IQS-BO: IN-CONTEXT QUERY SELECTION FOR BAYESIAN OPTIMISATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian Optimisation (BO) is a powerful framework for the optimisation of expensive black-box functions, but typically requires refitting a surrogate and maximising an acquisition function at every evaluation step. In-context approaches based on Prior-data Fitted Networks (PFNs) amortise part of this cost by pre-training transformers on functions drawn from synthetic priors. PFNS4BO amortises the surrogate but still relies on a numerically maximised acquisition function, while FIBO performs BO fully in-context by sampling optimiser locations from a learned density, which fixes the decision rule and admits no surrogate. Learned acquisition functions score a finite candidate set with a trained network, but, lacking a label for the query, learn the score by reinforcement learning on previously solved tasks. We propose IQS-BO, a PFN that learns the query decision by supervised learning on synthetic priors. In a single forward pass, IQS-BO predicts the probability that each candidate maximises the objective over the set, and we show that the minimiser of its objective is the posterior probability of this event. The model can be pre-trained without a surrogate for fully in-context BO, or take the predictions of a fixed probabilistic surrogate as additional input, amortising only the decision step. Our method proposes queries at a fraction of the cost of acquisition-based methods, while either matching or outperforming standard BO with Gaussian processes (GPs) and available in-context methods on synthetic and real-world benchmarks. Finally, we propose a mixture prior for pre-training PFNs which combines samples from GPs with functions exhibiting warped inputs, isolated narrow optima, or plateaus that are poorly modeled by stationary kernels common in GP surrogates. We show that pre-training on this prior can lead to improved optimisation performance.

1 INTRODUCTION

Bayesian optimisation (BO) (Jones et al., 1998; Shahriari et al., 2016; Garnett, 2023) aims to maximise an expensive black-box function, and has proven particularly beneficial when objective evaluations are costly and only a limited number of evaluations is available. Given an initial dataset, BO typically alternates between fitting a probabilistic surrogate and selecting the next evaluation point by maximising an acquisition function. Both steps are repeated for every new observation: the surrogate must adapt as the dataset grows, while the acquisition function is a non-convex objective whose optimisation often requires dedicated sampling, restart, and numerical optimisation procedures (Wilson et al., 2018).

In-context methods can reduce per-round costs by pre-training transformer architectures on functions drawn from priors from which sampling is efficient (Müller et al., 2022). Existing methods differ in which part of the BO process they amortize (Figure 1).

PFNS4BO (Müller et al., 2023) amortizes the surrogate with a Prior-data Fitted Network (PFN) which returns the posterior predictive of each input in a single forward pass. Surrogate fitting is no longer required, since the model adapts to new observations in context, but an acquisition function must still be specified and numerically maximised at every round. Moreover, the acquisition function requires a closed form on the PFN's output.

FIBO (de Carvalho et al., 2025) removes the surrogate fit as well as the choice and maximisation of the acquisition function. A pre-trained PFN encoder embeds the observations, and a normalizing flow conditioned on that embedding models the distribution of the optimum's location which is sampled

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

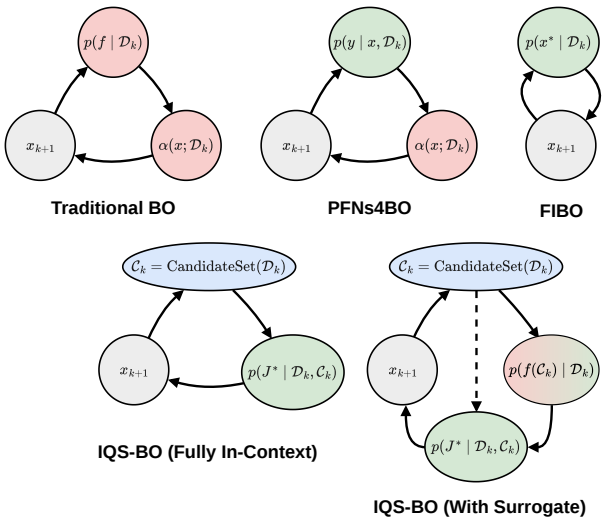


Figure 1: One optimisation round of traditional BO, PFNs4BO (Müller et al., 2023), FIBO (de Carvalho et al., 2025) and IQS-BO with and without a surrogate. $\mathcal{D}_k$: observations after k evaluations, x_{k+1}: next query, $\mathcal{C}_k$: candidates drawn from $\mathcal{D}_k$, α: acquisition function, J^*: index of the best candidate. Red: fitted or optimised every round. Green: one forward pass of a pre-trained network. Red-to-green: fixed surrogate of either kind. Blue: sampling without learning. Grey: quantities passed between steps.

directly to obtain proposals. The learned object is a density over the search space, and the authors show that sampling from it amounts to Thompson sampling under the pre-training prior. The method is especially suitable for high-throughput optimisation, since a batch of any size can be drawn almost for free as a set of i.i.d. draws, but implies a fixed query selection policy.

