In-Context Black-Box Optimization with Unreliable Feedback

Nicolas S. Blumer
ELLIS Institute Finland
Aalto University

Julien Martinelli
ELLIS Institute Finland
Aalto University

Samuel Kaski
ELLIS Institute Finland
Aalto University
University of Manchester

{firstname.lastname}@aalto.fi

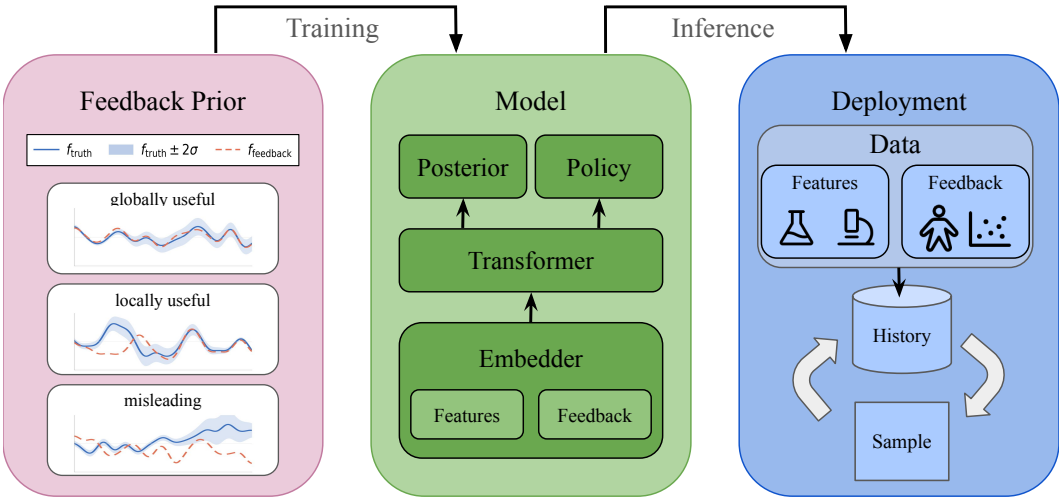
Abstract

Black-box optimization in science and engineering often comes with side information: experts, simulators, pretrained predictors, or heuristics can suggest promising candidates. Such feedback can accelerate search, but can also be biased, input-dependent, or misleading. Feedback-aware Bayesian optimization methods typically handle tasks independently, limiting adaptation across feedback sources. In-context optimizers enable cross-task adaptation, but usually rely only on optimization history at test time. We study *feedback-informed in-context black-box optimization* (FICBO), where a pretrained optimizer conditions on both the observed history and cheap auxiliary feedback for the current candidate set. We introduce structured priors over how auxiliary feedback relates to the objective, and use them to pretrain a feedback-aware transformer across regimes ranging from highly informative to misleading. At test time, the model estimates source reliability in context by comparing observed objective values with auxiliary signals, improving query selection. On synthetic and real-world tasks, FICBO effectively exploits informative feedback while remaining robust to weak or misleading sources. Empirical investigations further illustrate how the model perceives test-time sources, offering insights into its interpretability and decision-making process.

1 Introduction

Optimizing expensive black-box experiments is central to scientific and engineering applications, including biology [1], chemistry [2], and materials science [3]. In such domains, experts can guide search through manual assessment, hand-crafted heuristics, or surrogate models, but their feedback can be systematically unreliable [4, 5]. Expertise may be reliable only in familiar regions, while proxies can omit relevant mechanisms, for example when a chemical reaction model ignores a competing side reaction that becomes important in part of the design space, or extrapolate poorly beyond their design conditions. Feedback from a related but distinct task may likewise transfer imperfectly, as when CIFAR-10 accuracy ranks architectures differently from ImageNet16-120. Such feedback is rarely uniformly good or bad: it may be accurate for some tasks or regions and misleading in others, with its reliability unknown in advance and potentially changing as search progresses.

A natural framework for optimizing expensive objectives under such imperfect guidance is Bayesian optimization (BO), which models the unknown objective with a probabilistic surrogate, often a Gaussian process [16], and selects the next design through an acquisition function that balances exploration and exploitation [6]. This probabilistic and sequential structure makes BO particularly well suited to settings with auxiliary feedback, since such information can shape inference and guide design selection. Existing feedback-aware BO methods exploit expert knowledge through informative priors or user beliefs over the objective [2, 7, 4, 8]. Some reweight the acquisition function using a static user prior whose influence is controlled by a user-specified parameter [7, 17]. Yet the appropriate influence depends on source reliability, which is unknown a priori and may vary across the search space.



Methods	Surrogate amortized	Query policy amortized	Auxiliary feedback at test time	Feedback-aware pretraining
Vanilla BO [6]	✗	✗	✗	N. A.
πBO [7], CoExBO [8]	✗	✗	✓	N. A.
PFNs4BO [9], GIT-BO [10]	✓	✗	✗	✗
BoFormer [11], α-PFN [12]	✗	✓	✗	✗
NAP [13], PABBO [14], TAMO [15]	✓	✓	✗	✗
FICBO (This work)	✓	✓	✓	✓

Figure 1: **Feedback-informed in-context black-box optimization (FICBO) and related work.** During training, tasks are sampled from a feedback prior with auxiliary sources of varying reliability. A transformer learns both posterior prediction and query selection from features and feedback. At deployment, it conditions on the optimization history and the available feedback to guide search.

Amortized optimization offers a natural way to learn this feedback-adaptive behavior across tasks. By pretraining a neural optimizer on a distribution of synthetic tasks with diverse feedback, one can move part of the optimization burden offline and encode assumptions about the expected problem class directly into the task prior. The resulting model can then adapt *in context* to a new problem from its optimization history alone, without task-specific retraining. Recent work has shown this paradigm to be highly effective in settings such as Bayesian experimental design and BO [18, 13, 19, 15, 20].

However, existing amortized optimizers typically take only the standard optimization history as input. They are not designed for settings where an auxiliary source is available at test time and may help on some tasks, be irrelevant on others, or induce negative transfer if overused [4]. We therefore aim to learn through pretraining how such feedback should modulate posterior inference and query selection.

Contributions. We introduce feedback-informed in-context black-box optimization (FICBO), an in-context black-box optimization framework for settings with unreliable auxiliary feedback, using expert sources as a primary example (Figure 1). Concretely, we contribute: (1) conceptually, a formulation in which an optimizer receives auxiliary feedback at test time and must infer its reliability on-the-fly; (2) methodologically, a feedback-aware amortized black-box optimization framework pretrained under a feedback prior over source reliability; and (3) empirically, an evaluation protocol spanning synthetic and real tasks, with ablations over feedback strength and bias type, comparisons against classical and amortized baselines, and empirical investigations that illustrate how the model perceives test-time sources.

2 Background: *in-context* black-box optimization

We consider the black-box optimization problem

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

where $\mathcal{X} \subseteq \mathbb{R}^d$ is the design space and $f : \mathcal{X} \to \mathbb{R}$ is unknown and accessible only through expensive evaluations. At round t, querying $\boldsymbol{x}_t$ yields a noisy observation $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$ with $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$, and the history after t rounds is $H_t = \{(\boldsymbol{x}_i, y_i)\}_{i=0}^{t-1}$.

Classical methods solve each optimization problem independently. In contrast, *in-context* black-box optimization leverages commonalities across related problems by shifting computation offline. This motivates defining a distribution over related tasks, $\tau \sim p(\tau)$, where each task has its own objective f_τ. By pretraining across such tasks, a neural optimizer can adapt at test time to a new problem directly from its observed history, without requiring task-specific retraining.

We focus on a pool-based setting, which is the standard setting in the amortized BO/Bayesian Experimental Design (BED)/Active Learning (AL) literature (e.g., [18, 19, 15, 14, 21]). Each task τ is associated with a finite candidate set $Q^\tau = \{\boldsymbol{x}_j^\tau\}_{j=1}^{N_\tau}$. We discuss how our approach extends to the continuous setting in Appendix F. At round t, the optimizer selects $\boldsymbol{x}_t^\tau \in Q_t^\tau \subseteq Q^\tau$, observes $y_t^\tau = f_\tau(\boldsymbol{x}_t^\tau) + \varepsilon_t^\tau$ with $\varepsilon_t^\tau \sim \mathcal{N}(0, \sigma_\tau^2)$, and updates the corresponding history $H_t^\tau = H_0^\tau \cup \{(\boldsymbol{x}_i^\tau, y_i^\tau)\}_{i=0}^{t-1}$. The optimizer comprises a predictive component

$$q_\phi(y \mid \boldsymbol{x}, H_t^\tau), \tag{2}$$

which estimates the value of a candidate $\boldsymbol{x} \in Q_t^\tau$, and a policy that directly selects the next query,

$$\boldsymbol{x}_t^\tau \sim \pi_\psi(\cdot \mid H_t^\tau, Q_t^\tau), \tag{3}$$

with $Q_0^\tau = Q^\tau$ and $Q_{t+1}^\tau = Q_t^\tau \setminus \{\boldsymbol{x}_t^\tau\}$. The policy is the primary object, as it determines which candidate is evaluated next. The predictive model serves as an auxiliary training signal, providing predictions with uncertainty for interpretability and potentially acting as a faster GP surrogate [13, 19, 15].

Training proceeds offline over trajectories sampled from $p(\tau)$, optimizing

$$\max_\psi \ \mathbb{E}_{\tau \sim p(\tau)}\Big[\sum_{t=0}^{T_\tau - 1} r_t^\tau \gamma^t\Big], \tag{4}$$

where $y_t^* = \max\{y : (\boldsymbol{x}, y) \in H_t^\tau\}$ denotes the best value observed strictly before round t, and $r_t^\tau = \max(y_t^\tau - y_t^*,\ 0)$ is the improvement of the t-th query over it. $\gamma \in (0, 1]$ is the discount factor. Unlike classical BO, which maximizes an acquisition function at each step, the in-context optimizer is trained end-to-end to maximize cumulative reward over the full trajectory, replacing repeated model fitting and acquisition optimization with a learned procedure [13, 19, 15].

3 Method

We extend the *in-context* black-box optimization setup (Section 2) to a feedback-aware setting, where the optimizer also receives candidate-wise auxiliary feedback over the pool,

$$U^\tau = \{u^\tau(\boldsymbol{x}) : \boldsymbol{x} \in Q^\tau\}. \tag{5}$$

We assume feedback is available for every point in the query set [1].

The signal U^τ conveys source-specific information about candidate quality rather than ordinary input features. Its usefulness may vary across tasks and regions of the design space: $u^\tau(\boldsymbol{x})$ may be noisy, systematically biased, only locally informative, or misleading. The central challenge is to infer from the observed optimization history how strongly this signal should influence search on the current task.

Rather than a single feedback model, we construct two complementary priors, each representing a different way feedback can go wrong: access to a different, incomplete set of features that only partially explains the objective, or a wrong interpretation of how latent factors of the task should be weighted. On top of each, we apply additional distortions that further degrade the signal. Together, they are designed to cover a broad range of feedback, from uninformative to highly reliable. These priors were designed to be general and serve as a starting point, and practitioners can extend them to reflect their own knowledge and uncertainty about the feedback they expect to encounter.

3.1 Priors over feedback mechanisms

Following the amortized pretraining paradigm [19, 14, 23, 21], we pretrain on synthetic optimization episodes sampled from a joint prior over objectives and auxiliary feedback sources. Each task is

$$\tau = \big(Q_{\text{full}}^\tau,\ P_{\text{model}}^\tau,\ H_0^\tau,\ f_\tau,\ h_\tau,\ T_\tau\big), \tag{6}$$

[1] If the source is too expensive to query exhaustively, a cheap surrogate fit to a small number of labeled points can be used instead, as in preference elicitation [8, 14, 22], at the cost of additional bias. The model-based source (an amortized GP fit to source-private data) described in Sec. 3.1 is designed to let the model learn to discount exactly this kind of bias.

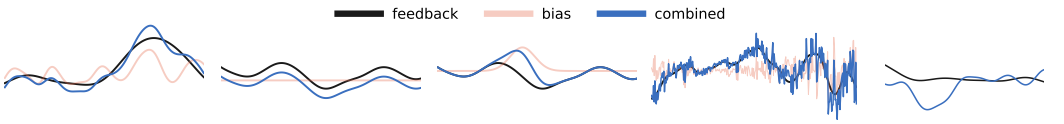


Figure 2: Illustration of five signal degradation bias types applied to a feedback GP.

where $Q^\tau_{\text{full}} = \{\boldsymbol{x}^{\text{full}}_j\}_{j=1}^{N_\tau}$ is a latent candidate pool, P^τ_{model} is the projection whose output determines visibility to the optimizer, H^τ_0 is the initial history, f_τ is the objective, h_τ is the feedback-generating map, and T_τ is the horizon. The optimizer observes only Q^τ, its feedback values, and the outcomes it collects. The features removed by the projection P^τ_{model}, along with the various latent functions introduced later, are all hidden from the optimizer. The optimizer sees only

$$\boldsymbol{x}^\tau_j = P^\tau_{\text{model}} \boldsymbol{x}^{\text{full}}_j, \qquad Q^\tau = \{\boldsymbol{x}^\tau_j : \boldsymbol{x}^{\text{full}}_j \in Q^\tau_{\text{full}}\}, \tag{7}$$

while objective and feedback values are generated as $y^\tau_j = f_\tau(\boldsymbol{x}^{\text{full}}_j) + \varepsilon^\tau_j$ and $u^\tau_j = h_\tau(\boldsymbol{x}^{\text{full}}_j)$. The source parameters and bias parameters are unobserved. We use expert and source interchangeably, as the feedback source is directly or indirectly an expert.

Every task follows the same three stages: (i) a prior generates the objective f_τ together with an *ideal* source signal g_τ which is informative about f_τ, but usually not identical to it; (ii) a degradation $\mathcal{B}_\tau$ gives $\tilde{g}_\tau = \mathcal{B}_\tau[g_\tau]$ (Fig. 2); (iii) optionally, $\tilde{g}_\tau$ is replaced by a surrogate fitted to source-private data of the function. The observed feedback is $u^\tau(\boldsymbol{x}_j) = h_\tau(\boldsymbol{x}^{\text{full}}_j)$:

$$\boldsymbol{x}^{\text{full}} \longrightarrow g_\tau \xrightarrow{\mathcal{B}_\tau} \tilde{g}_\tau \xrightarrow{M_\tau \text{ (opt.)}} u, \qquad \boldsymbol{x} = P_{\text{model}}\, \boldsymbol{x}^{\text{full}}. \tag{8}$$

The two priors below differ in stage (i), i.e., in *why* the source is imperfect: it sees different inputs than the optimizer (additive prior), or the source misinterprets the relationship of latent factors (mixture prior). Stages (ii)–(iii) can be applied to both, but we only applied (iii) to the additive prior.

Additive prior. For the additive prior, we randomize the visibility pattern by sampling the number of source-visible input dimensions d^τ_{src} and an overlap o^τ between the optimizer's and source's features, giving $d^\tau_{\text{full}} = d + d^\tau_{\text{src}} - o^\tau$ where d is the optimizer input dimension:

$$\boldsymbol{x}^{\text{full}} = \big(\boldsymbol{x}_{\text{model}}, \boldsymbol{x}_{\text{shared}}, \boldsymbol{x}_{\text{src}}\big), P_{\text{model}}\boldsymbol{x}^{\text{full}} = (\boldsymbol{x}_{\text{model}}, \boldsymbol{x}_{\text{shared}}), P_{\text{src}}\boldsymbol{x}^{\text{full}} = (\boldsymbol{x}_{\text{shared}}, \boldsymbol{x}_{\text{src}}). \tag{9}$$

Here $\boldsymbol{x}_{\text{model}} \in \mathbb{R}^{d-o^\tau}$ is optimizer-only, $\boldsymbol{x}_{\text{src}} \in \mathbb{R}^{d^\tau_{\text{src}}-o^\tau}$ is source-only, and $\boldsymbol{x}_{\text{shared}} \in \mathbb{R}^{o^\tau}$ is shared. This generates tasks where feedback may contain information absent from the optimizer's input yet remain incomplete or biased. d is fixed per trained model (we train one variant per $d \in \{1, 2, 3\}$), while d^τ_{src} and o^τ are resampled per task: the optimizer sees the first d of $d^\tau_{\text{full}} = d + d^\tau_{\text{src}} - o^\tau$ coordinates, the source the last d^τ_{src}, so the two views overlap in exactly o^τ coordinates.

The objective combines two independent GP components: one depending on features visible only to the optimizer, and one depending on features visible to the source as described above. We sample $f^{\text{model}}_\tau \sim \mathcal{GP}(0, k^{\text{model}}_\tau)$ on $\boldsymbol{x}_{\text{model}}$ and $f^{\text{src}}_\tau \sim \mathcal{GP}(0, k^{\text{src}}_\tau)$ on $(\boldsymbol{x}_{\text{shared}}, \boldsymbol{x}_{\text{src}})$, and set

$$f_\tau(\boldsymbol{x}^{\text{full}}) = \frac{(1-w_\tau)\, f^{\text{model}}_\tau(\boldsymbol{x}_{\text{model}}) + w_\tau\, f^{\text{src}}_\tau(\boldsymbol{x}_{\text{shared}}, \boldsymbol{x}_{\text{src}})}{\sqrt{(1-w_\tau)^2 + w_\tau^2}}, \qquad w_\tau \in [0, 1]. \tag{10}$$

The denominator keeps $\mathrm{Var}[f_\tau]$ between the component variances for every w_τ, so the output scale of f_τ reveals little about w_τ. Relevance must be inferred from the history. The source's ideal signal is $g_\tau(\boldsymbol{x}^{\text{full}}) = f^{\text{src}}_\tau(\boldsymbol{x}_{\text{shared}}, \boldsymbol{x}_{\text{src}})$, so w_τ controls source relevance: large w_τ makes the source-visible component central to the objective, while small w_τ makes it accurate but weakly relevant.

