, 2014) defined over $(x_1, x_2) \in [-1.25, 1.25]^2$. The main function is

$$f(x_1, x_2) = 0.1 x_1 x_2$$

We sample noise free data points and use the samples to normalize our output

$$f\left((x_1, x_2)\right)_{\text{normalize}} = \frac{f\left((x_1, x_2)\right) - mean(f)}{std(f)}.$$

std is a standard deviation. In the experiments, we sample Gaussian noise $\epsilon \sim \mathcal{N}\left(0, 0.1^2\right)$.

The task is subject to a constraint function:

$$\begin{aligned} q(x_1, x_2) &= \left[1 + 0.2\cos\left(8\arctan(x_1/x_2)\right)\right]^2 - x_1^2 - x_2^2, \\ q\left((x_1, x_2)\right)_{\text{normalize}} &= \frac{q\left((x_1, x_2)\right)}{std(q)}. \end{aligned}$$

std is a standard deviation. The constraint is $q \geq 0$, and we normalize only the standard deviation to ensure the constraint level remains the same. In the experiments, we sample Gaussian noise $\epsilon_q \sim \mathcal{N}\left(0, 0.1^2\right)$.

We randomly sample 200 safe test points to evaluate the modeling RMSE. The initial data points of each deployment are sampled at a central area $\mathcal{C} = [0.4, 0.6]^2 \subseteq \mathcal{X}$ (standardized $\mathcal{X} = [0, 1]^2$), under constraint $z \geq 0$.

Safe AL, continuous - Townsend function: This is a constrained problem (Townsend, 2017) [6] defined over $(x_1, x_2) \in [-2.25, 2.25] \times [-2.5, 1.75]$. The main function is

$$f(x_1, x_2) = -\left[\cos\left((x_1 - 0.1)x_2\right)\right]^2 - x_1 \sin(3x_1 + x_2).$$

We sample noise free data points and use the samples to normalize our output

$$f\left((x_1, x_2)\right)_{\text{normalize}} = \frac{f\left((x_1, x_2)\right) - mean(f)}{std(f)}.$$

std is a standard deviation. In the experiments, we sample Gaussian noise $\epsilon \sim \mathcal{N}\left(0, 0.1^2\right)$.

The task is subject to a constraint function:

$$\begin{aligned} q(x_1, x_2) &= \left(2\cos(b) - \frac{1}{2}\cos(2b) - \frac{1}{4}\cos(3b) - \frac{1}{8}\cos(4b)\right)^2 + \left(2\sin(b)\right)^2 - x_1^2 - x_2^2, \\ b &= \text{arctan2}(x1, x2) \\ &= \begin{cases} \arctan(x_1/x_2) & \text{, if } x_2 > 0 \\ \arctan(x_1/x_2) + \text{sign}(x_1)\pi & \text{, if } x_2 < 0, \text{ say } \text{sign}(0) = 1 \\ \text{sign}(x_1)\pi/2 & \text{, if } x_1 \neq 0, x_2 = 0 \\ 0 & \text{, if } x_1 = 0, x_2 = 0 \end{cases}, \\ q\left((x_1, x_2)\right)_{\text{normalize}} &= \frac{q\left((x_1, x_2)\right)}{std(q)}. \end{aligned}$$

std is a standard deviation. The constraint is $q \geq 0$, and we normalize only the standard deviation to ensure the constraint level remains the same. In the experiments, we sample Gaussian noise $\epsilon_q \sim \mathcal{N}\left(0, 0.1^2\right)$.

We randomly sample 200 safe test points to evaluate the modeling RMSE. The initial data points of each deployment are sampled at a central area $\mathcal{C} = [0.4, 0.6]^2 \subseteq \mathcal{X}$ (standardized $\mathcal{X} = [0, 1]^2$), under constraint $z \geq 0$.

[6] https://www.chebfun.org/examples/opt/ConstrainedOptimization.html

Safe AL, pool - Langley Glide-Back Booster (LGBB) dataset: As described above for unconstrained AL, this dataset has multiple outputs and "lift" is taken as y. We additionally take "pitch" as z (pitching moment coefficient, see Pamadi et al. (2004)). We take a threshold $z \geq 0$, corresponding to around 60% of the space.

The pitching moment coefficient is a quantity in aerodynamics, which is not necessarily safety critical but is important for the stability. Collecting data under the constraint $z \geq 0$ means we model more carefully for larger and positive pitching moment coefficient.