Relevant to our work, learned acquisition functions replace the analytic rule by a network that scores a finite candidate set. NAP (Maraval et al., 2023) uses a transformer neural process that reads the observations and candidates and returns a score for each, learning a surrogate and the acquisition rule jointly. POLAR (Huang et al., 2026) learns a similar score on top of the representations of a pretrained predictive model. Since the best point to query is unknown on real data, both learn the score from rewards, by reinforcement learning or policy gradient on rollouts over solved tasks.

To the best of our knowledge, no method learns the query decision by supervised learning on synthetic priors. Under such priors the training label is free, since each pre-training function can be evaluated at every candidate, so the best one is known and query selection becomes a classification problem. This is the basis for In-context Query Selection for Bayesian Optimisation (IQS-BO).

Concretely, IQS-BO trains a transformer that reads the observations and a candidate set and predicts, for each candidate, the probability that it has the largest function value in the set. The network is trained with a cross-entropy loss, whose minimiser is the posterior probability that each candidate is the best. At deployment, IQS-BO queries the candidate with the highest predicted probability, which approximates the Bayes-optimal choice among the candidates under the pre-training prior.

The proposed framework admits two implementations. First, the network can be trained directly on the observations, without a surrogate. This yields a model for fully in-context BO that, like FIBO, requires neither surrogate fitting nor acquisition function optimisation at deployment. Second, the network can take as additional input the predictions of a fixed probabilistic surrogate. We show this with a released PFNs4BO model and with a Gaussian process whose hyperparameters are fitted to each synthetic task during pre-training and refitted at every iteration during inference.

Empirically, IQS-BO, pre-trained for a fixed input dimension and batch size, (i) outperforms PFNs4BO pre-trained on the same prior and deployed via acquisition-function maximisation with L-BFGS-B, with gains increasing as both context size and batch size grow; (ii) matches or outperforms GP-based BO, FIBO and NAP across real and synthetic benchmarks in three and four dimensions; and (iii) excels in batch optimisation, reaching a better performance than FIBO at every batch size while maintaining comparable computational cost and proposing evaluations one to three orders of magnitude faster than PFNs4BO and GP-based baselines.

Finally, we propose and evaluate a pre-training prior that augments GP samples with function families that are poorly captured by fixed-kernel GPs. We show that PFN-based models pre-trained on this prior achieve better optimisation performance than on priors that only include GP samples. The gains are largest when the initial context is small and become less pronounced as the context grows.

1.1 Summary of Main Contributions

Our main contributions are:

- We formulate query selection in BO as supervised classification over a candidate set, where the class to predict is the index of the best candidate. Because this label is exact and cheap to compute for functions sampled from a prior, the selection rule can be learned with a standard cross-entropy loss.
- We show that the cross-entropy objective is minimised by the posterior probability that a candidate is the best element of the candidate set. Consequently, the learned policy approximates the Bayes-optimal decision rule induced by the pre-training prior.
- We introduce IQS-BO, a transformer that selects the next query in-context by scoring all candidates in a single forward pass. The model can be pre-trained without a surrogate, enabling fully in-context BO, or combined with a fixed probabilistic surrogate, such as a pre-trained PFN or a GP, to amortise the query-selection step only. Batches are constructed sequentially, with each selection conditioned on the preceding ones.
- We study the impact of the pre-training prior and propose a new prior that combines GP samples with warped, localised and piecewise-flat functions. We show that pre-training on this prior can improve the optimisation performance of PFN-based methods for BO.

2 Problem Statement

We consider the maximisation of an expensive black-box function $f : \mathcal{X} \to \mathbb{R}$ over $\mathcal{X} = [0, 1]^d$. At each optimisation round, we have observed $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, with noisy evaluations $y_i = f(x_i)+\varepsilon_i$, and must choose the next point at which to evaluate f. Standard BO makes this choice by fitting a probabilistic surrogate $p(f \mid \mathcal{D})$ and maximising an acquisition function α over $\mathcal{X}$.

Our goal is to replace the maximisation of the acquisition function or, more generally, the selection of the next query, by a decision rule that is learned in-context. At each optimisation round, a stochastic sampler constructs a finite candidate set $\mathcal{C} = \{c_1, \dots, c_M\} \subset \mathcal{X}$ of M points from $\mathcal{D}$ (Appendix C.2). Instead of maximising α over $\mathcal{X}$, we learn to select one candidate from $\mathcal{C}$ directly. Let

$$J^* = \operatorname*{arg\,max}_{k=1,\dots,M} f(c_k) \tag{1}$$

denote the index of the best candidate. Given the context $\mathcal{D}$ and the candidate set $\mathcal{C}$, we select the candidate with the largest posterior probability of achieving the highest function value in the set, i.e.,

$$j^\star = \operatorname*{arg\,max}_{j=1,\dots,M} \mathbb{P}(J^* = j \mid \mathcal{D}, \mathcal{C}). \tag{2}$$