The ideal signal is then distorted as $\tilde{g}_\tau = \mathcal{B}_\tau[g_\tau]$, where $\mathcal{B}_\tau$ may add noise, global shifts, smooth GP-shaped bias, local reliability effects, or rare replacements; examples are shown in Figure 2. The distorted signal can be used directly, or, in the model-based branch, passed through a source model (an amortized GP) conditioned on additional source-only observations.

$$\mathcal{D}^\tau_{\text{src}} = \big\{\big(P_{\text{src}}\boldsymbol{x}^{\text{full}}_i, \tilde{g}_\tau(\boldsymbol{x}^{\text{full}}_i)\big)\big\}_{i=1}^{n_{\text{src}}}, \qquad M_\tau = \mathrm{Fit}\big(\mathcal{D}^\tau_{\text{src}}\big). \tag{11}$$

Here Fit denotes conditioning a source model on $\mathcal{D}^\tau_{\mathrm{src}}$; in pretraining, this is an amortized GP whose predictive mean gives the feedback. The feedback map is

$$h_\tau(\boldsymbol{x}^{\mathrm{full}}) = \begin{cases} \tilde{g}_\tau(\boldsymbol{x}^{\mathrm{full}}), & \text{direct source,} \\ M_\tau(P_{\mathrm{src}}\boldsymbol{x}^{\mathrm{full}}), & \text{model-based source.} \end{cases} \tag{12}$$

The source model is initialized once from $\mathcal{D}^\tau_{\mathrm{src}}$, separate from the optimizer's data, and held fixed during the BO episode (the test-time XGBoost expert is also fit once, on source-only points, Appendix C.2).

Mixture prior. The second variant makes reliability a matter of component weighting rather than feature access. Let $K_\tau = K_{\mathrm{real}} + K^\tau_{\mathrm{decoy}}$ and sample independent GP components $z_{\tau,1}, \ldots, z_{\tau,K_\tau}$ on $\boldsymbol{x}^{\mathrm{full}}$. Only the first K_{real} components determine the objective:

$$f_\tau(\boldsymbol{x}^{\mathrm{full}}) = \frac{1}{\sqrt{K_{\mathrm{real}}}} \sum_{k=1}^{K_{\mathrm{real}}} z_{\tau,k}(\boldsymbol{x}^{\mathrm{full}}). \tag{13}$$

The source forms feedback from a masked and reweighted combination of real and decoy components. With $m_{\tau,1} = 1$, $m_{\tau,k} \sim \mathrm{Bernoulli}(p_{\mathrm{src}})$ for $k > 1$, and $S_\tau = \{k : m_{\tau,k} = 1\}$, we sample

$$\bar{\boldsymbol{a}}_{\tau,S_\tau} \sim \mathrm{Dirichlet}\big(\alpha_{\mathrm{src}}\mathbf{1}_{|S_\tau|}\big), \qquad a_{\tau,k} = \frac{\bar{a}_{\tau,k}}{\|\bar{\boldsymbol{a}}_{\tau,S_\tau}\|_2}\,\mathbb{I}\{k \in S_\tau\} \tag{14}$$

Fixing $m_{\tau,1} = 1$ means the source is never entirely uninformative by construction, and the ℓ_2 normalization gives the source weights unit norm, like the weights $1/\sqrt{K_{\mathrm{real}}}$ of f_τ, so that g_τ and f_τ are comparably scaled. The source signal and feedback are then

$$g_\tau(\boldsymbol{x}^{\mathrm{full}}) = \sum_{k=1}^{K_\tau} a_{\tau,k}\, z_{\tau,k}(\boldsymbol{x}^{\mathrm{full}}), \qquad h_\tau = \mathcal{B}_\tau[g_\tau], \tag{15}$$

where $\mathcal{B}_\tau$ may again be the identity or one of the degradation operators in Figure 2. This prior produces coherent but unreliable sources: the source may ignore real components, overemphasize some of them, or attend to decoys that do not affect the objective.

Inference problem induced by the prior. In both priors, the learner receives the pool feedback U^τ and the evaluated history H^τ_t, but not w_τ, hidden coordinates, latent components, source weights, bias parameters, or source-private data. Here u^τ_j denotes the feedback attached to the projected candidate $\boldsymbol{x}^\tau_j$. Source reliability must therefore be inferred in context, from how feedback values agree with observed objective evaluations. Source-data sampling scheme details are deferred to Appendix B.

3.2 Feedback-aware model

For a task τ at round $t < T_\tau$, each point $\boldsymbol{x} \in Q^\tau$ carries both its features and auxiliary feedback $u^\tau(\boldsymbol{x})$. Evaluated points additionally carry an observed outcome y, giving an augmented history $\widetilde{H}^\tau_t = \widetilde{H}^\tau_0 \cup \{(\boldsymbol{x}^\tau_i, u^\tau_i, y^\tau_i)\}_{i=0}^{t-1}$, where $\widetilde{H}^\tau_0$ is H^τ_0 with feedback attached. We embed each point into a token via learned embedders e_x, e_u, e_y,

$$\boldsymbol{E}^{\mathrm{ctx}} = \mathrm{Concat}(e_x(\boldsymbol{x}), e_u(u)) + e_y(y), \qquad \boldsymbol{E}^{\mathrm{qry}} = \mathrm{Concat}(e_x(\boldsymbol{x}), e_u(u)), \tag{16}$$

where context tokens use $(\boldsymbol{x}, u, y)$ from $\widetilde{H}^\tau_t$, and query tokens use $(\boldsymbol{x}, u)$ for $\boldsymbol{x} \in Q^\tau_t$ with $u = u^\tau(\boldsymbol{x})$. A shared transformer backbone maps them to contextualized representations $\widehat{\boldsymbol{E}}^{\mathrm{ctx}}$ and $\widehat{\boldsymbol{E}}^{\mathrm{qry}}$, with query tokens attending to both the context and the full query set. This allows each contextualized candidate representation $\widehat{\boldsymbol{E}}^{\mathrm{qry}}$ to depend on both the optimization history and the pool-wide pattern of auxiliary feedback.

Two heads operate on top of the backbone. The *policy head / acquisition head* maps each contextualized query token to a scalar logit, inducing a categorical distribution over Q^τ_t,

$$\pi_\psi(\boldsymbol{x}^\tau_j \mid \widetilde{H}^\tau_t, U^\tau) \propto \exp\big(\mathrm{MLP}_\psi(\widehat{\boldsymbol{E}}^{\mathrm{qry}}_j)\big). \tag{17}$$

The *inference head* maps each query token to an L-component Gaussian mixture,

$$q_\phi(y \mid \boldsymbol{x}, \widetilde{H}^\tau_t, U^\tau) = \sum_{\ell=1}^{L} \omega_\ell(\boldsymbol{x})\, \mathcal{N}(y; \mu_\ell(\boldsymbol{x}), \sigma^2_\ell(\boldsymbol{x})), \tag{18}$$

Task	d	Objective	Feedback	Alignment
Synthetic, marginal expert	2	13 analytic functions	objective marginalized over source-hidden dims	incomplete
Synthetic, XGBoost expert	2	13 analytic functions	shallow regressor on the source dim	biased
Branin (multi-fidelity)	2	high-fidelity Branin	low-fidelity variant	tunable (ρ)
Airfoil Small	3	C_L/C_D, large NeuralFoil	C_L/C_D, small NeuralFoil	aligned, noisy
Airfoil Lift	3	C_L/C_D, large NeuralFoil	C_L only	structurally biased
Neural Architecture Search	3	ImageNet16-120 accuracy	CIFAR-10 accuracy	optimum differs
Reactor	2	yield of intermediate $[B]$	conversion of reactant A	misleading late

Table 3: **Evaluation tasks.** d is the input dimensionality seen by the optimizer (discrete for NAS). Synthetic tasks use a full dimensionality of 2 or 3: the optimizer sees the first two dimensions and the source only the last, which gives one shared dimension at full dimensionality 2 and none at 3. The pool size is 300 (123 for NAS) and the horizon is $T = 30$ for all tasks.

and can be used in place of the policy head by pairing it with a standard acquisition function α, giving $\boldsymbol{x}_t^\tau = \arg\max_{\boldsymbol{x}\in Q_t^\tau} \alpha(\boldsymbol{x}; q_\phi, \widetilde{H}_t^\tau, U^\tau)$.

Training. Both heads are trained jointly after a warm-up on synthetic episodes sampled from a task prior. The inference head minimizes a step-wise negative log-likelihood over held-out targets $\mathcal{T}^\tau$, whose labels are never included in $\widetilde{H}_t^\tau$.

$$\mathcal{L}_{\text{pred}}(\phi) = -\mathbb{E}_\tau \left[\frac{1}{T_\tau} \sum_{t=0}^{T_\tau - 1} \frac{1}{|\mathcal{T}^\tau|} \sum_{(\boldsymbol{x},y)\in\mathcal{T}^\tau} \log q_\phi(y \mid \boldsymbol{x}, \widetilde{H}_t^\tau, U^\tau) \right]. \tag{19}$$

$\mathcal{L}_{\text{pred}}$ also contains, with equal weight, the NLL of $d+1$ latent GP hyperparameters (Additive prior: the optimizer-visible lengthscales and the output scale of f_τ^{model}; Mixture prior: the visible lengthscales and the output scale of $z_{\tau,1}$).

We cast policy learning as a finite-horizon MDP where the state at step t is the augmented history and remaining pool, and optimize with REINFORCE using the improvement reward $r_t^\tau = \max(y_t^\tau - y_t^*,\ 0)$. Further details can be found in section D.1. The policy loss is then

$$\mathcal{L}_{\text{pol}}(\psi) = -\mathbb{E}_{\tau,\pi_\psi} \left[\frac{1}{T_\tau} \sum_{t=0}^{T_\tau - 1} \log \pi_\psi(\boldsymbol{x}_t^\tau \mid \widetilde{H}_t^\tau, U^\tau) \tilde{R}_t^\tau \right], \tag{20}$$

where $\tilde{R}_t^\tau$ is the discounted return $G_t^\tau = \sum_{t'\geq t} \gamma^{t'-t} r_{t'}^\tau$, normalized across the batch at each timestep to reduce gradient variance (full definition in Appendix D.1). The code is available on GitHub.

4 Related work

Expert knowledge and feedback in Bayesian optimization and sequential design. A substantial literature studies how external expertise can improve BO and Bayesian experimental design, whether through informative priors, interactive feedback, or auxiliary predictive sources [24]. Prior-guided approaches such as πBO incorporate user beliefs about promising regions or the optimum location directly into the optimization procedure [7], while other methods use interactive preference elicitation or corrective human feedback during the search [8]. Closely related, robust multi-fidelity formulations show that a human expert can act as a cheap but imperfect signal alongside expensive ground-truth evaluations [4]. In our setting, however, the auxiliary signal is available over the candidate pool; the challenge is to amortize how its reliability is interpreted across tasks. Expert-informed BO has already found concrete applications in scientific discovery, for instance, in drug-design settings where chemist knowledge or domain feedback is incorporated into the optimization loop [25, 2]. Overall, these works illustrate the value of expert guidance, but they remain task-specific: they do not learn, across a distribution of tasks, how to interpret and calibrate a potentially biased feedback channel.

***In-context* black-box optimization.** A growing line of work amortizes parts of the BO loop using neural in-context models. Several works, for instance, replace the standard GP surrogate

with transformer-based models that infer task-specific predictive distributions in a single forward pass [23, 9, 26, 10]. Other approaches instead amortize the acquisition mechanism itself, learning to propose queries directly from optimization histories rather than optimizing a handcrafted acquisition function at test time [11, 12]. Beyond such partial amortization, recent work targets end-to-end amortized optimization, from early neural acquisition processes [13] to preferential optimization with pairwise feedback [14] and multi-objective optimization [15]. Other advances leverage representations from pretrained tabular foundation models to learn acquisition policies, reusing their predictive capabilities rather than training the entire optimizer from scratch [27]. Finally, while many of these methods rely on discrete candidate grids, recent work uses flow matching to generate queries directly in continuous search spaces, paving the way to higher dimensional problems [20]. Yet these methods typically assume a single reliable information channel. Here, by contrast, we consider an additional expert-feedback source whose usefulness cannot be taken for granted.

5 Experiments

We evaluate whether feedback-aware pretraining lets an in-context optimizer exploit useful auxiliary signals while remaining robust to misleading ones. Our evaluation spans analytic benchmark functions with controlled feedback, and real-world tasks where the auxiliary signal is a cheap but imperfect proxy for the expensive objective. All methods are run in the same pool-based setting. We run each experiment on 100 independent replicas (one fixed seed, a batch of 100 environments), resampling candidate pools, initial observations, and observation noise (0.1 std.) while keeping them shared across methods within each replica. Each step's simple regret is divided by the gap between the best value in the candidate pool and the best initial observation, which gives a common $[0, 1]$ scale; cumulative regret sums this normalized simple regret over the T steps.

Baselines. We compare the two variants of our method, FICBO (— Additive Prior; — Mixture Prior), against three families of in-context baselines. First, a feedback-agnostic baseline uses the same in-context optimization architecture, but does not receive the feedback channel (— GP-prior ICBO (in-context black-box optimizer)). Second, a naive feedback-feature baseline concatenates $u(\boldsymbol{x})$ to the input features, testing whether standard amortization can exploit feedback without source-aware pretraining (— GP-prior ICBO + feedback feature). Third, πBO-like baselines reweight EI or UCB scores from the GP-prior ICBO surrogate using feedback $u(\boldsymbol{x})$ (— πBO-UCB; — πBO-EI [7]). Overall, baselines differ only in how the feedback channel is used. We also include a greedy feedback baseline (— Feedback), evaluating candidates in decreasing feedback score, and random search (— Random). Details in Appendix C.1, including additional baselines from transfer learning.

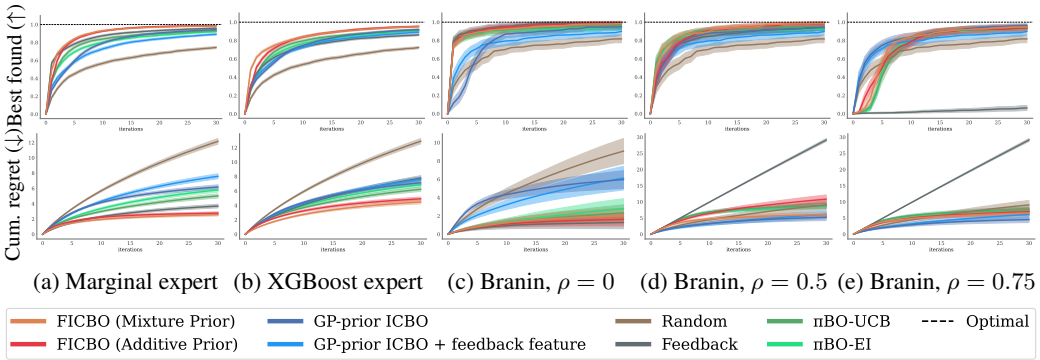


Figure 3: **Performance across synthetic benchmarks and multi-fidelity Branin tasks. Top row:** normalized best objective found so far. **Bottom row:** normalized cumulative regret. **(a–b)** Results averaged over 13 benchmark functions (Sphere, Ackley, Rastrigin, Rosenbrock, Schwefel, Levy, 2D and 3D versions, and Hartmann 3D) under two feedback regimes: marginal expert (observes only the last input dimension) and XGBoost expert (data-driven but informationally limited). **(c–e)** Results for multi-fidelity Branin function under three discrepancy levels ($\rho \in \{0, 0.5, 0.75\}$), where larger ρ increases the mismatch between feedback and objective. Mean $\pm$ 95% bootstrap confidence intervals.