This is a dataset of around 850 measurements. Before the experiments, we randomly pick 200 safe points as test data to evaluate RMSE, N_{init} initial data, and the remaining forms the pool. The initial data points of each deployment are sampled at a central area $\mathcal{C} = [0.4, 0.6]^2 \subseteq \mathcal{X}$, under constraint $z \geq 0$.

Safe AL, pool - Engine dataset: This is a dataset published by Bosch[7]. We use the second file from the link (engine2). We take engine_speed, engine_load, air_fuel_ratio as inputs, engine_roughness_s as output y, and temperature_exhaust_manifold as constraint $-z$. The negative sign $(-z)$ is added because we want a small temperature, and thus a large negative temperature z. The inputs need to be mapped to $[0, 1]^3$, y and z are already normalized by Bosch. We shift the constraint by 0.2 so that $z - 0.2 \geq 0$ is around half of the space.

This is a dataset of around 800 measurements. Before the experiments, we randomly pick 200 safe points as test data to evaluate RMSE, N_{init} initial data, and the remaining forms the pool. The initial data points of each deployment are sampled at a central area $\mathcal{C} = [0.4, 0.6]^3 \subseteq \mathcal{X}$, under constraint $z - 0.2 \geq 0$.

Safe AL, semi-continuous - high-pressure Fluid System: The high-pressure Fluid System is a nonlinear dynamical system, where we aim to model the rail pressure (our y) (Zimmer et al., 2018). The input has two free variables, actuation signal $v_k \in [0, 60]$ and speed of an external engine $n_k \in [1000, 4000]$, $k \in \mathbb{Z}^+$ is a time stamp. Originally, this is a time series problem: y_k is a function of $\boldsymbol{x}_k = (n_k, n_{k-1}, n_{k-2}, n_{k-3}, v_k, v_{k-1}, v_{k-3})$.

We sample 10^6 random trajectories with a bounded step size (0.3 of the domain) and treat the trajectories as a 7D dataset. This renders the Fluid System into a pool-based 7D safe AL problem. Note that our training does not incorporate time series nature of this task (Appendix F.1). Sampling a pool with controlled maximum reachable step ensures the trajectories are realistic (e.g. as system's max acceleration is constrained); 10^6 is dense enough yet not too expensive for baseline methods.

Concretely, each $\boldsymbol{x}_k$ is sampled by

$$
\begin{aligned}
n_{k-3} &\sim \text{Uniform}[1000, 4000], \\
n_i &\sim \text{Uniform}[\max(1000, n_{i-1} - \Delta n_{\max}), \min(n_{i-1} + \Delta n_{\max}, 4000)], \text{ for } i = k-2, k-1, k \\
\Delta n_{\max} &= 900 = 0.3 * (4000 - 1000) \\
v_{k-3} &\sim \text{Uniform}[0, 60], \\
v_{k-1} &\sim \text{Uniform}[\max(0, v_{k-3} - 2 * \Delta v_{\max}), \min(v_{k-3} + 2 * \Delta v_{\max}, 60)], \\
v_k &\sim \text{Uniform}[\max(0, v_{k-1} - \Delta v_{\max}), \min(v_{k-1} + \Delta v_{\max}, 60)], \\
\Delta v_{\max} &= 18 = 0.3 * 60.
\end{aligned}
$$

To obtain the data y, z, we take the implementation from Zimmer et al. (2018) with minor adaptation:

1. The noise-free observation is normalized $f_{\text{normalize}} = (f - 18)/15$.
2. The final observation is $y = f_{\text{normalize}} + \epsilon, \epsilon \sim \mathcal{N}\left(0, 0.1^2\right)$.
3. The safety constraint $f \leq 18$ ($f_{\text{normalize}} \leq 0$) is treated as $q_{\text{normalize}} \geq 0$ where $q_{\text{normalize}} = -f_{\text{normalize}}$.

The input domain $[1000, 4000]^4 \times [0, 60]^3$ is mapped to $\mathcal{X} = [0, 1]^7$. We randomly sample 10000 safe test points to evaluate the modeling RMSE. The initial data points are sampled at a central area $\mathcal{C} = [0.4, 0.6]^7 \subseteq \mathcal{X}$.

This 7D Fluid System exceeds the usual $D \leq 3$ or 4 in safe AL/BO. Conventional methods struggle as solving constrained acquisition optimization on such dimension is known to be difficult due to the naturally large pool

[7] https://github.com/boschresearch/Bosch-Engine-Datasets/tree/master/pengines

Table S.G.7: **Offline vs Online Time Thresholds**. The number of deployments required for the cumulative time savings of our trained policies to offset their initial pretraining time (Table S.F.4). Thresholds are calculated relative to GP and MGP reference baselines (Figures 2 and 3).