3 Methodology

We now describe how the network p_θ, with parameters θ, is trained to approximate the posterior

$$\mathbb{P}(J^* = j \mid \mathcal{D}, \mathcal{C}), \qquad j = 1, \dots, M, \tag{3}$$

and how it selects the next query at inference. The network reads the observations and the candidates and returns (3) for every candidate in a single forward pass. It is pre-trained on functions sampled from a synthetic prior, so that it selects queries in-context without any parameter update, and it can take the predictions of a fixed surrogate ϕ as additional input (Section 3.4). We first state the training objective (Section 3.1) and its Bayesian interpretation (Section 3.2), then describe the architecture (Section 3.3), pre-training (Section 3.4) and inference (Section 3.5). We denote our method by IQS-BO.

3.1 Training Objective

The network p_θ is pre-trained on synthetic tasks $(\mathcal{D}, \mathcal{C}, J^*)$, obtained by drawing a function f from a pre-training prior $p(f)$, a context $\mathcal{D}$ of noisy evaluations of f, a candidate set $\mathcal{C}$ from $\mathcal{D}$, and the label J^* of (1). Since f is known during pre-training, it can be evaluated without noise at every candidate,

so J^* is computed exactly, at the cost of M function evaluations and one $\arg\max$. The parameters θ are learned by minimising the expected cross-entropy

$$\mathcal{L}(\theta) = \mathbb{E}\big[-\log p_\theta(J^* \mid \mathcal{D}, \mathcal{C})\big], \tag{4}$$

where the expectation is over tasks generated as above. In practice, (4) is estimated by its average over mini-batches of sampled tasks. Its minimiser is the posterior (3), so pre-training reduces to a multiclass classification problem whose optimal solution is the posterior probability that each candidate is the best in the set, as shown in Proposition 1.

3.2 Posterior-Best-Candidate Interpretation

The cross-entropy objective (4) admits a simple Bayesian interpretation. Write $J^* = J^*(f, \mathcal{C})$ for the label (1), viewed as a function of the sampled function f and the candidate set $\mathcal{C}$. Training tasks are generated as $f \sim p(f)$, $\mathcal{D} \sim p(\mathcal{D} \mid f)$ and $\mathcal{C} \sim p(\mathcal{C} \mid \mathcal{D})$, so (4) is the population loss under the joint law of $(f, \mathcal{D}, \mathcal{C})$. Since the candidate sampler uses only $\mathcal{D}$ and independent randomness, $\mathcal{C} \perp f \mid \mathcal{D}$. The network output $p_\theta(\cdot \mid \mathcal{D}, \mathcal{C})$ is a probability mass function on $\{1, \dots, M\}$.

Proposition 1. *If the model is well specified, that is, $p_\theta(\cdot \mid \mathcal{D}, \mathcal{C}) = \mathbb{P}(J^* = \cdot \mid \mathcal{D}, \mathcal{C})$ almost surely for some θ, then every minimiser θ^* of $\mathcal{L}$ satisfies, almost surely and for all j,*

$$p_{\theta^*}(j \mid \mathcal{D}, \mathcal{C}) = \mathbb{P}(J^* = j \mid \mathcal{D}, \mathcal{C}) = \mathbb{P}_{f \sim p(f \mid \mathcal{D})}\big(J^*(f, \mathcal{C}) = j\big).$$

Proof. Let $\pi_j = \mathbb{P}(J^* = j \mid \mathcal{D}, \mathcal{C})$ and $H(\pi) = -\sum_j \pi_j \log \pi_j \le \log M$. Conditioning on $(\mathcal{D}, \mathcal{C})$, with all outer expectations below taken over $(\mathcal{D}, \mathcal{C})$,

$$\mathcal{L}(\theta) = \mathbb{E}\Big[-\sum_{j=1}^{M} \pi_j \log p_\theta(j \mid \mathcal{D}, \mathcal{C})\Big] = \mathbb{E}\big[H(\pi)\big] + \mathbb{E}\Big[\mathrm{KL}\big(\pi \,\big\|\, p_\theta(\cdot \mid \mathcal{D}, \mathcal{C})\big)\Big].$$

The first term does not depend on θ, and the second is non-negative and vanishes if and only if $p_\theta(\cdot \mid \mathcal{D}, \mathcal{C}) = \pi$ almost surely. Well-specification makes the value $\mathbb{E}[H(\pi)]$ attainable, so the minimisers are exactly such θ. Finally, J^* is a function of $(f, \mathcal{C})$, and $p(f \mid \mathcal{D}, \mathcal{C}) = p(f \mid \mathcal{D})$ because $\mathcal{C} \perp f \mid \mathcal{D}$, which gives the last equality. □

Consequently, a network trained by minimising (4) approximates the target (3), the posterior probability that each candidate is the best in the set. Under a 0–1 loss on the identity of the best candidate, selecting $\arg\max_j \mathbb{P}(J^* = j \mid \mathcal{D}, \mathcal{C})$ is the Bayes-optimal decision. The network, which selects $\arg\max_j p_\theta(j \mid \mathcal{D}, \mathcal{C})$, therefore approximates the Bayes-optimal myopic decision rule on the candidate set.