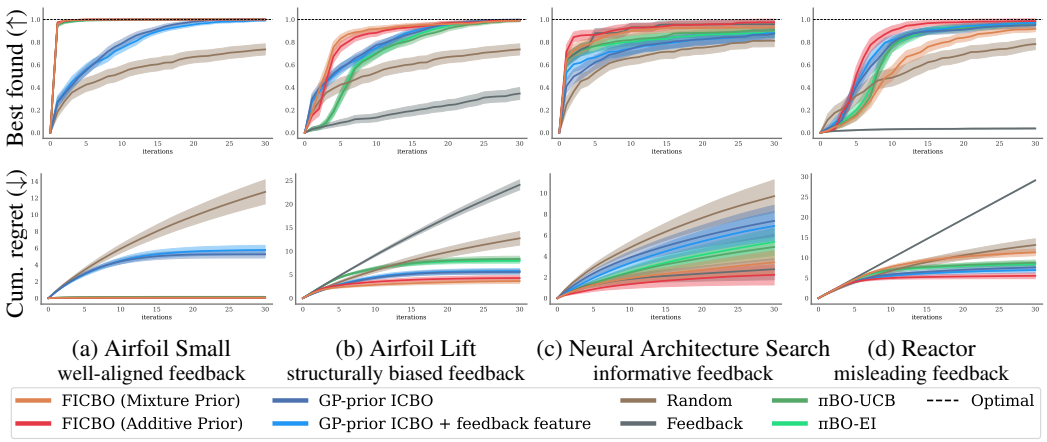


Figure 4: **Real-world optimization tasks. Top row:** normalized best value found. **Bottom row:** normalized cumulative regret. **(a)** Airfoil optimization with a cheaper neural surrogate as feedback (Airfoil Small); **(b)** airfoil optimization with lift-only feedback (Airfoil Lift); **(c)** Neural Architecture Search (NAS) with a cheap auxiliary performance signal; and **(d)** reactor optimization with conversion-based feedback. Mean $\pm$ 95% bootstrap confidence intervals. **FICBO exploits aligned feedback, but shows prior-dependent robustness to structurally biased proxies in the Reactor example.** The curves in the Airfoil Small panel overlap: FICBO and Feedback find the best solution within the first steps because the feedback is well aligned; πBO is slightly slower (Table 4).

5.1 Results

5.1.1 Synthetic benchmarks

Figure 3**a-b** first evaluates out-of-distribution analytic functions under the two feedback regimes of Table 3. The *marginal expert* gives a signal that is directionally useful but incomplete, while the *XGBoost expert* is additionally limited by its training data and model class (section C.2).

The results show a hierarchy in how feedback is used. First, in both regimes, the naive feedback-feature baseline improves over the feedback-agnostic amortized model (— > —), indicating that the auxiliary signal contains exploitable information even when treated as an ordinary input coordinate. Second, the πBO variants (— πBO-UCB; — πBO-EI) improve over the generic amortized prior by using feedback $u(\boldsymbol{x})$ as a factor on the acquisition function, suggesting it is a stronger inductive bias than simple feature concatenation (GP-prior ICBO + feedback feature) when feedback is informative. Finally, FICBO (— Additive; — Mixture) achieves the lowest regret in both regimes. Unlike πBO, whose feedback influence follows a fixed schedule, FICBO is pretrained under a prior over source usefulness and can infer from the observed history how much the current source should be trusted. Comparisons to more baselines can be found in section C.1.1.

The XGBoost regime is harder because feedback errors include extrapolation and model misspecification rather than only marginalization, but the same qualitative ordering remains, suggesting that the learned trust calibration transfers beyond the exact synthetic prior.

Lastly, Figure 3**c–e** shows settings where feedback is a smooth low-fidelity variant with increasing discrepancy ρ (Appendix C.3). As Figure S4 shows, larger ρ mainly corrupts the *ranking* induced by feedback, moving from near-aligned at $\rho = 0$ to rank-misaligned at $\rho = 0.75$. This explains the performance transition: feedback-aware methods benefit strongly at $\rho = 0$, FICBO remains competitive at the weakly informative intermediate setting $\rho = 0.5$, while at $\rho = 0.75$, methods that rely heavily on feedback, most noticeably the naive feedback baseline, are pulled toward misleading regions. This case suggests an out-of-prior regime with a smooth but globally misaligned proxy.

5.1.2 Real-world tasks

The real-world tasks complement the analytical benchmarks with practical optimization objectives and cheap proxies that arise naturally in each setting. The airfoil experiments optimize the lift-to-drag ratio C_L/C_D of a NACA airfoil over angle of attack, thickness, and camber. We use a higher-accuracy NeuralFoil configuration as the benchmark oracle, standing in for a more expensive aerodynamic evaluation (Appendix C.5). On Airfoil Small (Figure 4**a**), the errors of the proxy reflect only its approximation quality. The greedy feedback baseline is already strong, as expected, and FICBO

remains competitive, which shows that robustness to unreliable sources does not prevent the model from exploiting genuinely informative feedback.

Airfoil Lift (Figure 4**b**) represents a structurally biased feedback regime. The proxy uses lift alone: it captures the numerator C_L of the objective but ignores the drag penalty C_D. This is misleading because the true ratio C_L/C_D peaks before stall, where lift is high but drag remains moderate; at larger angles of attack, drag rises sharply and the ratio deteriorates. A source based only on lift can therefore keep recommending high-lift designs past the true optimum, so the feedback-only baseline accumulates large regret, whereas FICBO uses early objective evaluations to detect this mismatch and discount the proxy. This is the setting in which source-aware pretraining is most useful: the signal is a partially correct heuristic whose failure mode must be inferred in context. Next, on the Neural Architecture Search task (Figure 4**c**), CIFAR-10 accuracy transfers usefully but does not rank the best architectures correctly (Appendix C.6). On the two-dimensional reactor task (Figure 4**d**), conversion of reactant A is a cheap proxy for intermediate-product yield, but becomes misleading once high conversion favors the final product instead (Appendix C.4). In this harder regime, the additive prior is more robust than the mixture prior, suggesting that the choice of feedback prior matters when the proxy is weakly informative or structurally biased.

5.1.3 Interpreting feedback use

We use two post-hoc diagnostics to inspect how FICBO relates to the auxiliary source at test time. The first measures whether the learned policy continues to select candidates favored by the source as BO observations accumulate; the second visualizes how the feedback-error profiles of test tasks compare to the pretraining prior. Neither diagnostic is used during pretraining or deployment.

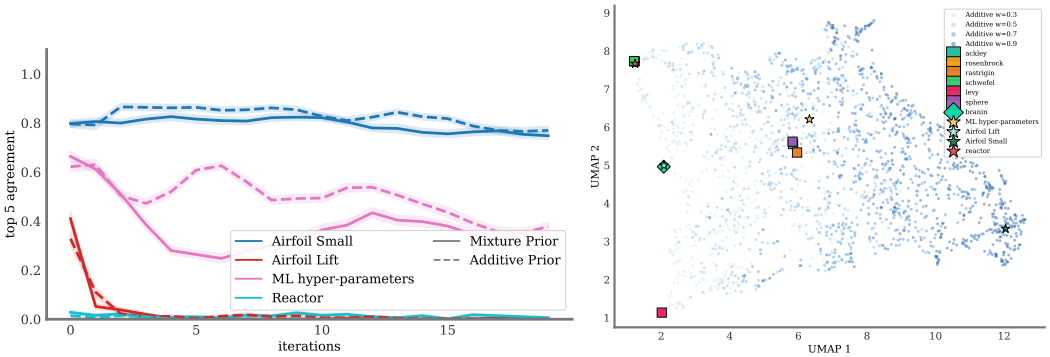


Figure 5: **Post-hoc diagnostics of feedback use. Left:** top-5 policy–feedback agreement over BO iterations for FICBO. Mean $\pm$ std computed over 100 replicas. **Right:** UMAP embedding of feedback-error profiles for synthetic prior samples and test tasks. Synthetic points are generated without additional bias operators; benchmark points use the 3D marginal-expert setting, where the expert observes the last dimension and the remaining dimensions are marginalized out.

Policy–feedback agreement. The left panel of Figure 5 reports a top-5 policy–feedback agreement diagnostic for both FICBO priors. At each BO step, we record the overlap between the policy's 5 highest-rated query points and the 5 highest auxiliary feedback points. This asks whether the policy's preferred designs are also strongly favored by the source.

On Airfoil Small, where feedback is a lower-accuracy evaluator of the same objective, agreement stays close to one for both priors, consistent with sustained use of a reliable proxy. Airfoil Lift shows the opposite pattern: agreement starts non-negligible but collapses within a few BO steps as observations reveal that lift C_L alone is misleading for C_L/C_D. The reactor task has near-zero agreement almost from the start, indicating that the policy largely avoids the conversion proxy. The Neural Architecture Search task lies between these extremes: CIFAR-10 accuracy is useful for transfer, but agreement decreases over time, because CIFAR-10 accuracy does not fully determine the ImageNet16-120 ranking, and the rate of decrease differs between priors. Overall, the curves show task- and prior-dependent source reliance rather than fixed decay. πBO imposes decay via a hand-tuned schedule [7, 8]; FICBO revises reliance from observed evidence.

Feedback-profile embedding. The right panel provides a complementary static view of the source itself. Each task is represented by the binned signed and absolute errors $d_i = u_i - y_i$, with points sorted by feedback value, and then projected with UMAP (Appendix C.7). This is a qualitative

coverage diagnostic, suggesting which test sources resemble high-alignment samples from the feedback prior and which occupy different regions of feedback–objective disagreement. For example, Airfoil Small lies near highly aligned synthetic sources, whereas biased or less directly useful proxies, such as Airfoil Lift and the reactor feedback, appear away from this regime. Finally, Appendix A.2 provides a qualitative rollout illustrating how FICBO updates its posterior and policy distribution as new observations revise its reliance on the feedback signal.

5.2 Ablation study

Table 4 contrasts well-aligned feedback (Airfoil Small) with a structurally biased proxy (Airfoil Lift). Removing bias operators barely affects Airfoil Small but clearly hurts Airfoil Lift, especially for the Mixture prior, suggesting that bias augmentation mainly improves robustness to misleading sources. The clearest ablation comes from the signal-variant rows. Using g_τ directly, without source-model fitting, degrades Airfoil Lift performance relative to the full Additive-prior model, suggesting that the fitting step helps. By contrast, generating pretraining feedback from the full objective f_τ makes the source overly informative: performance remains strong on Airfoil Small but falls to the feedback-only baseline on Airfoil Lift, suggesting that overly reliable pretraining feedback can induce over-trust in structurally biased proxies.

Method	Airfoil Small	Airfoil Lift
FICBO (Mixture)	$0.020_{\pm 0.068}$	$3.617_{\pm 2.401}$
Mixture, no bias operators	$0.029_{\pm 0.102}$	$10.386_{\pm 6.063}$
FICBO (Additive)	$0.014_{\pm 0.062}$	$4.295_{\pm 2.841}$
Additive, no bias operators	$0.014_{\pm 0.060}$	$4.999_{\pm 2.494}$
Additive-prior signal variants (no source-model fit)[a]		
g_τ-based feedback (standard)	$0.038_{\pm 0.079}$	$5.206_{\pm 2.975}$
g_τ-based, no catastrophic	$0.032_{\pm 0.078}$	$5.747_{\pm 3.742}$
g_τ-based, no shift	$0.016_{\pm 0.062}$	$5.924_{\pm 3.452}$
g_τ-based, no biases	$0.032_{\pm 0.093}$	$7.026_{\pm 3.542}$
f_τ-based, standard biases	$0.021_{\pm 0.080}$	$24.649_{\pm 5.341}$
πBO-EI	$0.097_{\pm 0.388}$	$7.807_{\pm 2.486}$
πBO-UCB	$0.113_{\pm 0.422}$	$8.167_{\pm 2.834}$
Feedback	$0.020_{\pm 0.079}$	$24.179_{\pm 5.486}$
Random	$12.692_{\pm 7.489}$	$12.692_{\pm 7.489}$

[a] The direct-source branch of Eq. (12) is used.

Table 4: Final cumulative regret (↓) on Airfoil Small and Airfoil Lift. Mean ± std over 100 seeds.

The bias ablations support this conclusion. Removing all bias operators worsens performance on Airfoil Lift, indicating that exposure to distorted sources helps the model learn when to discount feedback. The smaller effect of removing individual operators suggests that the benefit comes from diverse feedback–objective relationships rather than any single distortion type. The bias family thus broadens the range of source reliabilities seen offline, so the model learns to calibrate trust in context instead of assuming feedback is a faithful proxy for the objective.

6 Discussion

We introduced FICBO, an in-context optimization framework for settings with cheap but unreliable auxiliary feedback. It treats feedback as a stochastic information source and pretrains both posterior prediction and query selection under a prior over feedback access, relevance, and distortion. Across synthetic benchmarks and real-world tasks, the resulting optimizer learns to exploit aligned feedback while discounting unreliable proxies from in-context evidence.

Limitations. A central challenge is coverage of the feedback prior: transfer is strongest when the prior spans the main ways in which deployment feedback can agree or disagree with the objective. The feedback-profile embedding in Figure 5 provides one diagnostic for this coverage, while the Branin stress test illustrates a difficult regime: smooth but globally rank-misaligned feedback. Second, we assume feedback is available over the candidate pool, which is natural for cheap simulators or pretrained proxies, but expensive feedback would require an additional acquisition mechanism. Third, the feedback source is fixed during the BO episode and does not update after new objective evaluations, as opposed to changing feedback sources, as when dealing with human feedback. Limiting experiments to low dimensions leaves open the question of whether the approach scales to high-dimensional search spaces, where surrogate modelling and feedback alignment may behave differently. Finally, pretraining requires a horizon distribution, so very different test-time budgets are formally out-of-distribution; in practice, models trained with $T_\tau \in [10, 20]$ were successfully evaluated for $T = 30$.

Perspectives. A natural next step is to combine feedback-aware pretraining with richer objective priors, either problem-specific or broadly diversified in the spirit of tabular foundation models [28]. For instance, human-in-the-loop simulators or cognitive-agent models could provide partially reliable feedback priors [29]. The goal would be to cover not only a wide range of objective functions, but also a wide range of feedback–objective relationships. Another direction is to handle multiple test-time feedback sources, learning which cheap but imperfect signals to combine for each task.

Acknowledgments and Disclosure of Funding

This work was supported by the Research Council of Finland (Flagship programme: Finnish Center for Artificial Intelligence FCAI) and the EU funding ERC ODD-ML 101201120. NB and SK were supported by the Research Council of Finland (RCF-NSF FinBioFAB, grant agreement 365982). SK was also supported by UKRI Turing AI World-Leading Researcher Fellowship (EP/W002973/1). JM was supported by the Research Council of Finland (grant agreement 377145). The authors acknowledge the computational resources provided by the Aalto Science-IT Project from Computer Science IT. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.



References

[1] Di Wu, Natalie Maus, Anupama Jha, Kevin Yang, Benjamin D Wales-McGrath, San Jewell, Anna Tangiyan, Peter Choi, Jacob R Gardner, and Yoseph Barash. Generative modeling for RNA splicing predictions and design. *eLife*, 2025.

[2] Iiris Sundin, Alexey Voronov, Haoping Xiao, Kostas Papadopoulos, Esben Jannik Bjerrum, Markus Heinonen, Atanas Patronov, Samuel Kaski, and Ola Engkvist. Human-in-the-loop assisted de novo molecular design. *Journal of cheminformatics*, 2022.

[3] Qiaohao Liang, Aldair E Gongora, Zekun Ren, Armi Tiihonen, Zhe Liu, Shijing Sun, James R Deneault, Daniil Bash, Flore Mekki-Berrada, Saif A Khan, et al. Benchmarking the performance of bayesian optimization across multiple experimental materials science domains. *npj Computational Materials*, 2021.

[4] Petrus Mikkola, Julien Martinelli, Louis Filstroff, and Samuel Kaski. Multi-fidelity bayesian optimization with unreliable information sources. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, 2023.

[5] Mingzhou Fan, Byung-Jun Yoon, Edward Dougherty, Francis Alexander, Nathan Urban, Raymundo Arróyave, and Xiaoning Qian. Multi-fidelity bayesian optimization with multiple information sources of input-dependent fidelity. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

[6] Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

[7] Carl Hvarfner, Danny Stoll, Artur Souza, Luigi Nardi, Marius Lindauer, and Frank Hutter. πBO: Augmenting acquisition functions with user beliefs for bayesian optimization. In *International Conference on Learning Representations*, 2022.

[8] Masaki Adachi, Brady Planden, David Howey, Michael A. Osborne, Sebastian Orbell, Natalia Ares, Krikamol Muandet, and Siu Lun Chau. Looping in the human: Collaborative and explainable Bayesian optimization. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, 2024.

[9] Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. Pfns4bo: In-context learning for bayesian optimization. In *International Conference on Machine Learning*, 2023.

[10] Rosen Ting-Ying Yu, Cyril Picard, and Faez Ahmed. GIT-BO: High-dimensional bayesian optimization with tabular foundation models. In *The Fourteenth International Conference on Learning Representations*, 2026.

[11] Yu-Heng Hung, Kai-Jie Lin, Yu-Heng Lin, Chien-Yi Wang, Cheng Sun, and Ping-Chun Hsieh. Boformer: Learning to solve multi-objective bayesian optimization via non-markovian rl. *arXiv preprint arXiv:2505.21974*, 2025.

[12] Herilalaina Rakotoarison, Steven Adriaensen, Tom Julian Viering, Carl Hvarfner, Samuel Müller, Frank Hutter, and Eytan Bakshy. α-PFN: Fast entropy search via in-context learning. In *Forty-third International Conference on Machine Learning*, 2026.

[13] Alexandre Maraval, Matthieu Zimmer, Antoine Grosnit, and Haitham Bou Ammar. End-to-end meta-bayesian optimisation with transformer neural processes. *Advances in Neural Information Processing Systems*, 36:11246–11260, 2023.