	(D, N_{init}, T)	Our AAL	DAD	ALINE	Our ASAL
GP_AL	$(1, 1, 20)$	$\sim$ 17x-18x	$\sim$ 21x-22x	$\sim$ 275x-342x	
	$(2, 1, 30)$	$\sim$ 10x-12x	$\sim$ 18x-21x	$\sim$ 281x-302x	
	$(5, 20, 40)$	$\sim$ 17x	$\sim$ 38x	$\sim$ 773x	
Safe_GP_AL	$(2, 5, 30)$				$\sim$ 12x-15x
	$(3, 5, 60)$				$\sim$ 10x
	$(7, 10, 100)$				$\sim$ 9x
MGP_AL	$(1, 1, 20)$	$\sim$ 1x	$\sim$ 1x	$\sim$ 10x-13x	
	$(2, 1, 30)$	$\sim$ 1x	$\sim$ 1x	$\sim$ 10x-11x	
	$(5, 20, 40)$	$\sim$ 1x	$\sim$ 1x	$\sim$ 28x	
Safe_MGP_AL	$(2, 5, 30)$				$\sim$ 1x
	$(3, 5, 60)$				$\sim$ 1x
	$(7, 10, 100)$				$\sim$ 1x

and the subsequent high complexity (Bottero et al. 2022; Kirschner et al. 2019; see Appendix F.5, Table S.F.5 for the complexity).

G Ablation Studies

G.1 Unconstrained AL Objectives, Train and Run Time Thresholds

We analyze the number of deployments required for our reusable trained policies to offset their initial pretraining time. This metric can help inform decisions on whether training an amortized policy is computationally beneficial overall.

Before presenting these thresholds, we emphasize that our work caters primarily to scenarios with strictly constrained deployment times. In such settings, deployment time is the primary bottleneck; if an experiment takes too long at runtime, it cannot be executed regardless of the pretraining cost.

However, if no such constraint is present, we can use the timing results (Table S.F.4 and Figures 2 and 3) to calculate this threshold as follows:

$$\text{NumDeployTrials} = \frac{\text{PolicyTrainingTime}}{5 \times (\text{BaselineDeployTime} - \text{PolicyDeployTime})},$$

the denominator includes a coefficient of 5 to reflect that each task is deployed for 5 repetitions for statistical robustness; additionally, for a fair comparison, the policy training times of the baselines are normalized to match our 10k training steps: DAD is divided by 2 (from 20k steps) and ALINE is divided by 4 (from 40k steps).

Because our policy deployment time is significantly shorter than that of conventional model-based baselines, running a baseline beyond this threshold results in a higher total time than offline training and subsequently deploying our policy. Note that our AAL and ASAL methods generalize to various T, making the initial training effort more reusable than methods such as ALINE and DAD.

We compute these thresholds for GP AL, Safe GP AL, MGP AL, and Safe MGP AL under the same configurations as (Figures 2 and 3) (e.g., deployed T settings). GP is a standard baseline in the literature, while MGP serves as a strongly performing alternative, making both reasonable benchmarks for this comparison.

The results are detailed in Table S.G.7. In summary, our methods (AAL and ASAL) offset their pretraining costs relatively quickly, requiring only a modest number of deployments. Notably, when compared to the computationally-intensive MGP baseline, our methods achieve amortization almost immediately. In contrast, ALINE requires hundreds of deployments (compared to baselines based on vanilla GPs) to justify its extensive training time.

G.2 Unconstrained AL Objectives, Main vs Appendix

This experiment provide two pieces of information: (i) how the training is monitored, and (ii) comparison of the main objective and appendix objective.

Table S.G.8: **Policies of varied sizes, training metrics**. Shown is the average training loss (negative $\mathcal{I}$) and GP test RMSE (last 500 training steps). Both metrics are meant to be minimized, giving first insight which trained policy to select.

	loss / GP test RMSE		
	AAL ed 128	AAL ed 64	AAL ed 32
1D	-0.6844 / 0.137	-0.6841 / 0.141	-0.6835 / 0.141
2D	-1.3582 / 0.177	-1.3292 / 0.201	-1.2761 / 0.245
5D	-0.4216 / 0.834	-0.4044 / 0.845	-0.3870 / 0.850

Table S.G.9: **Policies of varied sizes, model performance after deploying**. Shown is the RMSE on the test set. Smaller NNs (smaller ed) seem to result in noticeable performance deterioration.