3.3 Architecture

The network p_θ is a transformer that reads $\mathcal{D}$ and $\mathcal{C}$ and returns a probability for every candidate in a single forward pass. Each context point (x_i, y_i) is represented by the token $s = i$ and each candidate c_j by the token $s = n + j$, giving $n + M$ tokens, each with a feature vector t_s that includes the coordinates of the point and its relation to the data, and differs between context and candidate tokens (Appendix B). If a fixed surrogate ϕ is used, its outputs at each token, collected in a vector m_s, are added to the embedding

$$h_s^{(0)} = W_{\mathrm{feat}}\, t_s + W_\phi\, m_s, \tag{5}$$

where W_{feat} and W_ϕ are learned and the second term is absent without a surrogate.

The tokens are processed by a transformer encoder with L layers and no positional encoding. The encoder maps the embeddings $h_s^{(0)}$ to final states $h_s^{(L)}$, and a scoring head g_θ maps the final state of each candidate token to a score $z_j = g_\theta\big(h_{n+j}^{(L)}\big)$. A softmax over the candidates gives

$$p_\theta(j \mid \mathcal{D}, \mathcal{C}) = \frac{\exp(z_j)}{\sum_{k=1}^{M} \exp(z_k)}, \qquad j = 1, \dots, M. \tag{6}$$

Context tokens attend only to each other and candidate tokens only to context tokens, so each score depends only on $\mathcal{D}$ and its own candidate, and M can differ between pre-training and inference (Appendix B.2).

Algorithm 1 Pre-training IQS-BO (no surrogate, single query)

Require: prior $p(f)$, network parameters θ, candidate-set size M

for each optimiser step **do**
  Sample $f \sim p(f)$ and a context $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ from f ▷ Section 3.4
  Draw $\mathcal{C} = \textsc{CandidateSet}(\mathcal{D}, M)$ ▷ Appendix C.2
  Compute J^* by (1) on noise-free f
  Compute $p_\theta(\cdot \mid \mathcal{D}, \mathcal{C})$ and take a gradient step on $-\log p_\theta(J^* \mid \mathcal{D}, \mathcal{C})$ ▷ Section 3.3
end for

Algorithm 2 One optimisation round of IQS-BO (single query)

Require: context $\mathcal{D}$, network θ, surrogate ϕ or $\varnothing$, candidate-set size M

Draw $\mathcal{C} = \textsc{CandidateSet}(\mathcal{D}, M)$
If $\phi \neq \varnothing$, compute the surrogate outputs $m = \phi(\mathcal{D}, \mathcal{C})$
Compute $p_\theta(\cdot \mid \mathcal{D}, \mathcal{C})$ in one forward pass, with m as input if available
Select $x_{\text{next}} = c_{\hat{\jmath}}$ with $\hat{\jmath} = \arg\max_{j \leq M} p_\theta(j \mid \mathcal{D}, \mathcal{C})$
Evaluate f at x_{next}, observe y_{next} and set $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_{\text{next}}, y_{\text{next}})\}$

3.4 Pre-Training

Pre-training only requires sampling functions from a prior and evaluating them at arbitrary points. We use two priors. The first is the HEBO prior of PFNs4BO (Müller et al., 2023), sampled with the hyperparameters of its released model. The second is a mixture prior that combines GP samples with three types of functions that stationary GP surrogates model poorly, namely input-warped GP samples, sums of local Gaussian bumps and piecewise-flat plateau functions (Appendix C.1).

Without a surrogate, the network reads only the observed data and is trained as in Algorithm 1. It learns from $\mathcal{D}$ both what the data imply about f and which candidate to select, and the result is a fully in-context model that requires neither surrogate fitting nor an acquisition function at inference.

With a surrogate ϕ, Algorithm 1 has one additional step. Before the forward pass, ϕ is evaluated on the context of the task, and its outputs m_s enter the embedding (5). The loss, the data and the rest of the network are unchanged. In our experiments we use two surrogates:

- IQS-BO-PFN uses the released PFNs4BO-HEBO model (Müller et al., 2023), with parameters kept frozen during training. The network reads a learned combination of the PFN's hidden states at every token and five summaries of its predictive distribution at every candidate. Since both the surrogate and IQS-BO are in-context, this model performs BO fully in-context.
- IQS-BO-GP uses a GP with a Matérn-5/2 kernel with automatic relevance determination (ARD), whose hyperparameters are fitted by maximum a posteriori (MAP) estimation to the context of each pre-training task. The network reads the posterior mean, the posterior standard deviation, the posterior correlation with the best context point and the fitted hyperparameters. At inference, the GP is refitted at every evaluation, as in standard BO.