[14] Xinyu Zhang, Daolang Huang, Samuel Kaski, and Julien Martinelli. PABBO: Preferential amortized black-box optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[15] Xinyu Zhang, Conor Hassan, Julien Martinelli, Daolang Huang, and Samuel Kaski. In-context multi-objective optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

[16] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* MIT Press, 2006.

[17] Artur Souza, Luigi Nardi, Leonardo Oliveira, Kunle Olukotun, Marius Lindauer, and Frank Hutter. Prior-guided bayesian optimization. 2020.

[18] Adam Foster, Desi R Ivanova, Ilyas Malik, and Tom Rainforth. Deep adaptive design: Amortizing sequential bayesian experimental design. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[19] Daolang Huang, Xinyi Wen, Ayush Bharti, Samuel Kaski, and Luigi Acerbi. ALINE: Joint amortization for bayesian inference and active data acquisition. In *Thirty-ninth Conference on Neural Information Processing Systems*, 2025.

[20] Quoc-Anh Hoang Nguyen, Xinyu Zhang, Daolang Huang, Julien Martinelli, and Samuel Kaski. In-context batch black-box optimization via flow matching. *arXiv preprint arXiv:2610.20138*, 2026.

[21] Kevin Swersky, Yulia Rubanova, David Dohan, and Kevin Murphy. Amortized bayesian optimization over discrete spaces. In *Conference on Uncertainty in Artificial Intelligence*, 2020.

[22] Juan Ungredda and Juergen Branke. When to elicit preferences in multi-objective bayesian optimization. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023.

[23] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

[24] Petrus Mikkola. Humans as information sources in Bayesian optimization. 2024.

[25] Yasmine Nahal, Janosch Menke, Julien Martinelli, Markus Heinonen, Mikhail Kabeshov, Jon Paul Janet, Eva Nittinger, Ola Engkvist, and Samuel Kaski. Human-in-the-loop active learning for goal-oriented molecule generation. *Journal of Cheminformatics*, 2024.

[26] Yiming Yao, Fei Liu, Liang Zhao, Xi Lin, Yilu Liu, and Qingfu Zhang. Fomemo: Towards foundation models for expensive multi-objective optimization. *arXiv preprint arXiv:2509.03244*, 2025.

[27] Daolang Huang, Zhuoyue Huang, Conor Hassan, Luigi Acerbi, Samuel Kaski, and Tom Rainforth. Efficient adaptive data acquisition via pretrained belief representations. In *Fortieth Conference on Neural Information Processing Systems*, 2026.

[28] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabiclv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026.

[29] Yifan Zhu, Sammie Katt, and Samuel Kaski. More than irrational: Modeling belief-biased agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[30] Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, 2007.

[31] Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 2012.

[32] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. In *AutoML workshop at ICML*, volume 7, page 5, 2018.

[33] David JJ Toal. Some considerations regarding the use of multi-fidelity kriging in the construction of surrogate models. *Structural and Multidisciplinary Optimization*, 2015.

[34] H Scott Fogler. *Elements of chemical reaction engineering*. Pearson Educacion, 1999.

[35] Octave Levenspiel. *Chemical reaction engineering*. John wiley & sons, 1998.

[36] Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 2021.

[37] Peter Sharpe and R John Hansman. Neuralfoil: An airfoil aerodynamics analysis tool using physics-informed machine learning. *arXiv preprint arXiv:2503.16323*, 2025.

[38] R Priyanka and M Sivapragasam. Multi-fidelity surrogate model-based airfoil optimization at a transitional low reynolds number. *Sādhanā*, 2021.

[39] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

[40] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019.

[41] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, 2019.

[42] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

The appendix is organized as follows:

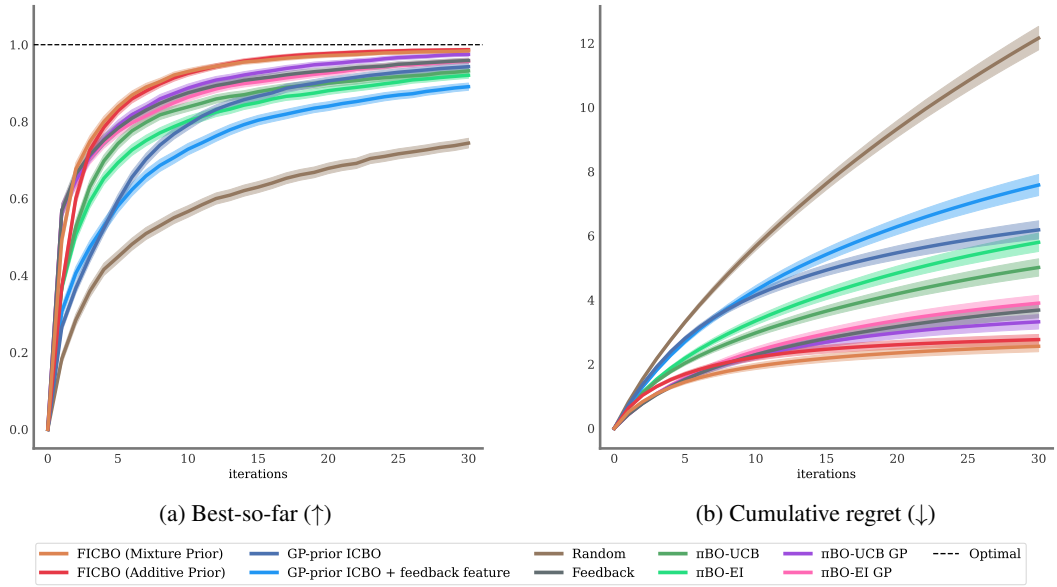
- **Appendix A** presents additional experiments, including classical-GP πBO comparisons and a qualitative policy rollout.
- **Appendix B** details synthetic pretraining task generation, including GP draws, feedback priors, and structured bias operators.
- **Appendix C** provides baseline and benchmark details, including synthetic and real-world tasks and the feedback-profile embedding diagnostic.
- **Appendix D** describes the MDP formulation, training objective, deployment modes, and pretraining algorithm.
- **Appendix E** reports detailed per-benchmark results across feedback configurations.
- **Appendix F** investigates continuous optimization through gradient-based query refinement and compares it with pool-based selection.
- **Appendix G** specifies the model architecture, computational resources, and hyperparameters.

A Additional experiments

A.1 Classical-GP baselines

The main text compares only amortized in-context baselines, so that differences in performance can be attributed to how the feedback channel is integrated. In particular, our πBO-style baselines use the GP-prior ICBO surrogate: the inference head outputs a Gaussian-mixture posterior, from which we extract moments to form EI or UCB scores before applying the feedback reweighting. This isolates the role of feedback integration, but it is not necessarily the strongest possible implementation of πBO.

Ultimately, however, black-box optimization is judged by the quality of the designs it finds, regardless of whether the optimizer is amortized or fit online. For completeness, we also evaluate πBO with a classical GP surrogate (batched GPyTorch exact GP, ARD Matérn-3/2 kernel, 100 Adam steps per BO step, warm-started from the previous step) and closed-form EI and UCB, refit online at every step. This gives a stronger non-amortized comparison: it is slower, but provides a standard GP posterior that is better matched to EI and UCB. Figure S1 shows that, on the marginal expert, this replacement improves the πBO baselines: the GP-surrogate variants (— πBO-UCB GP; — πBO-EI GP) outperform their in-context-surrogate counterparts (— πBO-UCB; — πBO-EI). However, FICBO remains the strongest method in both best-so-far value and cumulative regret, although the gap is reduced. This suggests that source-aware pretraining still provides a benefit over classical feedback reweighting.



(a) Best-so-far (↑)

(b) Cumulative regret (↓)

Figure S1: **Comparison with classical-GP πBO baselines on marginalized benchmark functions.** The main-text πBO baselines use the GP-prior ICBO surrogate. Here, πBO-UCB GP and πBO-EI GP instead use a classical GPyTorch GP surrogate with closed-form EI and UCB, refit online at each step. Mean $\pm$ 95% bootstrap confidence intervals. **GP-based πBO improves over the in-context πBO variants, but FICBO remains best in both best-so-far value and cumulative regret.**

A.2 Rollout Analysis

To provide qualitative insight into the behaviour of the learned policy, Figure S2 shows a single rollout on the Forrester function, using FICBO (mixture prior). At each acquisition step, the model observes the current context in blue and a set of candidate query points, and must select the most promising candidate. The expert feedback signal (second column) provides a biased proxy for the hidden objective, encoding the expert's preference over the query set. In this case, this is the classic low-fidelity variant of the function [30]. The model posterior (third column) reflects the model's current belief about the objective, which sharpens as more points are observed. The policy distribution (fourth column) shows the learned acquisition scores over the query set, which ideally concentrates mass in regions of high objective value. The model initially agrees with the expert to some extent, but as new information arrives, the posterior is updated.

B Generative process of synthetic pretraining dataset

B.1 GP draws

We construct each synthetic BO task based on a procedure first described by [19]:

Input space and dimensionality. We sample the expert input dimension d_{src} and the number of overlapping dimensions o shared between the main model and the expert; the total input dimension is $d_{\text{full}} = d + d_{\text{src}} - o$. The main model observes inputs $\boldsymbol{x} \in \mathbb{R}^d$ (the first d coordinates), while the expert operates over a different and potentially partially overlapping slice of the full d_{full}-dimensional space; the last d_{src} dimensions. A model has a fixed number of dimensions; for our experiments, we trained a variant for 1-, 2-, and 3-dimensional inputs. This decoupling allows us to model realistic settings where the expert has access to features invisible to the main model, and vice versa.

GP hyperparameter prior. All GP components, regardless of prior type, share the same hyperparameter prior: Matérn family with $\nu \in \{3/2, 5/2, \infty\}$ (each with probability $1/3$), length-scales $\mathcal{U}(0.1\sqrt{d}, 2.0\sqrt{d})$, output scale $\mathcal{U}(0.1, 1.0)$, and are isotropic with probability 0.5; in the mixture prior, an isotropic draw gives all real components one shared lengthscale and all decoy components

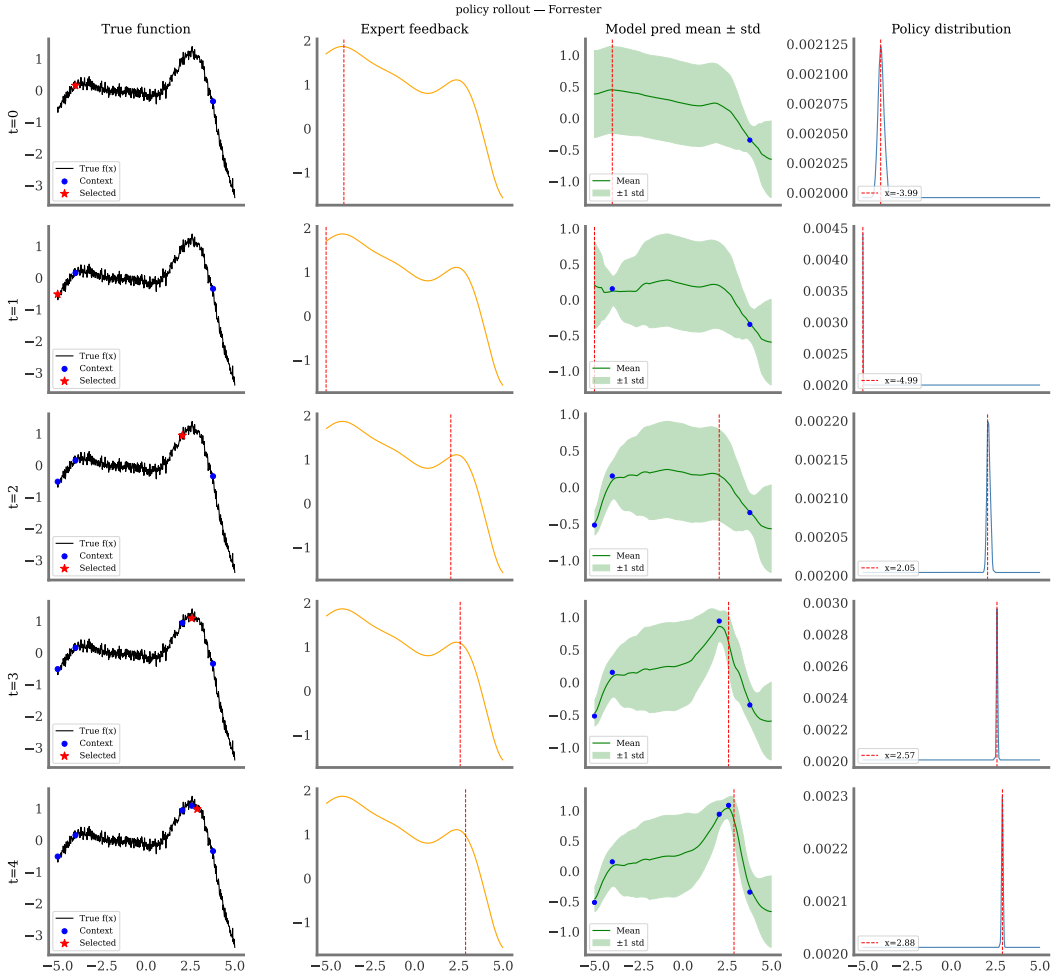


Figure S2: Policy rollout of the amortized policy acquisition function on the Forrester function over $T = 5$ steps. Each row corresponds to one acquisition step. From left to right: (1) the true objective function with context points (blue) and the selected candidate (red star), (2) the expert feedback signal over the query set, (3) the model posterior mean (green) with ± 1 standard deviation (shaded), and (4) the learned policy distribution over candidate points. The red dashed vertical line indicates the selected point across all panels.

another shared lengthscale. The expert feedback signal can be further distorted by the structured bias model described in section B.5. The query, target, and context points are sampled uniformly in $[-5, 5]^{d_{\text{full}}}$; the additional expert points are sampled as described in section B.2. These numbers, as well as sections of the code, are based on [19].

B.2 Additive prior

Function generation. We draw two independent GPs: a main GP over the main-only dimensions and a hidden GP over the expert dimensions. The observed signal is a combination of both:

$$f_\tau = \frac{(1-w)\, f_\tau^{\text{model}} + w\, f_\tau^{\text{src}}}{\sqrt{(1-w)^2 + w^2}}, \tag{S1}$$

where $w_\tau \sim \mathcal{U}(w_{\min}, w_{\max})$ controls the relative contribution of the hidden component, and the denominator normalizes the variance. When the main model has no exclusive dimensions ($d = o$), f_τ^{model} is set to zero and $w = 1$ (only for the 1D model using our hyper-parameters).

The expert's raw signal is g_τ (or optionally f_τ): the output of the hidden GP evaluated at all points. This signal is optionally perturbed by the structured bias model described in section B.5, yielding a

biased feedback signal $\tilde{g}_\tau$. Depending on the expert model, the biased signal is further processed. For the mixture prior, feedback is returned directly without further changes. For the additive prior, we fit a model (amortized GP) to some points generated by the previously constructed function. This is to simulate that the additional source may only have information in some area of the design space. Of these points, 25% are sampled uniformly and 75% around 1-5 random cluster centers (std $\mathcal{U}(0.5, 1.5)$) in the expert input space, which creates an additional bias. Its predictive mean at the context, query, and target points is the feedback signal seen by the main model. The output of the expert model is u. In both cases, the main model only ever observes noisy evaluations of f_τ as the ground truth signal and $\boldsymbol{x}_{1:d}$ as inputs, as well as the final feedback u. The expert's extra observations are never directly available to the main model.

B.3 Mixture prior

As an alternative that moves the source error from feature access to component weighting, we also consider a latent-component mixture prior. It keeps the visibility split: the components are GPs on all d_{full} coordinates, the optimizer sees the first d, and the source signal uses all of them. For each task, we sample a multi-output Gaussian process

$$\boldsymbol{z}_\tau(\boldsymbol{x}) = \big(z_{\tau,1}(\boldsymbol{x}), \dots, z_{\tau,K_\tau}(\boldsymbol{x})\big), \tag{S2}$$

whose components represent latent factors underlying the task. The true objective is defined by a scaled sum over a subset of relevant components,

$$f_\tau(\boldsymbol{x}) = \frac{1}{\sqrt{K_{\text{real}}}} \sum_{k=1}^{K_{\text{real}}} z_{\tau,k}(\boldsymbol{x}) \tag{S3}$$

where $\{1, \dots, K_{\text{real}}\} \subseteq \{1, \dots, K_\tau\}$ indexes the components that actually matter for optimization. Components outside $\{1, \dots, K_{\text{real}}\}$ act as decoys.