	AAL ed 128	AAL ed 64	AAL ed 32	GP AL	Random
Sinus $N_{\text{init}} + T = 1 + 20$	0.14 ± 0.004	0.13 ± 0.005	0.12 ± 0.005	0.13 ± 0.009	0.33 ± 0.055
Airline $N_{\text{init}} + T = 1 + 20$	0.41 ± 0.022	0.46 ± 0.027	0.42 ± 0.023	0.43 ± 0.038	0.41 ± 0.023
Branin $N_{\text{init}} + T = 1 + 30$	0.21 ± 0.020	0.19 ± 0.013	0.35 ± 0.032	0.19 ± 0.011	0.28 ± 0.052
LGBB $N_{\text{init}} + T = 1 + 30$	0.17 ± 0.013	0.22 ± 0.016	0.20 ± 0.014	0.17 ± 0.010	0.21 ± 0.050
Airfoil $N_{\text{init}} + T = 20 + 40$	0.62 ± 0.012	0.70 ± 0.017	0.70 ± 0.031	0.62 ± 0.021	0.79 ± 0.047

We compare the two objectives $\mathcal{I}, \mathcal{I}_{\text{mean}}$ (Eqs. (5) and (S.8)) under the same numerical setup as described in Tables S.E.1 and S.F.3. These objectives have the same complexity (Table S.E.2).

At the end of each training epoch, we sample GP functions, deploy the policy for T steps, where this T is the training max sequence length, and then we use the ground truth GP and the T queries to evaluate the modeling RMSE. We call this the GP test RMSE. Please be aware of the difference that we deploy policies on benchmarks problems *after* training, and such deployment RMSE requires GP fitting as ground truth is not available.

In Figure S.G.5, we demonstrate the training losses (negative training objectives) as well as the deployment results. The behavior of the two objectives do not seem to have obvious differences. Note that this paper use RBF kernels, which means the functions are assumed stationary. To avoid confusion, we perform only Type II maximum likelihood (no AGP) for the deployment GP modeling.

The training loss and GP test RMSE appear to be good indicators of the training performance, as the $1D$ and $2D$ examples show good deployments for policies of good training loss or GP test RMSE (Figure S.G.5). The $2D$ policies sometimes get stuck in patterns where queries are put only at the border of $\mathcal{X}$, namely those with worse training losses and GP test RMSEs. This pattern happens way more often if we train with the entropy losses $\mathcal{H}, \mathcal{H}_{\text{mean}}$ (results not shown).

The policies shown in our main paper are the one here with the best average training loss in the last 10 epochs (last 500 training steps).

G.3 Unconstrained AL of Varied NN Sizes

As described in Appendix F.1, the embedding dimension plays a crucial role in the NN size. Here, we would like to study whether we can make the NN even smaller.

We set the embedding dimension, denoted by ed, to 32 or 64, in comparison to our main 128, and we train on the unconstrained AL problems. Similar to the main Table 1, we report the test RMSE after AL deployment. The results are shown in Table S.G.9. It seems that ed 32 can be too small on 2D and 5D problems, ed 64 performs worse than ed 128 on 1D Airline, 2D LGBB, 5D Airfoil but comparable on 1D Sinus and 2D Branin problems. This ablation study shows that our current setting, ed 128, seems to give an effective yet small NNs.

Table S.G.10: **RMSE on standard AL tasks of smaller T** ($T = 10$ for Sinus, Airline and $T = 20$ for Branin, LGBB). The results are attached to the main Table 1, which demonstrates results of $T = 20, 20, 30, 30$ for Sinus, Airline, Branin, LGBB, respectively. On Sin function, $T = 10$ is usually too few for convergence.

Method	Sinus $(1 + 10)$	Airline $(1 + 10)$	Branin $(1 + 20)$	LGBB $(1 + 20)$
Our AAL	0.64 ± 0.056	0.50 ± 0.032	0.33 ± 0.008	0.23 ± 0.005
ALINE	0.56 ± 0.022	0.47 ± 0.022	0.37 ± 0.024	0.18 ± 0.007
DAD	0.84 ± 0.119	0.45 ± 0.014	0.81 ± 0.040	0.32 ± 0.012
PFN_AL	1.16 ± 0.037	0.52 ± 0.053	0.43 ± 0.036	0.20 ± 0.003
TabPFN_AL	1.19 ± 0.026	0.44 ± 0.025	0.34 ± 0.028	0.21 ± 0.012
AGP_AL	0.28 ± 0.016	0.53 ± 0.028	0.35 ± 0.031	0.23 ± 0.010
GP_AL	0.54 ± 0.046	0.46 ± 0.033	0.36 ± 0.014	0.18 ± 0.014
SVGP_AL	0.52 ± 0.013	0.50 ± 0.060	0.38 ± 0.024	0.20 ± 0.013
MGP_AL	0.22 ± 0.040	0.47 ± 0.024	0.30 ± 0.023	0.19 ± 0.015
Random	0.51 ± 0.070	0.53 ± 0.088	0.45 ± 0.068	0.20 ± 0.009

G.4 Unconstrained AL of Smaller T

We demonstrate in Table S.G.10 the learning performance on unconstrained AL tasks of smaller T, as attached to Table 1. The results show similar conclusion that our amortized approach (AAL) performs comparable learning outcomes. ALINE and DAD require retraining for different T and are not shown.