All three variants can be pre-trained to select batches of q queries (Appendix B.3). The q points are selected one at a time, and each earlier pick is passed to the network as a pending token, so that it learns to take previous decisions into account. Selecting the best q points from a candidate set is a different decision problem for every q. For optimal performance, each model was pre-trained for a specific batch size. Depending on the variant, pre-training takes between 2.7 and 6.1 hours on a single NVIDIA H100 GPU (Appendix C.3).

3.5 Inference

After pre-training, the network is deployed without any parameter update (Algorithm 2). At every round a new candidate set with M elements is drawn, one forward pass gives $p_\theta(\cdot \mid \mathcal{D}, \mathcal{C})$, and the candidate with the highest probability is evaluated. A round of IQS-BO costs one forward pass, IQS-BO-PFN adds one forward pass of the frozen PFN, and IQS-BO-GP adds one GP fit. Batches are selected sequentially with pending tokens, as described in Section 3.4 and Appendix B.3.

4 EMPIRICAL EVALUATION

This section evaluates IQS-BO in four steps. Section 4.2 compares its three variants, without a surrogate, with a PFNs4BO surrogate (IQS-BO-PFN) and with a GP surrogate (IQS-BO-GP), against GP-based and in-context baselines. Section 4.3 isolates the effects of the learned query selection and of the mixture prior. Section 4.4 evaluates batch optimisation, and Section 4.5 ablates the design of the candidate set and the sequential batch rule. Implementation details, datasets, baselines, metrics and additional results are given in Appendices B–F.

4.1 EXPERIMENTAL SETUP

We use ten 3D problems, eight synthetic and two real-world, and five 4D problems, four synthetic and one real-world, two of which were chosen to be suited for GP surrogates (Appendix E). As baselines, we use GP-based BO with MAP (GP-MAP) and full Bayes (GP-FB), as well as two released PFNs4BO models, FIBO, NAP (Maraval et al., 2023) and Sobol search (Appendix D).

All runs start from the same Sobol design of $n_{\text{init}} \in \{10, 20, 40\}$ points and run for $T = 30$ rounds with $q = 2$, with 90 seeds per problem. We report the regret $\tilde{r}(t) = f^* - f(x_t^+)$ after t rounds, where x_t^+ is the incumbent, clipped at zero and normalised by the mean regret of the initial design on each problem. Each run is summarised by the average regret $R^{\text{AUC}} = \frac{1}{T}\sum_{t=1}^{T} \tilde{r}(t)$, which rewards reaching low regret early, and by $R^{\text{final}} = \tilde{r}(T)$, which measures the regret at the final round. To compare methods across problems, we rank them on each problem and seed by these quantities, with 1 being best, and report the average ranks. For pairwise comparisons, we report the win rate, the share of paired runs, with the same problem, seed and initial design, in which a method attains the lower regret. For batch optimisation, we report the GAP (Huang et al., 2006), $(y_t - y_0)/(y^* - y_0)$, where y_t is the best value after t evaluations, y_0 the best value in the initial design and y^* the optimum.

4.2 MAIN RESULTS

Figure 2 shows the average regret and the average rank per round with a starting context of $n_{\text{init}} = 40$ on the 3D problems, and Table 1 reports the average ranks at every context on the 3D and 4D problems. In both, methods pre-trained on the mixture prior are compared separately from methods pre-trained on the HEBO or GP-based priors. GP-MAP, GP-FB and Sobol require no pre-training and are included in all comparisons. Results at $n_{\text{init}} \in \{10, 20\}$ and per-problem results are given in Appendix F.1.

Table 1: Mean rank on R^{AUC} (left) and R^{final} (right), over 90 seeds and the ten 3D or five 4D problems, $\pm$ 95% bootstrap CI half-width. † mixture prior, * HEBO or GP prior. Ranks are computed within each block, and bold marks the best rank per column.