The auxiliary source instead forms its latent signal from a possibly different subset S_τ, constructed by masking each component independently. With $m_{\tau,1} = 1$ and $m_{\tau,k} \sim \text{Bernoulli}(p_{\text{src}})$ for $k > 1$, the active set is $S_\tau = \{k : m_{\tau,k} = 1\}$; equivalently, $|S_\tau| - 1 \sim \text{Binomial}(K_\tau - 1,\, p_{\text{src}})$. We draw source weights from a symmetric Dirichlet prior on S_τ,

$$\bar{\boldsymbol{a}}_{\tau,S_\tau} \sim \text{Dirichlet}\big(\alpha_{\text{src}} \mathbf{1}_{|S_\tau|}\big), \tag{S4}$$

and normalize them to unit ℓ_2 norm, like the weights $1/\sqrt{K_{\text{real}}}$ of f_τ,

$$a_{\tau,k} = \frac{\bar{a}_{\tau,k}}{\|\bar{\boldsymbol{a}}_{\tau,S_\tau}\|_2} \mathbb{I}\{k \in S_\tau\}. \tag{S5}$$

The resulting latent source signal is

$$g_\tau(\boldsymbol{x}) = \sum_{k=1}^{K_\tau} a_{\tau,k}\, z_{\tau,k}(\boldsymbol{x}), \tag{S6}$$

and the observed feedback again takes the form

$$u^\tau = \mathcal{B}_\tau[g_\tau]. \tag{S7}$$

This alternative prior treats auxiliary information as a biased combination of latent task factors. Its informativeness depends on the overlap between S_τ and $\{1, \dots, K_{\text{real}}\}$, while $\mathcal{B}_\tau$ captures further distortions. This connects naturally to latent-factor and coregionalization views, where different signals share hidden components but weight them differently [31].

B.4 Combined Prior

We do not force a single choice, since both priors have sensible interpretations. One can train a single model on a 50/50 mixture of both priors; we call this variant FICBO (Combined). As an alternative that does not require training an additional model, the resulting policies can be combined by averaging their softmax probabilities, now called FICBO (Ensemble). Unsurprisingly, the results of the benchmarks are in between what the individual priors achieve as can be seen in table S1, or in more detail in section E.

Table S1: Cumulative regret for FICBO combination variants across marginal and XGBoost expert regimes (lower is better)

Method	Marginal expert		XGBoost expert	
	Mean	Std	Mean	Std
FICBO (Mixture prior)	2.548	3.033	4.329	4.635
FICBO (Ensemble)	2.750	3.176	4.683	4.638
FICBO (Combined) (= train on both priors)	2.858	3.103	4.932	4.605
FICBO (Additive Prior)	3.259	3.319	5.181	4.670

B.5 Description of Biases

In our framework, feedback can be perturbed by a set of structured biases. Each bias can be activated independently and is controlled by one or more parameters drawn from predefined priors. Inactive biases are effectively set to zero. Biases are applied sequentially to the feedback, and the order may matter for some spatially-dependent transformations.

B.5.1 Noise

Description: Experts may make random, unstructured errors in their feedback across the input space. Homoscedastic noise models this uniform uncertainty.

Implementation: Add Gaussian noise with constant variance to all feedback points.

Parameters and Priors:

- σ: Standard deviation of the noise, drawn from a uniform prior over a specified lower and upper bound.

Knowledge Center Extension
Description: Instead of the overly simplistic homoscedastic noise alone, we designed a second component. Experts and models tend to be reliable near regions they are familiar with, and increasingly uncertain further away. This bias models a single center of competence: feedback remains accurate close to this center, while noise grows with distance from it.

Implementation: Select one random point in the input space as the knowledge center. Compute distances from all feedback points to this center, normalize by the maximum distance, and add Gaussian noise whose standard deviation scales linearly with this normalized distance.

Parameters and Priors:

- `noise_radiation_scale`: Maximum noise level at the furthest point from the center, drawn from a uniform prior within specified bounds.

For the Mixture prior, the noise we apply throughout training is the combination of both mechanisms and is applied 100% of the time. For the Additive prior we leave the knowledge center off: its source already only knows part of the space, since it sees a subset of the features and is fit to source-private points that are partly clustered (section B.2). This produces a similar effect, accurate where the source has data and uncertain elsewhere, so a knowledge center would be redundant.

B.5.2 Constant Shift

Description: Experts may exhibit a systematic over- or under-estimation of the true function across the entire input space. A constant shift models this global bias in feedback. **Implementation:** Add a constant offset to all feedback points. **Parameters and Priors:**

- $\delta \sim \mathcal{U}(\delta_{\min}, \delta_{\max})$: The shift applied to all feedback points.

B.5.3 Additive GP Bias

Description: By adding a sampled GP with a smooth kernel to the signal, we create correlated noise. It introduces smooth, structured deviations resembling latent functions that the expert might have learned but misrepresented.

Implementation: Sample a Gaussian process with RBF kernel using lengthscale and scale parameters. Add the GP output to feedback.

Parameters and Priors:

- lengthscale: Controls smoothness of the GP, sampled uniformly from prior bounds.
- scale: Magnitude of GP deviations, drawn uniformly from prior bounds. (determines the shape of the fluctuations before normalization. A larger scale will give more contrast between points in the GP sample.)
- max_std: Maximum standard deviation for normalization, drawn uniformly from prior bounds. (caps the overall magnitude)

B.5.4 Local Distortion

Description: Models localized deviations in expert feedback due to over/underestimations in certain regions.

Implementation: Randomly select n_{centers} points, assign a distortion magnitude, and decay this effect with distance from each center.

Parameters and Priors:

- n_{centers}: Number of distortion centers, sampled uniformly from a specified integer range.
- mag: Magnitude of local distortion, drawn from a uniform prior, with a random sign per center.
- decay: Spatial decay factor, sampled uniformly from prior bounds.

B.5.5 Catastrophic Failure

Description: Models rare but severe expert failures where the feedback has no meaningful relationship to the true function. For instance, due to a fundamental misunderstanding of the task or a complete breakdown in the expert's model.

Implementation: Replace all feedback values with an independently sampled RBF Gaussian process (lengthscale $\mathcal{U}(0.1, 2)$), rescaled to the standard deviation of the signal it replaces.

Parameters and Priors:

- No continuous parameters; the bias is either fully active or inactive, indicated by a binary flag.

B.5.6 Application order and activation

Biases are applied sequentially to the feedback signal, and each bias is activated independently with its own probability for each training batch; all tasks in a batch share the same active biases. The application order is: homoscedastic noise, catastrophic failure, additive GP bias, local distortion, knowledge-center noise, and finally constant shift. When a bias is inactive, it contributes no perturbation. For more details, we refer to table S5.

C Experiment Details

C.1 Baselines

All baselines are evaluated in the same pool-based setting as our method: at each step t, a candidate is selected from a fixed query pool Q_t^τ ($|Q_0^\tau| = 300$ unless stated otherwise), observed, and added to

the context. No gradient-based acquisition optimization is performed; all methods score each pool candidate and select the argmax. For a fair comparison, Q_t^τ is shared between all baselines.

Surrogate models. Methods that score candidates with an acquisition function use one of three surrogates:

- **GP-prior ICBO**: the same pretrained transformer model used by our method, except for the feedback encoder, producing a GMM predictive posterior over each query point. The model has been trained on generic GP data following [19]. When run on its own, GP-prior ICBO selects candidates with its policy head.
- **FICBO inference head**: the GMM predictive posterior q_ϕ of the trained FICBO model (Mixture Prior), which also conditions on the feedback (semi-amortized mode, section D).
- **Classical GP**: a batched GPyTorch exact GP with ARD Matérn-3/2 kernel, fit with 100 Adam steps per BO step, warm-started from the previous step.

For both GMM posteriors, the weighted mean and variance are extracted as $\mu(\boldsymbol{x}) = \sum_\ell \omega_\ell \mu_\ell(\boldsymbol{x})$ and $\sigma^2(\boldsymbol{x}) = \sum_\ell \omega_\ell \sigma_\ell^2(\boldsymbol{x})$. The main-text πBO baselines use the GP-prior ICBO surrogate. This is done to isolate the effect of the feedback integration strategy from the surrogate quality.

Acquisition functions. We use the following closed-form acquisition functions, which can be combined with any surrogate:

- **UCB**: $\alpha(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \kappa\sigma(\boldsymbol{x})$. κ has been set to 1.
- **EI**: $\alpha(\boldsymbol{x}) = (\mu(\boldsymbol{x}) - y^+)\Xi(z) + \sigma(\boldsymbol{x})\xi(z)$ where $z = (\mu(\boldsymbol{x}) - y^+)/\sigma(\boldsymbol{x})$, Ξ and ξ the standard normal CDF and PDF, and y^+ the best observed value.

Semi-amortized FICBO. One can also replace the policy head of FICBO with its inference head and an acquisition function (section D). We use UCB and EI, reweighted with the πBO feedback method, described below, and call these variants FICBO-UCB and FICBO-EI. table S2 shows the results are comparable to the end-to-end policy with the same prior. For a more detailed comparison, see section E.

Table S2: Cumulative regret across marginal and XGBoost expert regimes (lower is better), comparing end-to-end and semi-amortized FICBO with the baselines.

Method	Marginal expert		XGBoost expert	
	Mean	Std	Mean	Std
FICBO (Mixture Prior), end-to-end policy	2.548	3.033	4.329	4.635
FICBO-UCB (Mixture Prior), semi-amortized	2.657	3.204	4.897	4.613
FICBO-EI (Mixture Prior), semi-amortized	2.771	3.388	4.880	4.688
FICBO (Additive Prior), end-to-end policy	3.259	3.319	5.181	4.670
Feedback	3.683	4.392	7.774	6.178
πBO-UCB	4.966	4.845	6.199	5.355
πBO-EI	5.466	4.988	6.477	5.513
GP-prior ICBO + feedback feature	6.000	4.820	7.129	5.524
GP-prior ICBO	8.462	5.939	8.462	5.939
Random	12.158	6.753	12.879	6.836

Incorporating expert feedback. For baselines that augment the acquisition with feedback (πBO-style), the feedback signal scores are first normalized before combination. Concretely, the feedback is converted to a policy via a softmax function: $\tilde{u}(\boldsymbol{x}) = \text{softmax}(u(\boldsymbol{x}_1), \ldots, u(\boldsymbol{x}_{|Q_t|}))$ The adjusted score is then:

$$\alpha_{\text{adj}}(\boldsymbol{x}) = \alpha(\boldsymbol{x}) \cdot \tilde{u}(\boldsymbol{x})^{\beta/(t+1)} \tag{S8}$$

where the decaying exponent $\beta/(t+1)$ follows Hvarfner et al. [7], ensuring the feedback influence vanishes as more objective observations accumulate and the surrogate becomes more reliable. We set $\beta = 2.0$ following the original paper.

GP-prior ICBO with Feedback Feature. The concatenation baseline appends the feedback value $u(\boldsymbol{x})$ directly to the input features before passing to the amortized model. The model receives $(\boldsymbol{x}, u(\boldsymbol{x}))$ as input rather than $\boldsymbol{x}$ alone. This tests whether the model can implicitly learn to use feedback when it is treated as just another input feature, without any explicit feedback-aware architecture.

Classical-GP controls. GP-UCB uses the classical GP surrogate with UCB and no feedback. GP-UCB + feedback feature fits the same GP on $(\boldsymbol{x}, u(\boldsymbol{x}))$ instead of $\boldsymbol{x}$. They are the classical-GP counterparts of GP-prior ICBO and GP-prior ICBO + feedback feature, and show how much of the performance comes from the surrogate and how much from the feedback.

C.1.1 Additional Baselines

Notation. For the three additional baselines below we treat the pool-wide expert feedback as a second *information source* $u(\boldsymbol{x})$ that is informative about, but distinct from, the sparse real objective $f(\boldsymbol{x})$. We reuse $\mu_i(\boldsymbol{x}), \sigma_i^2(\boldsymbol{x})$ for the mean/variance of the i-th component model, Q_t^τ for the pool at step t, and index sources with $j \in \{f, u\}$ (real objective and feedback, respectively).

iMFBO [5]. Multi-fidelity BO with multiple information sources treats every source as a noisy view of one shared latent objective. Our feedback is not on the scale of the objective, thus we add an affine calibration per environment:

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + \sigma_f\,\varepsilon, \qquad u(\boldsymbol{x}) = a\,f(\boldsymbol{x}) + b + \sigma_u(\boldsymbol{x})\,\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0,1),\ a > 0, \tag{S9}$$

so that the real objective and the feedback are jointly modeled by a single GP over f whose noise diagonal is input- and source-dependent:

$$\hat{K} = SKS + \mathrm{diag}\big(\sigma_{j_i}(\boldsymbol{x}_i)^2\big), \qquad S = \mathrm{diag}(s_i),\ s_i = 1 \text{ for objective rows},\ s_i = a \text{ for feedback rows}. \tag{S10}$$

The objective noise σ_f is a learned constant. The fidelity function $\sigma_u(\boldsymbol{x})$, i.e. how much noise the feedback row contributes at $\boldsymbol{x}$, is learned jointly with a, b and the GP hyperparameters by MAP. iMFBO parameterizes $\sigma_u(\boldsymbol{x})$ with a small MLP. iMFBO-GP uses one free log-noise value per feedback row, with a GP prior over these values (the MAP estimate of Fan et al. [5], Eq. 12). Candidates are scored with NVUCB, $\alpha(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \kappa\,\gamma(\boldsymbol{x})\,\sigma(\boldsymbol{x})$ with $\gamma(\boldsymbol{x}) = \sigma(\boldsymbol{x}) / \sqrt{\sigma(\boldsymbol{x})^2 + \sigma_f^2}$, which decreases the exploration bonus where a query is expected to give little information. The iMFBO-UCB and iMFBO-GP-UCB variants use plain UCB on the same model. In all variants, $\kappa = 1$.

RGPE [32]. The rank-weighted GP ensemble treats the dense pool-wide feedback as a *related* (base) task and the sparse real-objective observations as the *target* task, and combines a GP fit on each: GP_u trained on all $(\boldsymbol{x}, u(\boldsymbol{x}))$ pairs in Q_t^τ, and GP_f trained on the observed $(\boldsymbol{x}, f(\boldsymbol{x}))$ pairs so far. The ensemble posterior at a candidate $\boldsymbol{x}$ is a weighted combination

$$\mu_{\mathrm{RGPE}}(\boldsymbol{x}) = \sum_{i \in \{u,f\}} w_i\,\mu_i(\boldsymbol{x}), \qquad \sigma^2_{\mathrm{RGPE}}(\boldsymbol{x}) = \sum_{i \in \{u,f\}} w_i^2\,\sigma_i^2(\boldsymbol{x}), \tag{S11}$$

where the weights w_i are estimated from a leave-one-out ranking-loss comparison between each base model's predicted pairwise ordering on the target task and the true observed ordering, so that GP_u is down-weighted automatically as it disagrees with the accumulating real-objective observations. We use $\mu_{\mathrm{RGPE}}, \sigma_{\mathrm{RGPE}}$ in place of μ, σ in the UCB/EI acquisition functions defined above.

Multi-Task GP (ICM). As a transfer-learning baseline, we fit a single multi-task GP jointly over the real-objective task and the feedback task using a batched re-implementation of BoTorch's `MultiTaskGP` with one added free outputscale, with an intrinsic co-regionalization (ICM) kernel

$$K\big((\boldsymbol{x}, j), (\boldsymbol{x}', j')\big) = K_{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{x}') \cdot B_{j,j'}, \qquad j, j' \in \{f, u\}, \tag{S12}$$

where $K_{\boldsymbol{x}}$ is a scaled RBF kernel with BoTorch's dimension-scaled lengthscale prior and $B \in \mathbb{R}^{2\times 2}$ is a learned full-rank, non-negative task covariance matrix coupling the real objective and the feedback. All available feedback in Q_t^τ and the real-objective observations collected so far are used to fit B

and the GP hyperparameters jointly, refit at every step t with 30 Adam steps, warm-started from the hyperparameters of the previous step. Candidates are scored via UCB on the task-f posterior slice, $\alpha(\boldsymbol{x}) = \mu_f(\boldsymbol{x}) + \kappa\,\sigma_f(\boldsymbol{x})$, with $\kappa = 1$ as above.

All mentioned baselines get outperformed by FICBO and are considerably more computationally expensive. RGPE and iMFBO-GP follow (iMFBO-GP is better on the marginal expert, RGPE on the XGBoost expert), then iMFBO and the multi-task GP (table S3). Detailed results can be found in section E.