G.5 Main Safe AL Objective, $\gamma = 0.05$ vs $\gamma = 0$

We train on $2D$ with $\mathcal{S}_{\mathcal{H}}, \gamma = 0.05$ or $\gamma = 0$ (Eq. (7)). Note that we add 10^{-5} to the unsafe likelihood for numerical stability, i.e. the safety term of $\mathcal{S}_{\mathcal{H}}$ has a bound $\log \max(\gamma, p(z(\boldsymbol{x}_{\phi,t+1}) < 0 | z_{\phi,1:t}, Z_{\text{init}})) \geq \log\left(10^{-5}\right)$. To avoid confusion, we perform only Type II maximum likelihood for GP modeling. We see from Figure S.G.7 that $\gamma = 0$ makes the safety scores dominate the loss values. As a result, the deployment is safer but the collected data lead to worse modeling performance.

Please be aware that our policy still does not need GP to deploy, which means we are faster than all the GP-based baselines, including Safe Random. Overall, we should specify γ depending on the safety criticality.

We train with a few different random seed values. In our main paper, we again select the policy based on the average loss values of the last 10 epochs (last 500 training steps).

G.6 ASAL Objectives Main vs Appendix, Safe AL of Minimum Unsafe Criterion Eq. (6)

We compare a couple of methods:

1. safe AL of policy trained on our main $\mathcal{S}_{\mathcal{H}}$, $\gamma = 0.05$ (Eq. (7)),
2. safe AL of policy trained on our appendix $\mathcal{S}_{\mathcal{H}_{\text{mean}}}$, $\gamma = 0.05$ (unconstrained $\mathcal{H}_{\text{mean}}$ decorated with our main min unsafe likelihood, see Figure S.C.3),
3. safe AL of policy trained on our appendix $\mathcal{S}_{\mathcal{H},\text{division}}$ (unconstrained $\mathcal{H}$ decorated with our appendix max safe likelihood, see Eq. (S.10) and Figure S.C.3),
4. safe AL of policy trained on our appendix $\mathcal{S}_{\mathcal{H}_{\text{mean}},\text{division}}$ (unconstrained $\mathcal{H}_{\text{mean}}$ decorated with our appendix max safe likelihood, see Figure S.C.3), and
5. conventional GP based safe AL (Algorithm S.7) but we add the unconstrained safety-aware acquisition criterion (Eq. (6)), named MinUnsafe GP AL: $\boldsymbol{x}_t = \text{argmax}\{\mathbb{H}[y(\boldsymbol{x})|y_{1:t-1}, Y_{\text{init}}] - \log \max(\gamma, p(z(\boldsymbol{x}) < 0 | z_{1:t-1}, Z_{\text{init}}))\}$ ($\gamma = 0.05$, this is the same as Eq. (6) if we take expectation over the forecasted $y(\boldsymbol{x})$, and this corresponds to objectives $\mathcal{S}_{\mathcal{H}}, \mathcal{S}_{\mathcal{H}_{\text{mean}}}$, see the paragraph of Eqs. (6) and (7)).

All objectives are leveraged under the same numerical setup as described in Tables S.E.1 and S.F.3. These objectives have the same complexity (Table S.E.2).

To avoid confusion, we perform only Type II maximum likelihood for GP modeling. We demonstrate the result on all benchmark problems and datasets (Figure S.G.8). Firstly, the performance of Safe GP AL (constrained

predictive entropy acquisition) does not seem to have obvious empirical differences from MinUnsafe GP AL. This shows an advantage of MinUnsafe GP AL because its acquisition criterion is safety-aware but unconstrained, and an unconstrained optimization is much easier to be solved. Note that, for a fair comparison, our paper solves the acquisition optimization of MinUnsafe GP AL on discretized set.

Our ASAL policies trained with different objectives all seem to query reasonable data. In average, minimizing unsafe likelihood ($\mathcal{S}_{\mathcal{H}}, \mathcal{S}_{\mathcal{H}_{\text{mean}}}$) prioritizes safety over AL exploration. This is consistent to what we observe from the objective values (Figure S.C.3).