	R^{AUC}						R^{final}					
	$d=3$			$d=4$			$d=3$			$d=4$		
n_{init}	10	20	40	10	20	40	10	20	40	10	20	40
IQS-BO†	**3.1**±.1	2.8±.1	2.8±.1	3.3±.1	3.5±.1	3.4±.1	**2.8**±.1	2.6±.1	2.7±.1	3.4±.1	3.5±.1	3.3±.1
IQS-BO-PFN†	3.1±.1	**2.7**±.1	**2.6**±.1	3.6±.1	3.5±.1	3.3±.1	2.8±.1	**2.5**±.1	**2.5**±.1	3.5±.1	3.5±.1	3.3±.1
IQS-BO-GP†	3.6±.1	3.3±.1	3.3±.1	3.6±.1	**3.2**±.1	**3.1**±.1	2.8±.1	2.8±.1	2.7±.1	**3.2**±.1	**3.0**±.1	**2.9**±.1
PFNs4BO†	3.8±.1	4.2±.1	4.2±.1	4.1±.1	4.4±.1	5.1±.1	4.9±.1	5.1±.1	5.1±.1	5.2±.1	5.3±.1	5.4±.1
FIBO†	5.5±.1	5.6±.1	5.6±.1	6.1±.1	5.8±.1	5.3±.1	5.8±.1	5.9±.1	5.9±.1	5.6±.1	5.6±.1	5.6±.1
GP-MAP	4.3±.1	4.5±.1	4.8±.1	**3.2**±.1	3.3±.1	3.6±.1	4.5±.1	4.6±.1	4.6±.1	3.5±.1	3.5±.1	3.8±.1
GP-FB	5.2±.1	5.1±.1	4.9±.1	4.5±.1	4.5±.1	4.3±.1	4.6±.1	4.7±.1	4.8±.1	3.9±.1	3.7±.1	3.8±.1
Sobol	7.5±.1	7.7±.1	7.8±.1	7.6±.1	7.9±.0	7.9±.0	7.7±.1	7.7±.1	7.8±.0	7.8±.0	7.9±.0	7.9±.0
IQS-BO*	4.2±.1	**3.9**±.1	3.6±.1	4.7±.2	4.4±.2	3.9±.1	3.1±.1	3.0±.1	**3.0**±.1	3.8±.2	3.5±.1	3.2±.1
IQS-BO-PFN*	5.0±.1	4.6±.1	4.3±.1	5.1±.2	4.9±.2	4.6±.2	4.4±.1	4.1±.1	3.7±.1	4.3±.2	4.2±.2	4.1±.1
IQS-BO-GP*	4.4±.1	4.0±.1	**3.6**±.1	4.7±.2	3.9±.2	3.7±.1	**2.9**±.1	**2.8**±.1	3.0±.1	3.8±.2	**3.1**±.1	**2.7**±.1
PFNs4BO*	**3.6**±.1	3.9±.1	4.5±.1	3.9±.2	4.7±.2	5.1±.2	5.1±.1	5.4±.1	5.7±.1	5.1±.2	5.7±.1	6.0±.1
FIBO*	5.3±.1	5.5±.1	5.4±.1	5.5±.2	5.5±.1	5.6±.1	6.3±.1	6.3±.1	6.2±.1	5.8±.1	6.1±.1	6.1±.1
NAP*	6.0±.1	6.1±.1	6.5±.1	6.0±.2	5.7±.2	5.9±.1	6.2±.1	6.1±.1	6.1±.1	6.9±.1	6.8±.1	6.6±.1
GP-MAP	3.6±.1	3.9±.1	4.1±.1	**2.7**±.1	**2.8**±.1	**3.3**±.1	4.2±.1	4.4±.1	4.3±.1	**3.2**±.1	3.3±.1	3.7±.1
GP-FB	4.9±.1	4.7±.1	4.4±.1	4.4±.1	4.4±.1	4.1±.1	4.3±.1	4.5±.1	4.5±.1	3.7±.1	3.6±.1	3.7±.1
Sobol	8.0±.1	8.4±.1	8.6±.1	8.2±.1	8.7±.1	8.8±.1	8.3±.1	8.5±.1	8.6±.1	8.5±.1	8.7±.0	8.8±.0

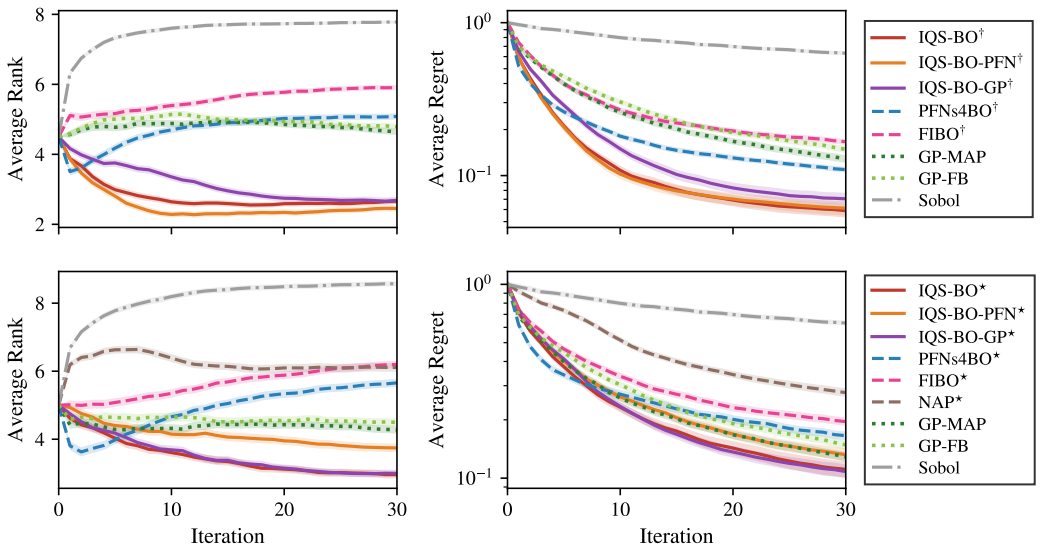


Figure 2: Average rank (left) and average regret (right) per iteration, ten 3D problems, $n_{\text{init}} = 40$, 90 seeds, $q = 2$, with 95% confidence intervals. Top, methods pre-trained on the mixture prior (†). Bottom, methods pre-trained on the HEBO prior or on GP draws (*). Ranks are computed within each row, and regret is normalised by the mean initial regret of each problem.