Table S3: Final cumulative regret across marginal and XGBoost expert regimes (lower is better) for additional baselines

Method	Marginal		XGBoost	
	Mean	Std	Mean	Std
FICBO (Mixture Prior)	2.548	3.033	4.329	4.635
FICBO (Additive Prior)	3.259	3.319	5.181	4.670
Feedback	3.683	4.392	7.774	6.178
RGPE-UCB	5.743	5.354	6.849	5.670
GP-prior ICBO + feedback feature	6.000	4.820	7.129	5.524
iMFBO-NVUCB	7.203	5.894	8.628	6.142
iMFBO-GP-NVUCB	4.840	4.414	8.200	6.153
iMFBO-UCB	7.452	5.998	8.767	6.115
iMFBO-GP-UCB	5.090	4.461	8.330	6.193
MTGP-UCB (ICM)	8.007	6.026	9.104	6.373
GP-prior ICBO	8.462	5.939	8.462	5.939
Random	12.158	6.753	12.879	6.836

C.2 Benchmark functions

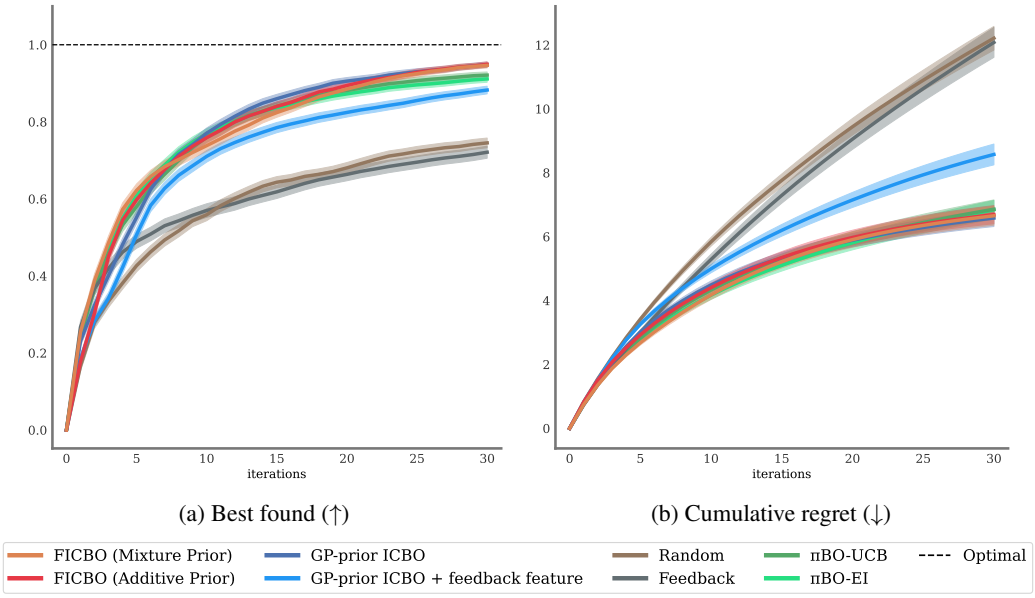
Sampling. We sample n input points $\boldsymbol{x} \in \mathbb{R}^{d_{\text{full}}}$ uniformly in the normalized space $[-s, s]^{d_{\text{full}}}$, where s is a fixed design scale (set to 5 to match the training distribution used). Each point is mapped to the natural domain of the chosen function before evaluation, and independent Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is added to the output. The dataset is split into context, query, and target sets.

Expert feedback. For functions with a registered low-fidelity variant, the expert signal u is the output of that variant evaluated at the same inputs, without further noise. For functions without a low-fidelity variant, u is the Monte Carlo marginal of f over the dimensions not observed by the expert (the expert sees only the last dimension, $d_{\text{src}} = 1$),

$$u(\boldsymbol{x}_{\text{src}}) \approx \frac{1}{L_{\text{MC}}} \sum_{l=1}^{L_{\text{MC}}} f\left(\boldsymbol{x}_{\text{model}}^{(l)}, \boldsymbol{x}_{\text{src}}\right), \quad \boldsymbol{x}_{\text{model}}^{(l)} \sim \mathcal{U}([-s, s]^{d_{\text{full}} - d_{\text{src}}}), \tag{S13}$$

with $L_{\text{MC}} = 2000$ samples over the non-expert dimensions (the point's own coordinates plus 1999 uniform draws shared across points). This is to synthetically make the feedback independent of the dimensions that the expert has no knowledge of. The main model observes noisy evaluations of f, u, and the main features (similar to the training procedure section B) only; the expert's inputs are never directly available.

XGBoost bias To simulate a systematically biased expert, we fit a shallow gradient-boosted tree (XGBoost, 10 estimators, max depth 3, learning rate 0.15, subsample and column sample 0.8) to the noise-free objective values at available context and expert-only points (50 uniformly sampled points), using only the d_{src} dimensions visible to the expert. The model is trained once in the beginning, and its predictions serve as u at all context, query, and target locations. Because the surrogate is deliberately under-specified (low capacity, partial inputs), it produces a *biased* but correlated signal, mimicking a domain expert whose knowledge is both limited and biased (via the limited model class).



(a) Best found (↑)

(b) Cumulative regret (↓)

Figure S3: **Marginalized Feedback with Fake peak. Left:** normalized best value found. **Right:** normalized cumulative regret. Mean ± 95% bootstrap confidence intervals.

Fake peak To test whether the model blindly follows the expert, we corrupt the expert signal with a single fake optimum. Per task instance we sample one center $\mathbf{c}$ uniformly in the expert-visible dimensions, $\mathbf{c} \sim \mathcal{U}([-s, s]^{d_{\text{src}}})$, and add an exponential bump to the marginalized feedback,

$$\tilde{u}(\mathbf{x}_{\text{src}}) = u(\mathbf{x}_{\text{src}}) + 1.1\, y^\star \exp\big(-\lambda \,\|\mathbf{x}_{\text{src}} - \mathbf{c}\|_2\big), \tag{S14}$$

where $y^\star$ is, per task instance, the largest observed (noisy, normalized) objective value over the sampled context, query, and target points, and $\lambda = 0.5$ is fixed. The bump height at $\mathbf{c}$ is $1.1\, y^\star$, so the expert's apparent best region is usually well above the real optimum. Since $\mathbf{c}$ depends only on the expert dimensions and is independent of f, the peak carries no information about the true optimum. A model that trusts the feedback uncritically will be pulled to $\mathbf{c}$, whereas one that has learned to cross-check u against observed f_τ values should discount it. Results are shown in fig. S3. In this example FICBO and πBO variants have very similar results, with FICBO reaching slightly better final solutions.

C.2.1 Function descriptions

The following functions are included in the experiments. All functions below are shown in their standard minimisation form. Every objective is then standardized with 10,000 uniform samples. In the implementation, all are negated so that larger values are better. The functions are commonly used for BO, BED, or AL benchmarking [19].

Ackley ($d \geq 1$). A multimodal function with a single global optimum surrounded by a nearly flat outer region and many local optima,

$$f(\boldsymbol{x}) = -20 \exp\left(-0.2\sqrt{\tfrac{1}{d}\textstyle\sum_i x_i^2}\right) - \exp\big(\tfrac{1}{d}\textstyle\sum_i \cos 2\pi x_i\big) + 20 + e, \tag{S15}$$

defined over $[-5, 5]^d$.

Rastrigin ($d \geq 1$). A highly multimodal function with a large number of regularly spaced local optima,

$$f(\boldsymbol{x}) = 10d + \sum_{i=1}^{d} \left(x_i^2 - 10\cos 2\pi x_i\right), \tag{S16}$$

defined over $[-5.12, 5.12]^d$.

Levy ($d \geq 1$). A multimodal function defined via the transformation $w_i = 1 + (x_i - 1)/4$,

$$f(\boldsymbol{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1} (w_i - 1)^2 \left[1 + 10 \sin^2(\pi w_i + 1)\right] + (w_d - 1)^2 \left[1 + \sin^2(2\pi w_d)\right], \quad \text{(S17)}$$

over $[-10, 10]^d$.

Sphere ($d \geq 1$). The simplest unimodal baseline $f(\boldsymbol{x}) = \sum_i x_i^2$, over $[-5.12, 5.12]^d$.

Schwefel ($d \geq 1$). A deceptive function whose global optimum is geometrically distant from the next-best local optima,

$$f(\boldsymbol{x}) = 418.9829\, d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{|x_i|}\right), \quad \text{(S18)}$$

over $[-500, 500]^d$.

Rosenbrock ($d \geq 2$). A classic unimodal-but-narrow-valley function whose global minimum lies inside a long, curved, flat valley, making convergence hard even though the function is unimodal,

$$f(\boldsymbol{x}) = \frac{1}{3}\left[\log\left(1 + \sum_{i=1}^{d-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)\right) - 3\right], \quad \text{(S19)}$$

defined over $[-2, 2]^d$.[2]

Hartmann ($d = 3$). A smooth multimodal function with four local minima, of which one is global, commonly used as a moderate-dimensional BO benchmark,

$$f(\boldsymbol{x}) = -\frac{1}{4} \sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right), \quad \text{(S20)}$$

with $\boldsymbol{\alpha} = (1.0, 1.2, 3.0, 3.2)$ and coefficient matrices

$$A = \begin{pmatrix} 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \\ 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \end{pmatrix}, \qquad P = \begin{pmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}, \quad \text{(S21)}$$

defined over $[0, 1]^3$, with global minimum $f(\boldsymbol{x}^*) \approx -0.966$ at $\boldsymbol{x}^* \approx (0.1146, 0.5556, 0.8526)$.[3]

C.3 Branin multi-fidelity function ($d = 2$)

A standard two-dimensional test function; with the input mapping below, two of its three global minima lie in the domain.

$$f_h(\boldsymbol{x}) = \left(x_2' - \frac{5.1}{4\pi^2} x_1'^2 + \frac{5}{\pi} x_1' - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1') + 10 \quad \text{(S22)}$$

where $x_i' = 15x_i$, over $[0, 1]^2$. The objective is standardized to zero mean and unit variance over the domain (estimated from 10,000 uniform samples). We use the correlation-adjustable low-fidelity variant of [33]:

$$f_l(\boldsymbol{x}) = f_h(\boldsymbol{x}) - (\rho + 0.5)\left(x_2' - \frac{5.1}{4\pi^2} x_1'^2 + \frac{5}{\pi} x_1' - 6\right)^2 \quad \text{(S23)}$$

[2] We apply a $\log(1 + \cdot)$ compression and affine rescaling to the raw Rosenbrock sum for numerical stability and to bring the output range in line with the other benchmark functions, since the raw sum otherwise grows extremely large in magnitude.

[3] We use a $1/4$-scaled version of the canonical Hartmann-3 function (whose unscaled global minimum is ≈ -3.863) to keep its output range comparable to the other benchmark functions. As with the other functions in this list, this minimization form is negated in the implementation to produce the maximization objective used in experiments.

where $\rho \in [0, 1]$ controls the HF–LF (High Fidelity–Low Fidelity) correlation: the squared term enters f_l with weight $0.5 - \rho$; it is halved at $\rho = 0$, removed at $\rho = 0.5$ (then f_l depends on x_1 only), and inverted for $\rho > 0.5$, which moves the low-fidelity optimum away from that of f_h. The feedback is standardized with mean 0.4484 and std 13.52 (the values for $\rho = 0.75$) for every ρ.

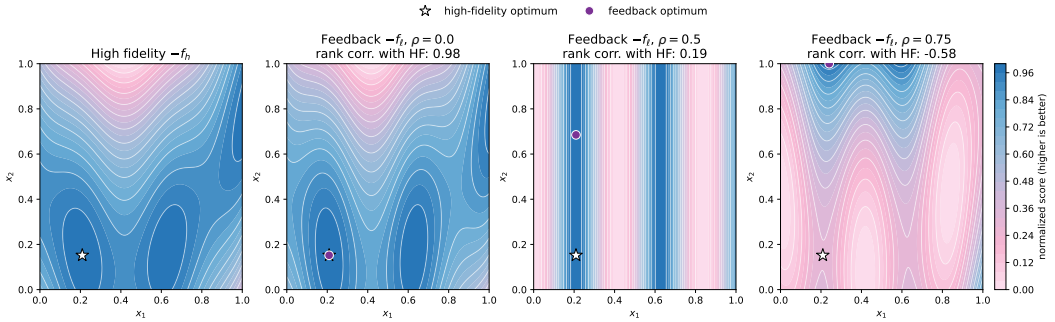


Figure S4: **Correlation-adjustable Branin feedback surfaces.** We plot normalized maximization scores for the high-fidelity objective $-f_h$ and the low-fidelity feedback $-f_l$. White stars mark the high-fidelity optimum, and purple circles mark the feedback optimum. As the discrepancy parameter ρ increases, the feedback surface becomes less rank-aligned with the true objective, as measured by Spearman rank correlation.

C.4 Sequential Reaction with Auxiliary Signal

We consider a standard sequential reaction system:

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C,$$

a classical setting in chemical reaction engineering for studying the trade-off between conversion and selectivity (e.g., [34, 35]). The objective is the concentration of the intermediate B, which we denote as $[B](t_{\mathrm{res}})$, as a function of residence time t_{res} and temperature T_{rx}. The reaction follows first-order kinetics with Arrhenius temperature dependence:

$$k_1(T_{\mathrm{rx}}) = A_1 \exp\left(-\frac{E_{a1}}{RT_{\mathrm{rx}}}\right), \tag{S24}$$

$$k_2(T_{\mathrm{rx}}) = A_2 \exp\left(-\frac{E_{a2}}{RT_{\mathrm{rx}}}\right), \tag{S25}$$

where A_i are pre-exponential factors, E_{ai} activation energies, and R the universal gas constant. We use $A_1 = 10^{10}$, $E_{a1} = 60\,\mathrm{kJ\,mol^{-1}}$, $A_2 = 10^{13}$, $E_{a2} = 80\,\mathrm{kJ\,mol^{-1}}$, $R = 8.314\,\mathrm{J\,mol^{-1}\,K^{-1}}$, $[A]_0 = 1$, $T_{\mathrm{rx}} \in [300, 600]\,\mathrm{K}$ and $t_{\mathrm{res}} \in [0.01, 2]$.

The concentration of B can be obtained analytically as:

$$[B](t_{\mathrm{res}}) = [A]_0 \frac{k_1}{k_2 - k_1}\left(e^{-k_1 t_{\mathrm{res}}} - e^{-k_2 t_{\mathrm{res}}}\right), \tag{S26}$$

which is physically bounded in $[0, 1]$ when normalized by the initial reactant concentration $[A]_0$.

In addition, we use an auxiliary feedback signal corresponding to the conversion of A:

$$\text{Conversion} = 1 - e^{-k_1 t_{\mathrm{res}}}, \tag{S27}$$

which is a standard, easily measured quantity in chemical reaction engineering. In sequential reactions, however, maximizing conversion is not aligned with maximizing $[B]$, since higher conversion drives the system toward the final product C. This induces a systematic mismatch between the proxy signal and the true objective. While Bayesian optimization has been applied to reaction yield optimization [36], the use of conversion as a structured auxiliary signal in this setting and its systematic bias relative to the true objective has not been explicitly studied.

Objective and auxiliary signal. Let $\boldsymbol{x} = (T_{\mathrm{rx}}, t_{\mathrm{res}})$ denote temperature and residence time. The expensive objective is the yield of the intermediate product B,

$$f(\boldsymbol{x}) = [B](t_{\mathrm{res}}, T_{\mathrm{rx}}) = [A]_0 \frac{k_1(T_{\mathrm{rx}})}{k_2(T_{\mathrm{rx}}) - k_1(T_{\mathrm{rx}})} \left(e^{-k_1(T_{\mathrm{rx}})t_{\mathrm{res}}} - e^{-k_2(T_{\mathrm{rx}})t_{\mathrm{res}}} \right), \tag{S28}$$

whereas the auxiliary feedback is the conversion of reactant A,

$$u(\boldsymbol{x}) = 1 - e^{-k_1(T_{\mathrm{rx}})t_{\mathrm{res}}}. \tag{S29}$$

Both f and u are clipped to $[0, 1]$. The objective is standardized to zero mean and unit variance over the domain, while the conversion feedback is used unscaled and without noise. This proxy is cheap to measure and operationally meaningful, but only partially aligned with the objective: consumed A first forms the desired intermediate B, which then reacts further to the undesired by-product C. Consequently, $u(\boldsymbol{x})$ is monotone in t_{res} for fixed T_{rx}, while $f(\boldsymbol{x})$ typically attains an interior optimum because B is eventually depleted into C.

C.5 Aerodynamics

We consider the problem of maximizing the lift-to-drag ratio C_L/C_D of a NACA 4-digit airfoil (maximum camber at 40% chord; thickness and camber rounded to whole percent) as a function of angle of attack $\alpha \in [-2, 15]$, thickness $t \in [0.08, 0.20]$, and camber $c \in [0.00, 0.06]$, a standard aerodynamic shape optimization setting [30].

The true objective is evaluated using NeuralFoil [37], a neural network surrogate trained on approximately one million XFOIL runs that captures viscous boundary-layer effects, including flow separation and stall. At moderate α, C_D remains low in a laminar drag bucket; as α increases toward stall, boundary-layer separation causes C_D to grow rapidly, so C_L/C_D peaks at an interior point ($\alpha^* \approx 5$–7) before declining.

As auxiliary feedback, we consider two proxy variants. The first is structurally biased: it uses lift alone and therefore ignores the drag penalty that makes the true lift-to-drag ratio peak before stall. The second is a lower-accuracy neural surrogate for the same lift-to-drag objective, and therefore serves as a noisy but well-aligned control.

Misleading Feedback Variant (Airfoil Lift) uses C_L alone (from NeuralFoil) as the feedback signal. Since C_L is the numerator of the true objective and increases monotonically with α until hard stall, an agent relying on it will systematically overshoot toward higher angles of attack, ignoring the drag penalty entirely.

Aligned Feedback Variant (Airfoil Small) feedback is generated by a smaller NeuralFoil model (model size `small` instead of `large`), computing C_L/C_D from the same viscous surrogate but at reduced accuracy. Unlike the misleading variant above, this proxy shares the same physical structure as the truth, so the optimum location should be approximately the same as the ground truth. This variant therefore represents *noisy* feedback rather than *structurally biased* feedback, and serves as a control.

Pairing a cheap and an accurate solver as low- and high-fidelity levels is common practice in aerodynamic optimization, for example inviscid panel codes with viscous RANS solvers [30, 38]. Here, both levels are viscous NeuralFoil models of different size. Airfoil Lift has a wrong optimum because it ignores the drag rise that causes C_L/C_D to turn over. Meanwhile, Airfoil Small uses a smaller model, which leads to slightly worse accuracy globally.