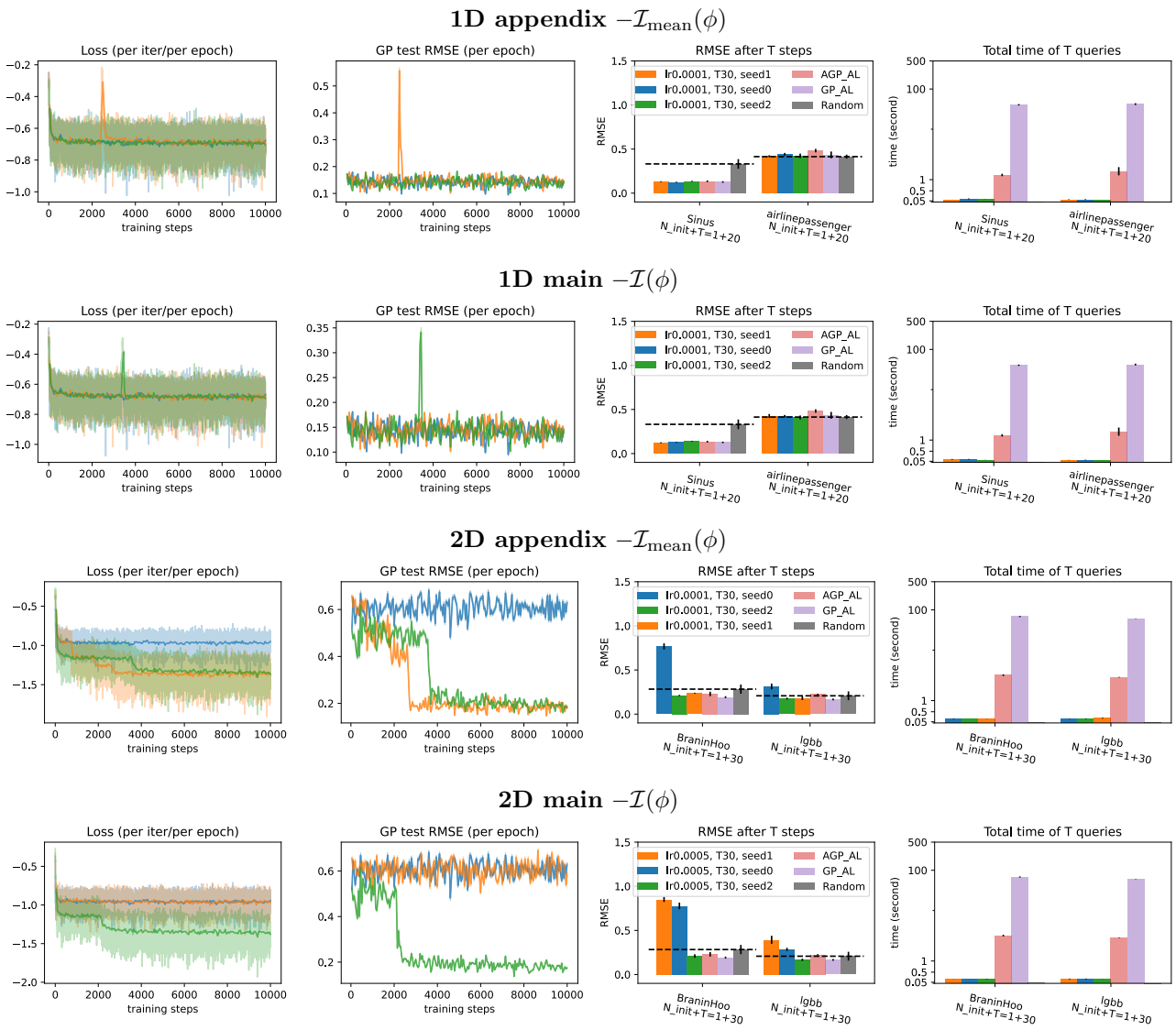

Figure S.G.5: **Unconstrained AL results of objectives main $\mathcal{I}$ (Eq. (5)) and appendix $\mathcal{I}_{\text{mean}}$ (Eq. (S.8)).** The numerical congifurations (e.g. D, T) are identical for both objectives (Tables S.E.1 to S.F.3).We deploy the policy with $T = 20$ for $1D$ and $T = 30$ for $2D$ problems, and further results on $5D$ problems are shown in Figure S.G.6. The first column shows the training losses $-\mathcal{I}$ or $-\mathcal{I}_{\text{mean}}$ per training step and per epoch mean (mean of 50 steps). At the end of each epoch, we sample a batch of GP functions, not for training, but we deploy the policy for T steps (e.g. train $T_{\text{sim}} \leq T = 30$, then deploy $T = 30$) and evaluate the GP RMSE against the ground truth. The GP RMSEs per training epoch are shown in the second column. After the training, we deploy each policy on benchmark problems and the results are the third and fourth columns (time illustrated in seconds).