Mixture Prior Under the mixture prior, all three variants of IQS-BO consistently outperform the competing learned methods. On the 3D benchmarks, they achieve the best mean rank on both R^{AUC} and R^{final} across all initial context sizes, substantially outperforming PFNs4BO and FIBO despite being trained on the same prior. Figure 2 shows that the three variants behave similarly, indicating that the gain is mostly due to the learned query selection rather than to the surrogate choice. In 4D the GP baselines become more competitive, partly because two of the five benchmark problems are well specified by GP surrogates. Nevertheless, IQS-BO-GP attains the best R^{final} rank at every context size, while all three variants of IQS-BO significantly outperform PFNs4BO and FIBO. Overall, the results indicate that IQS-BO outperforms in-context methods pre-trained on the same prior, and matches or outperforms GP-based BO.

HEBO and GP-based Priors Similar results are found when comparing models trained on the HEBO or GP-based priors. At small context sizes, PFNs4BO and GP-based BO remain competitive in terms of average regret. As more observations become available, IQS-BO achieves the best or near-best final regret across all 3D settings and the strongest final-regret performance in 4D for moderate and large context sizes. PFNs4BO often performs well at the beginning of the optimization process, whereas IQS-BO consistently reaches better solutions by the end of the budget. Finally, IQS-BO achieves these results at substantially lower computational cost. Because each query requires only forward passes, without surrogate fitting or acquisition function maximisation, IQS-BO proposes queries about 11 times faster than GP-MAP and more than two orders of magnitude faster than PFNs4BO-HEBO (Appendix F.4).

4.3 Controlled Comparisons and Effect of the Prior

Learned Query Selection Against Expected Improvement The methods compared in Section 4.2 differ in more than how they select queries. To isolate the contribution of the network, we compare IQS-BO-GP, pre-trained on the mixture prior, with GP-MAP, which fits the same GP at every round. GP-MAP maximises expected improvement over the whole domain, whereas IQS-BO-GP lets the network choose among a finite set of candidates. We therefore add a control, EI-CAND, which uses the GP, the candidate set and the batch rule of IQS-BO-GP but scores the candidates by expected improvement, so that IQS-BO-GP and EI-CAND differ only in how candidates are scored (Appendix F.2). IQS-BO-GP outperforms EI-CAND at every context size, and its advantage grows as more observations become available (Figure 3, left, and Table 15). EI-CAND in turn outperforms GP-MAP, mainly because its candidate set includes perturbations of the best

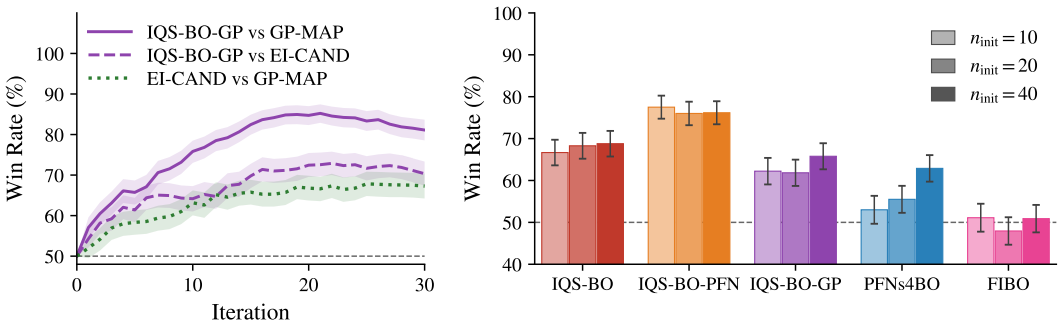


Figure 3: Paired win rates (Section 4.1), ten 3D problems, 900 paired runs, mean with 95% CI. Left, on $\tilde{r}(t)$ per iteration, $n_{\text{init}} = 40$. Right, on R^{final}, each model pre-trained on the mixture prior against its original prior.

observations (Appendix F.5). Taken together, these comparisons show that the network selects better queries than expected improvement from the same GP and the same candidates, and that both the network and the candidate design contribute to the gain of IQS-BO-GP over GP-MAP.