Objective and auxiliary signal Let $\boldsymbol{x} = (\alpha, t, c)$ denote angle of attack, thickness, and camber. The expensive objective is the viscous lift-to-drag ratio

$$f(\boldsymbol{x}) = \frac{C_L^{\mathrm{NF,large}}(\boldsymbol{x})}{C_D^{\mathrm{NF,large}}(\boldsymbol{x})}, \tag{S30}$$

where $(C_L^{\mathrm{NF,large}}, C_D^{\mathrm{NF,large}})$ are given by the large NeuralFoil model. The auxiliary feedback is instead provided by a cheaper or structurally simplified proxy,

$$u(\boldsymbol{x}) \in \{u_A(\boldsymbol{x}), u_B(\boldsymbol{x})\}, \tag{S31}$$

with

$$u_A(\boldsymbol{x}) = C_L^{\mathrm{NF,large}}(\boldsymbol{x}), \tag{S32}$$

$$u_B(\boldsymbol{x}) = \frac{C_L^{\mathrm{NF,small}}(\boldsymbol{x})}{C_D^{\mathrm{NF,small}}(\boldsymbol{x})}. \tag{S33}$$

The key mismatch for u_A is that $f(\boldsymbol{x})$ has an interior optimum due to drag growth and stall, whereas lift alone tends to favor larger angles of attack. In contrast, u_B shares the same lift-to-drag structure as f but is evaluated with a smaller NeuralFoil model, making it a cheaper, noisier proxy rather than a structurally biased one.

We use NeuralFoil [37], a physics-informed machine learning tool for airfoil aerodynamics analysis, available at `https://github.com/peterdsharpe/NeuralFoil` under the MIT License.

C.6 Neural Architecture Search (NAS)

We work on the problem of selecting the optimal cell architecture in NAS-Bench-201 [39], a tabular benchmark that enumerates all neural architectures definable on a fixed directed acyclic graph with six edges, each assigned one of five candidate operations: `none`, `skip_connect`, `nor_conv_1x1`, `nor_conv_3x3`, and `avg_pool_3x3`. We fix three edges to canonical values (edges 0, 3, 5 to `nor_conv_3x3`, `skip_connect`, and `nor_conv_3x3`, respectively, a well-performing choice) to reduce the dimension, and treat the remaining three free edges as the design variables, which results in a discrete search space of $5^3 = 125$ architectures, resulting in a reduced query set size of 123, as well as 1 target and initial context point.

Objective and auxiliary signal. Let $\boldsymbol{x} = (e_1, e_2, e_4) \in \{0, \ldots, 4\}^3$ index the operation assigned to each free edge. The expensive objective is the test accuracy on ImageNet16-120 [39], a downsampled variant of ImageNet that is substantially harder than CIFAR-scale tasks and requires architectures with higher representational capacity:

$$f(\boldsymbol{x}) = \mathrm{Acc}_{\text{ImageNet16-120}}(\boldsymbol{x}). \tag{S34}$$

The auxiliary feedback is the test accuracy on CIFAR-10, which is cheaper to evaluate and widely used as a transferability signal in NAS [40, 41]:

$$u(\boldsymbol{x}) = \mathrm{Acc}_{\text{CIFAR-10}}(\boldsymbol{x}). \tag{S35}$$

Both quantities are pre-computed and available via exact lookup in the benchmark table, so no training is performed at evaluation time.

The two signals are positively correlated as a strong architecture on CIFAR-10 tends to perform well on ImageNet16-120, but the proxy optimum and the true optimum are different. CIFAR-10 is a low-complexity task on which even lightweight convolutions saturate performance; ImageNet16-120 penalizes architectures that lack sufficient capacity or spatial resolution handling, so the rankings differ non-trivially between the two datasets, including at the top [40].

We use NAS-Bench-201 [39], available at `https://github.com/D-X-Y/NAS-Bench-201`, under the MIT License.

C.7 Task Embedding via Feedback Error Profiles

To explain the relationship between ground-truth labels and feedback signals across different task configurations, we represent each task instance as a vector that summarizes the distribution of feedback errors.

Task vector construction. Given a batch of query points with ground-truth labels $\mathbf{y} \in \mathbb{R}^N$ and corresponding feedback signals $\mathbf{u} \in \mathbb{R}^N$, we define the pointwise error as

$$d_i = u_i - y_i, \qquad i = 1, \ldots, N. \tag{S36}$$

To capture how errors vary across the output space, we sort the points by either y_i or u_i and partition them into $B = 20$ equal-sized bins. For each bin b, we calculate both the signed mean error (capturing

directional bias) and the mean absolute error (capturing the magnitude of the errors regardless of direction). This results in four statistics per bin:

$$\mu_b^{(y)} = \frac{1}{|b|}\sum_{i\in b} d_i, \qquad \mu_b^{(u)} = \frac{1}{|b|}\sum_{i\in b} d_i, \tag{S37}$$

$$\bar{\mu}_b^{(y)} = \frac{1}{|b|}\sum_{i\in b} |d_i|, \qquad \bar{\mu}_b^{(u)} = \frac{1}{|b|}\sum_{i\in b} |d_i|, \tag{S38}$$

where the superscripts (y) and (u) denote binning by y and u, respectively. We use the two statistics generated by binning by feedback ($\bar{\mu}_b^{(u)}, \mu_b^{(u)}$) and concatenate them to generate an embedding of each task that incorporates how feedback corresponds to the value of the objective.

Dimensionality Reduction and Visualisation

All task vectors are projected to two dimensions using UMAP [42] with $n_{\text{neighbors}}$ (set to 5) and $\min_{\text{dist}}$ (set to 0.1) as hyperparameters. Each point in the resulting embedding corresponds to a single task instance, colored and shaped by its label (Figure 5).

D Pretraining and test-time algorithms

D.1 Training procedure

MDP formulation. We cast policy learning as a finite-horizon MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. At step t, the state is

$$s_t^\tau = (\widetilde{H}_t^\tau, U^\tau, Q_t^\tau), \tag{S39}$$

the action space is the current candidate pool $\mathcal{A}_t^\tau = Q_t^\tau$, and the policy selects $\boldsymbol{x}_t^\tau \sim \pi_\psi(\cdot \mid \widetilde{H}_t^\tau, U^\tau)$. The transition appends the new observation to the history and removes the selected candidate from the pool.

$$\widetilde{H}_{t+1}^\tau = \widetilde{H}_t^\tau \cup \{(\boldsymbol{x}_t^\tau, u^\tau(\boldsymbol{x}_t^\tau), y_t^\tau)\}, \qquad Q_{t+1}^\tau = Q_t^\tau \setminus \{\boldsymbol{x}_t^\tau\}, \tag{S40}$$

where $u^\tau(\boldsymbol{x})$ is the feedback of $\boldsymbol{x}$ in U^τ. The reward is the improvement of the t-th query over the best value observed strictly before round t,

$$r_t^\tau = \max\big(y_t^\tau - y_t^*, 0\big), \qquad y_t^* = \max\big(\max_{(\boldsymbol{x},y)\in H_0^\tau} y,\ \max_{i<t} y_i^\tau\big), \tag{S41}$$

and the discounted return from step t is

$$G_t^\tau = \sum_{t'=t}^{T_\tau - 1} \gamma^{t'-t}\, r_{t'}^\tau. \tag{S42}$$

We use $\gamma = 0.98$ in all experiments. The policy is optimized with REINFORCE,

$$\mathcal{L}_{\text{pol}}(\psi) = -\mathbb{E}_{\tau,\pi_\psi}\left[\frac{1}{T_\tau}\sum_{t=0}^{T_\tau-1} \log \pi_\psi(\boldsymbol{x}_t^\tau \mid \widetilde{H}_t^\tau, U^\tau)\tilde{R}_t^\tau\right], \tag{S43}$$

where $\tilde{R}_t^\tau$ is the normalized return defined below.

Return normalization. To reduce variance in the REINFORCE gradient estimates, returns are normalized per timestep across the batch,

$$\tilde{R}_t^\tau = \frac{G_t^\tau - \bar{G}_t}{s_{G,t} + \epsilon}, \tag{S44}$$

where $\bar{G}_t$ and $s_{G,t}$ are the mean and standard deviation of $\{G_t^\tau\}$ across the batch at step t, and $\epsilon = 10^{-8}$. This centers and scales the gradient signal at each step regardless of the absolute magnitude of rewards, which can vary substantially across tasks and feedback conditions.

Warmup schedule. Training proceeds in two phases. During the warmup phase, lasting η iterations, only the predictive loss $\mathcal{L}_{\text{pred}}$ is active. Candidates are sampled from the policy head, which gets no gradient during warm-up, in a smaller query pool of size $\max T_\tau = 20$. After η iterations, the pool reverts to its standard size $|Q^\tau|$, the policy head is activated, and joint optimization of (ϕ, ψ) under $\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda_{\text{pol}}\mathcal{L}_{\text{pol}}$ begins, with a new optimizer in which all parameters outside the policy head use one fifth of the learning rate.

Deployment modes. At test time, the model supports two deployment modes.

End-to-end amortized. The policy head π_ψ selects the next candidate directly,

$$\boldsymbol{x}_t^\tau \sim \pi_\psi(\cdot \mid \widetilde{H}_t^\tau, U^\tau). \tag{S45}$$

During training we sample from the softmax distribution, while during evaluation the argmax is taken.

Semi-amortized. The inference head q_ϕ is used as a feedback-aware surrogate and paired with a standard acquisition function α,

$$\boldsymbol{x}_t^\tau = \arg\max_{\boldsymbol{x} \in Q_t^\tau} \alpha\Big(\boldsymbol{x};\, q_\phi, \widetilde{H}_t^\tau, U^\tau\Big). \tag{S46}$$

In our experiments, we use q_ϕ with expected improvement (EI) and upper confidence bound (UCB) as acquisition functions, combined with the πBO feedback reweighting (FICBO-UCB and FICBO-EI, section C.1). Both modes share the same pretrained backbone and differ only in which head is used at deployment.

Implementation details. Log-probabilities in the REINFORCE objective are clamped to a minimum of -10 to prevent numerical instability from near-zero policy probabilities. Gradients are clipped to unit ℓ_2 norm at each update step. $\lambda_{\text{pol}} = 10$ scales the normalized return in the policy loss.

Algorithm. The full pretraining procedure is given in Algorithm 1.

Algorithm 1 Feedback-aware pretraining

Require: Task prior $p(\tau)$, warmup iterations η, total iterations N, loss weight λ_{pol}

for $i = 1, \ldots, N$ **do**
  Sample a batch of 200 tasks $\tau \sim p(\tau)$, with one horizon $T_\tau \sim \mathcal{U}\{10, \ldots, 20\}$ shared by the batch, each with objective f_τ, feedback source u^τ, query set Q^τ, target set $\mathcal{T}^\tau$, horizon T_τ
  Initialize $\widetilde{H}_0^\tau$ from initial context points, compute $U^\tau = \{u^\tau(\boldsymbol{x}) : \boldsymbol{x} \in Q^\tau\}$
  for $t = 0, \ldots, T_\tau - 1$ **do**
    if $i \le \eta$ **then**
      $\boldsymbol{x}_t^\tau \sim \pi_\psi(\cdot \mid \widetilde{H}_t^\tau, U^\tau)$ ▷ warmup: policy head not trained
    else
      $\boldsymbol{x}_t^\tau \sim \pi_\psi(\cdot \mid \widetilde{H}_t^\tau, U^\tau)$ ▷ policy selection
    end if
    Observe $y_t^\tau = f_\tau(\boldsymbol{x}_t^\tau) + \varepsilon_t^\tau$
    $\widetilde{H}_{t+1}^\tau \leftarrow \widetilde{H}_t^\tau \cup \{(\boldsymbol{x}_t^\tau, u^\tau(\boldsymbol{x}_t^\tau), y_t^\tau)\}$
    $Q_{t+1}^\tau \leftarrow Q_t^\tau \setminus \{\boldsymbol{x}_t^\tau\}$
    if $i > \eta$ **then**
      Record $r_t^\tau \leftarrow \max(y_t^\tau - y_t^*, 0)$
    end if
  end for
  Compute $\mathcal{L}_{\text{pred}}(\phi)$ from $\{\widetilde{H}_t^\tau, U^\tau, \mathcal{T}^\tau\}_{t=0}^{T_\tau - 1}$ ▷ Eq. (19)
  if $i \le \eta$ **then**
    Update ϕ by minimizing $\mathcal{L}_{\text{pred}}$
  else
    Compute $\mathcal{L}_{\text{pol}}(\psi)$ from $\{r_t^\tau, \log \pi_\psi(\boldsymbol{x}_t^\tau \mid \widetilde{H}_t^\tau, U^\tau)\}_{t=0}^{T_\tau - 1}$ ▷ Eq. (20)
    Update (ϕ, ψ) by minimizing $\mathcal{L}_{\text{pred}} + \lambda_{\text{pol}}\mathcal{L}_{\text{pol}}$
  end if
end for

Table S4: Final normalized best-so-far value and cumulative regret, averaged over 13 benchmarks with the marginal expert for continuous optimization. Best-so-far: higher is better. Cumulative regret: lower is better. Here the regret values are calculated using the global optimum, instead of the pool optimum.

Method	Best-so-far (↑)	Cumulative Regret (↓)
FICBO (Mixture Prior) + UCB (continuous)	$\mathbf{0.8106} \pm 0.0441$	$\mathbf{8.7665} \pm 1.6063$
FICBO (Mixture Prior) (continuous)	0.8036 ± 0.0457	8.8110 ± 1.7620
FICBO (Mixture Prior) + EI (continuous)	0.8014 ± 0.0496	8.7736 ± 1.8338
FICBO (Mixture Prior) (pool)	0.7775 ± 0.0550	9.0800 ± 1.9253
FICBO (Mixture Prior) + UCB (pool)	0.7775 ± 0.0536	9.2440 ± 1.9226
FICBO (Mixture Prior) + EI (pool)	0.7745 ± 0.0545	9.1886 ± 1.9102
GP Surrogate (pool, botorch)	0.7499 ± 0.0585	10.7315 ± 2.0342
GP Surrogate (pool)	0.7458 ± 0.0582	10.6288 ± 1.9952

E Detailed Results

The heatmaps in figs. S5 to S7 summarize the final normalized best value and the cumulative regret of all methods for each benchmark function. The results in the appendix were obtained by recomputing the results, which can lead to slightly different results due to randomness.

F Continuous BO

PFNs4BO: In-Context Learning for Bayesian Optimization [9] samples several sets of points and filters good candidates. This workflow alone does not work well with our methods since also areas with low chance to result in desirable solutions are valuable to have in the context due to the feedback. As a consequence randomly sampled points are added to the query set, which will not be optimized and serve as background information. Then the x coordinates are iteratively refined via gradient updates, either starting from the acquisition head or classic acquisition functions. We decided to simplify the setup from PFNs4BO since our problems are low-dimensional, and sampling and discarding thousands of points is a waste in this setting. The resulting procedure is as follows:

Each BO step:

1. Draws a fresh Sobol set split into an editable candidate block (100 points) and a fixed background block (200 points), matching the 300 points the pool-based variant uses
2. Scores the whole block in one forward pass
3. Runs multistart Adam ascent on the acquisition to optimize the editable coordinates only with ten steps. The background points are scored alongside but get no gradient and are excluded from the ascent objective: the query block self-attends and the head softmaxes over it, so a block of only refined, high-acquisition points would be far off the IID query distribution the model was trained on; they still stay eligible to win. Expert feedback is re-observed from the simulator without gradients, as we cannot assume the feedback is differentiable, after every step and fed back into the score (never differentiated through), and the argmax over [refined, background] is evaluated for real and appended to the context.

Table S4 shows that pool-based optimization variants have been beaten by their continuous counterparts, but this comes at the cost of drastically increasing the computational cost and number of feedback oracle calls (for every optimization step, all optimized points need feedback at that point). In our implementations, the computation time jumps from 0.5 sec to about 20 seconds for the 100 rollouts of one benchmark function.

The different acquisition functions performed very similarly. The main difference comes from the surrogates.

It is worth noting that despite the improved performance for this experiment, the policy has not been trained with continuous optimization in mind, and the optimization steps may cause the distribution of x coordinates to be OOD for what the model has been trained on, as the candidates will often cluster. Proper continuous optimization support is left as future work.