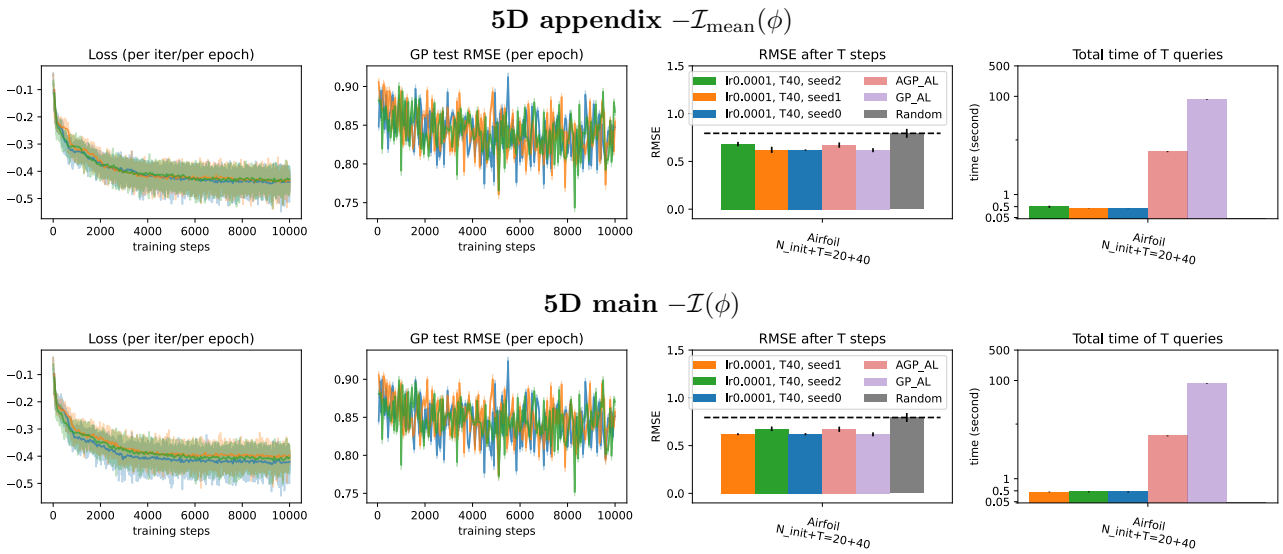

Figure S.G.6: **Unconstrained 5D AL results of objectives main $\mathcal{I}$ (Eq. (5)) and appendix $\mathcal{I}_{\text{mean}}$ (Eq. (S.8)).** We deploy the policy with $T = 40$ for $5D$ problems. The columns are illustrate as Figure S.G.5.

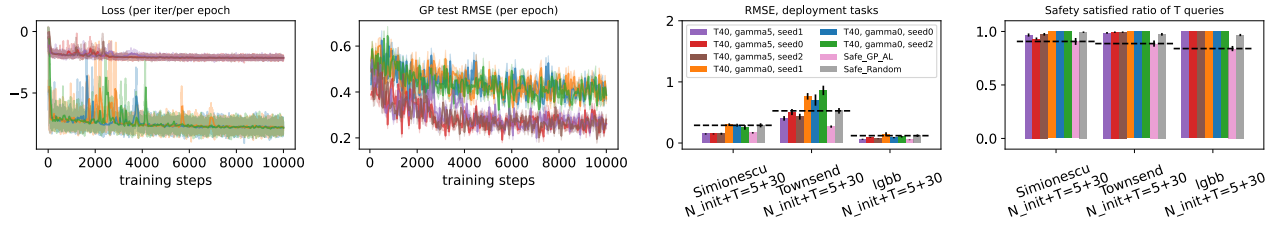

Figure S.G.7: **The main safe AL objective $\mathcal{S}_{\mathcal{H}}$ with $\gamma = 0.05$ vs $\gamma = 0$, on $2D$ problems.** The training loss values $-\mathcal{S}_{\mathcal{H}}$ are dominated by the minimized likelihood of unsafe queries if $\gamma = 0$ (first column), and safety is thus prioritized more than the AL exploration in the training, as indicated by the RMSEs against ground truth GPs (second column). After the training, the policies are deployed on three problems, and the results show similar exploration and safety trade-off as observed during the training (third and fourth columns). The deployment colored bars are sorted according to the averaged training losses of the last 10 epochs (left to right: highest to lowest losses). The baseline methods Safe GP AL and Safe Random are deployed with $\gamma = 0.05$. When $\gamma = 0$, a conventional safe AL cannot operates because the entire input space will be identified unsafe.

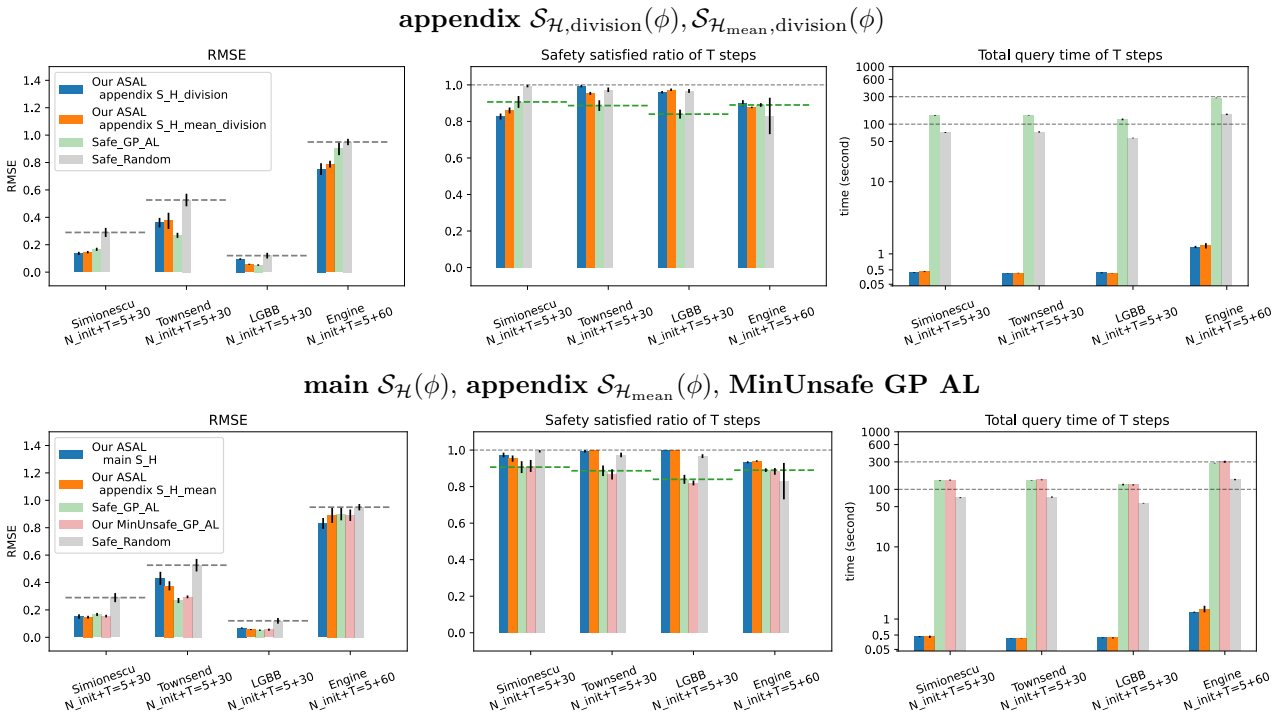

Figure S.G.8: **Results of different amortized safe AL objectives and the safe AL of MinUnsafe criterion.** Our safe AL policies are trained on four objectives: main $\mathcal{S}_{\mathcal{H}}, \gamma = 0.05$, appendix $\mathcal{S}_{\mathcal{H}_{\text{mean}}}, \gamma = 0.05$, and appendix $\mathcal{S}_{\mathcal{H},\text{division}}, \mathcal{S}_{\mathcal{H}_{\text{mean}},\text{division}}$ which have no γ clamping. As ordered in Figure S.C.3, $\mathcal{S}_{\mathcal{H},\text{division}}, \mathcal{S}_{\mathcal{H}_{\text{mean}},\text{division}}$ are on the top and $\mathcal{S}_{\mathcal{H}}, \mathcal{S}_{\mathcal{H}_{\text{mean}}}$ are at the bottom. The MinUnsafe acquisition function corresponds to $\mathcal{S}_{\mathcal{H}}, \mathcal{S}_{\mathcal{H}_{\text{mean}}}$ objectives and is thus shown at the bottom.