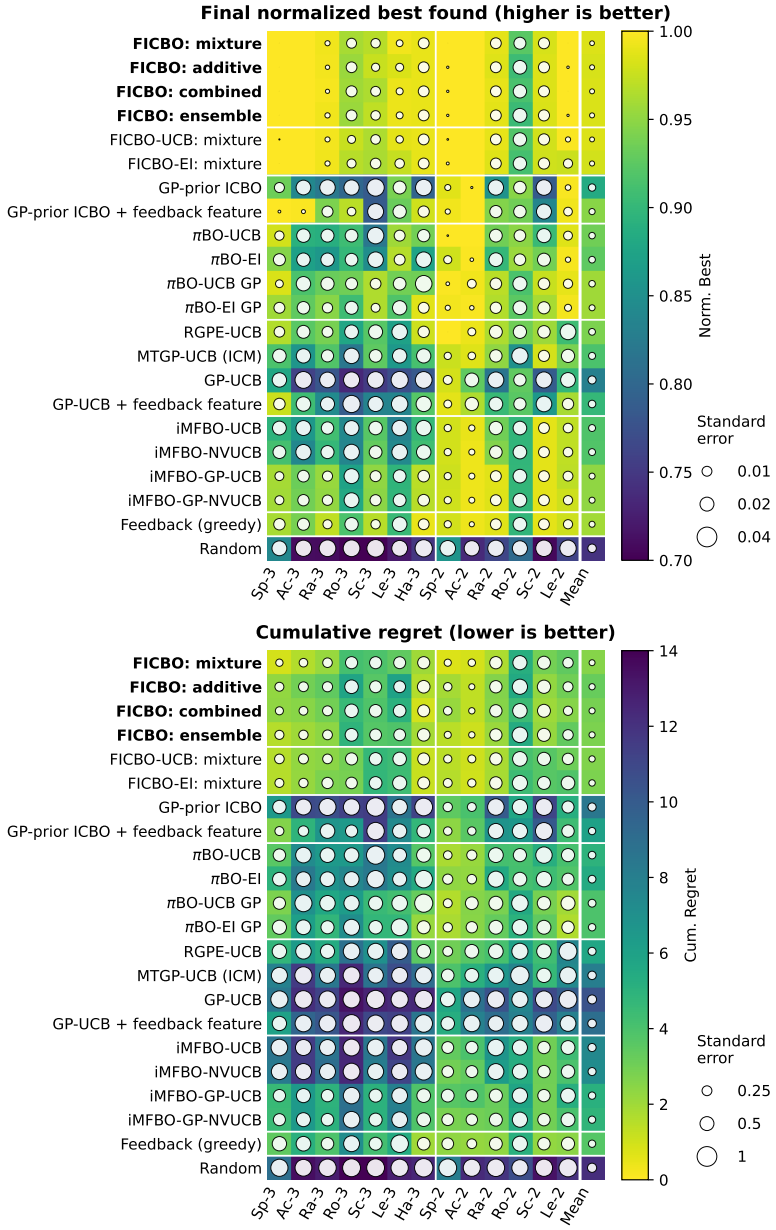


Figure S5: **Per-function results, marginal expert. Top:** final normalized best value found (↑). **Bottom:** normalized cumulative regret (↓). Columns: Sp = Sphere, Ac = Ackley, Ra = Rastrigin, Ro = Rosenbrock, Sc = Schwefel, Le = Levy, Ha = Hartmann; the suffix describes which dimension the benchmark has. Mean averages over the 13 functions. Color shows the mean over 100 replicas; circle size shows the standard error. Bold rows are FICBO with the end-to-end policy. Methods are described in section C.1.

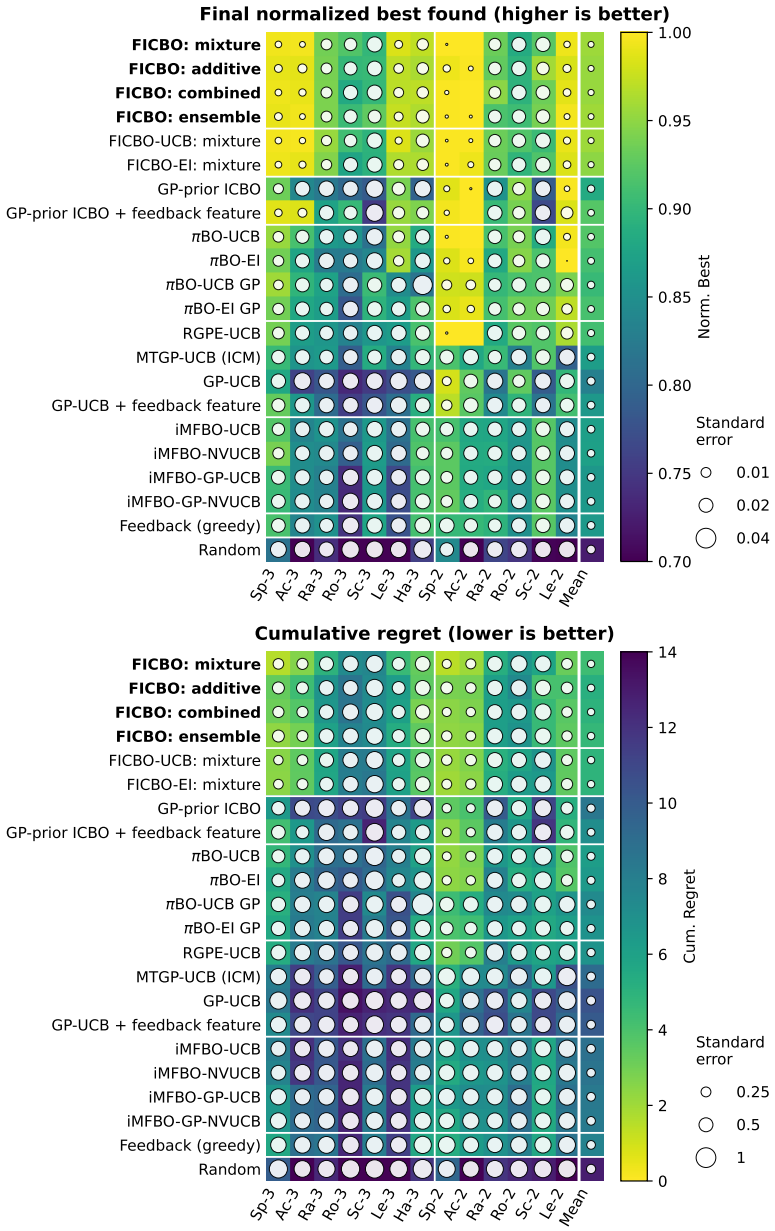


Figure S6: **Per-function results, XGBoost expert** (section C.2). Layout and abbreviations as in fig. S5.

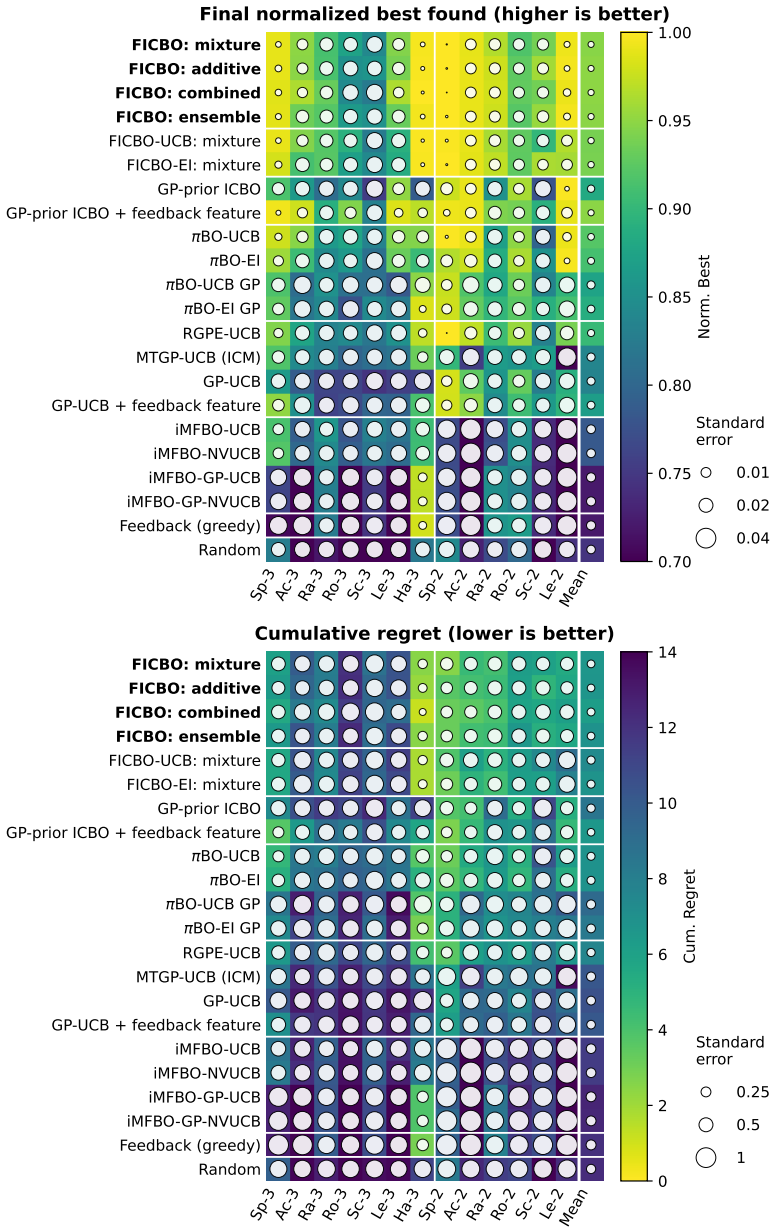


Figure S7: **Per-function results, marginal expert with fake peak** (fig. S3). Layout and abbreviations as in fig. S5.

G Detailed model architecture

G.1 Computational Resources

All experiments were conducted on a high-performance computing cluster using a single NVIDIA V100 GPU with 32GB of memory. Training took approximately 70 hours per run. All models were implemented in PyTorch [43].

G.2 Model Architecture

The model follows an encode-then-decode architecture consisting of three components: an embedder, a Transformer encoder, and an output head. We base our implementation on [19]. We do not claim that this is the optimal architecture to use, but it is good enough to demonstrate the benefit of training on our FICBO priors.

Embedder. We provide two embedder variants. The standard embedder maps input locations x through a two-layer MLP with ReLU activations to produce d_{emb}-dimensional tokens. Observed values y are embedded separately and added to the context tokens, so query and target tokens carry only location information.

The expert-aware embedder additionally incorporates the expert feedback signal. Input locations and feedback are each embedded by separate two-layer MLPs. In *concat* mode (used in the experiments), the x and feedback embeddings (each of dimension $d_{\text{emb}}/2$) are concatenated to form a d_{emb}-dimensional token; in *add* mode they are summed directly. As before, observed values y are embedded and added to context tokens only. In both variants, query and target tokens therefore carry no y information, only location, and, in the expert-aware case, feedback.

Encoder. The embeddings are processed by a Transformer encoder with N_{layers} layers, N_{heads} attention heads, embedding dimension d_{emb}, and feed-forward dimension d_{ff}. All tokens attend freely to context tokens. Optionally, all tokens can also attend to query tokens to find patterns in the feedback (turned on in experiments).

Output head. The output head splits the encoder output into query and target embeddings. Query embeddings are passed through the *policy head*, a two-layer MLP that produces a scalar logit per query point; these logits are converted to a probability distribution over candidate points via softmax, forming the policy $\pi_\psi(\boldsymbol{x} \mid \widetilde{H}_t^\tau, U^\tau)$. Target and query embeddings are passed through the *inference head*, which predicts a Gaussian mixture posterior with L components over the target outputs, parameterizing means, standard deviations, and mixture weights via separate linear projections. An optional time token t/T can be concatenated to the query embeddings before the policy head; it had negligible influence and is not used in the reported models.

G.3 Hyperparameters

These hyperparameters were used to train the models used in the experimental section. They do not represent the optimal hyperparameters for every project, but work reasonably well over a diverse set of feedback strengths and biases.

Table S5: Hyperparameter settings. Symbols follow the notation of Sec. 3 and sections B and D.1.

Hyperparameter	**Value / Range**
Synthetic task prior	
Candidate pool size N_τ	200
Horizon T_τ	10–20
Optimizer input dimension d	varies by experiment (1–3)
Number of held-out targets $\lvert\mathcal{T}^\tau\rvert$	100
Kernel family	Matérn with $\nu \in \mathcal{U}(\{3/2, 5/2, \infty\})$
Lengthscale prior	$\mathcal{U}([0.1\sqrt{d},\ 2.0\sqrt{d}])$, isotropic w.p. 0.5
Output scale prior	$\mathcal{U}([0.1,\ 1.0])$
Observation noise σ_τ	0.01 (evaluation: 0.1)

Table S5 – continued

Hyperparameter	Value / Range
Design space	$[-5, 5]^{d^{\tau}_{\text{full}}}$
Additive prior	
Source input dimension d^{τ}_{src}	$\mathcal{U}(\{1, 2\})$
Overlap dimensions o^{τ}	$\mathcal{U}(\{0, 1\})$ [c]
Source weight w_{τ}	$\mathcal{U}([0.25,\ 1.0])$
Source-private observations n_{src}	$\mathcal{U}(\{10, \ldots, 100\})$ for $d = 1$; $\mathcal{U}(\{30, \ldots, 300\})$ for $d \in \{2, 3\}$
Fraction of $\mathcal{D}^{\tau}_{\text{src}}$ sampled uniformly	0.25 (remainder sampled in clusters)
Source cluster std	$\mathcal{U}([0.5,\ 1.5])$
Number of source cluster centers	$\mathcal{U}(\{1, \ldots, 5\})$
Mixture prior	
Number of real components K_{real}	3
Number of decoy components K^{τ}_{decoy}	$\mathcal{U}(\{0, \ldots, 5\})$ per batch (Combined model: $\{2, \ldots, 8\}$)
Dirichlet concentration α_{src}	0.7
Component inclusion probability p_{src}	0.8
Bias operators $\mathcal{B}_{\tau}$ [a]	
Activation probabilities (section B.5)	additive GP bias: 0.15; local distortion: 0.15; constant shift: 0.15; catastrophic failure: 0.05; knowledge-center noise: 1.0 (Mixture), off (Additive)
Noise: scale σ_{fb}	$\mathcal{U}([0.0,\ 0.2])$
Noise: knowledge-center scale	$\mathcal{U}([0.0,\ 1.0])$ (Mixture only)
Additive GP bias: lengthscale	$\mathcal{U}([0.5,\ 3.0])$
Additive GP bias: scale	$\mathcal{U}([0.2,\ 2.0])$
Additive GP bias: max std	$\mathcal{U}([0.2,\ 1.0])$
Local distortion: number of centers n_{centers}	$\mathcal{U}(\{1, 2\})$
Local distortion: magnitude	$\mathcal{U}([0.2,\ 1.0])$, random sign
Local distortion: decay	$\mathcal{U}([0.5,\ 1.5])$
Constant shift: δ	$\mathcal{U}([-0.5,\ 0.5])$
Transformer backbone	
Embedding dimension d_{emb}	32
Number of layers N_{layers}	3
Number of attention heads N_{heads}	4
Feed-forward dimension d_{ff}	128
Dropout	0.0
Heads	
Number of mixture components L	10
Hyperparameter targets in $\mathcal{L}_{\text{pred}}$	d+1 (d lengthscales + output scale)
Policy head	two-layer MLP, applied to each query token
Training	
Optimizer	AdamW
Learning rate	10^{-3} (after warm-up: $2 \cdot 10^{-4}$ for all parameters except the policy head)
Weight decay	10^{-7} (after warm-up: 10^{-2}, AdamW default)
LR schedule	cosine annealing ($T_{\max} = 200{,}000$ during warm-up, reset to 180,000 after warm-up)
Batch size (parallel tasks)	200
Total iterations N	200,000
Warm-up iterations η	20,000 [b]
Discount γ	0.98
Policy loss weight λ_{pol}	10

Table S5 – continued

Hyperparameter	**Value / Range**
Return-normalization constant ϵ	10^{-8}
Log-probability clamp	-10
Gradient clipping	unit ℓ_2 norm
Evaluation	
Initial observations $\lvert H_0^\tau \rvert$	1
Candidate pool size N_τ	300
Horizon T_τ	30
Number of seeds	100

[a] Due to computational constraints, the hyperparameters were not optimized and were set arbitrarily.
[b] Following ALINE [19], the warm-up uses only the prediction loss and a smaller query size.
[c] With this range, $o^\tau = d$ never occurs for the $d \in \{2, 3\}$ models reported in the main paper. It does occur for the $d = 1$ trained variant (see Appendix B.2), where the optimizer then has no private features, and we default to $w_\tau = 1$, since there is no optimizer-only component to weigh against.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims are supported by experiments in later sections.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The discussion section has a paragraph about limitations addressing some core limitations.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A].

   Justification: No theoretical results to prove.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The appendix contains extensive descriptions of the architecture, hyperparameters (section G), benchmarks (section C) and the prior variants (section B).

   Guidelines:

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

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

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

   Answer: [Yes]

   Justification: Code is publicly available on GitHub. The repository is linked in the paper.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

Justification: The appendix contains extensive descriptions of the architecture, hyperparameters (section G), benchmarks (section C) and the prior variants (section B).

Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Our plots display error bars with explanation, usually the 95% confidence interval is visualized.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer:[Yes]

   Justification: we refer to section G.1.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: the work is methodological. It does not involve human subjects, private data, scraped data, or high-risk deployment.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [N/A]

Justification: Not relevant.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: Not relevant.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We use NeuralFoil [37] and NAS-Bench-201 [39]. License information for all assets is provided in the appendix.

Guidelines:

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

13. **New assets**

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

Answer: [N/A]

Justification: The publication does not introduce new assets.

Guidelines:

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

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

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

Answer: [N/A]

Justification: Not relevant.

Guidelines:

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

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

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

Answer: [N/A]

Justification: Not relevant.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A].

Justification: No important, original, or non-standard LLM use.

Guidelines:

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