Effect of the Prior We pre-train and evaluate other in-context BO methods on the mixture prior. Figure 3 compares each model pre-trained on the mixture prior with the same model pre-trained on its original GP-based prior, which measures the gain of the mixture over pre-training on GP samples alone(Appendix F.3). The mixture prior improves IQS-BO, IQS-BO-PFN and PFNs4BO in most runs, most strongly in the early rounds, and the same pattern holds in 4D. This gain can be explained by the more complex and varied functional forms, such as local bumps and plateaus, which the models see during pre-training and which a GP prior alone specifies poorly. Nonetheless, models pre-trained on the mixture prior perform as well as their GP-prior counterparts on problems for which GP surrogates are well specified, such as Rosenbrock and the GP-prior draws, with the exception of a few cases discussed in Appendix F.3. PFNs4BO gains less than IQS-BO, likely because its prior only shapes the posterior, which expected improvement reduces to a mean and a variance at each point, whereas in IQS-BO the prior shapes the choice of query directly. FIBO is the exception, with a negligible effect, possibly because it compresses the observations into a single averaged context vector, which discards much of the added functional variety. In summary, the benefit of the mixture prior is not specific to IQS-BO, as it also improves PFNs4BO, and it is largest at early evaluations.

4.4 Batch Bayesian Optimisation

In many applications, such as laboratory experiments, several queries are evaluated in parallel, and the batch size can be large (Sin et al., 2025). We evaluate batch sizes $q \in \{10, 20, 50\}$ on seven of the 3D problems, starting from $n_{\text{init}} = 40$ points and running to 200 evaluations. For each batch size, IQS-BO and IQS-BO-PFN are pre-trained on the HEBO prior. Figure 4 reports the GAP after 200 evaluations against the proposal time, the total time spent selecting all queries of a run. Further results can be found in Appendix F.4.

Performance IQS-BO and IQS-BO-PFN reach the highest GAP at every batch size, and their margin over the baselines grows with the batch size. Every baseline loses quality as the batch grows, whereas IQS-BO is almost unaffected, so that at $q = 50$ it is still ahead of the best baseline at $q = 10$. This robustness comes from pre-training at the deployed batch size, where the network learns to account for earlier picks in the batch and to diversify it (Appendix F.5). Our models have the best GAP on every problem except Rosenbrock, where GP-MAP leads as in the sequential setting, and Hartmann at $q = 50$, where GP-FB leads by a negligible margin. The margins are largest on the multimodal problems, Ackley, Levy and Rastrigin.

Computational Cost IQS-BO achieves excellent batch-BO performance at a fraction of the cost of most baselines. Since each point of a batch requires one forward pass, proposing all queries of a run takes 0.9 to 1.2 s, and about 3 s for IQS-BO-PFN, which also runs the frozen PFN. Only FIBO is faster, with 0.3 to 0.6 s, because it samples the points of a batch independently, but it consistently reaches a lower GAP. The GP baselines and PFNs4BO-HEBO, which maximise an acquisition

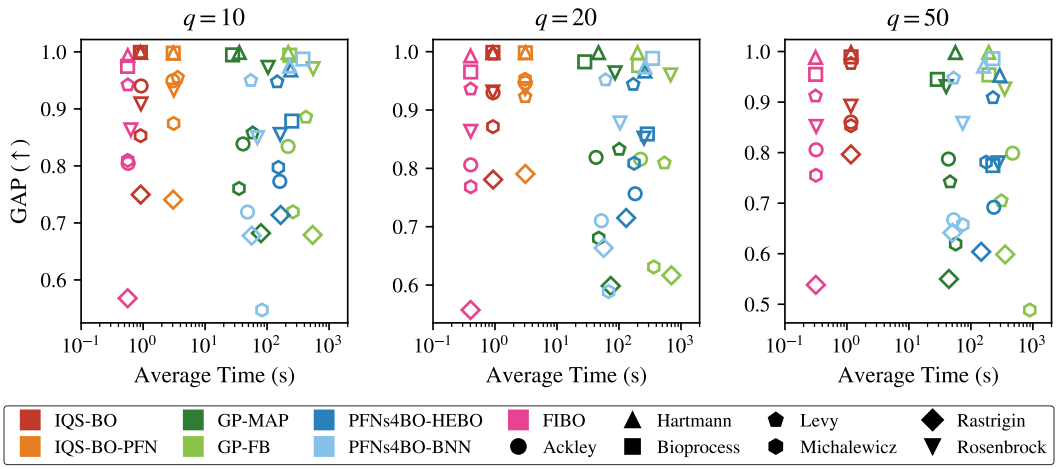


Figure 4: GAP after 200 evaluations against the time spent proposing queries over a complete run, one marker per problem and method, seven 3D problems, $n_{\text{init}} = 40$, mean over 90 seeds. Upper left is better.

function for every batch, are one to three orders of magnitude slower, and IQS-BO is 38 to 68 times faster than GP-MAP. Overall, IQS-BO scales well to large batches, reaching the highest GAP at every batch size while proposing queries more than an order of magnitude faster than GP-based BO.

4.5 Ablation Study

To conclude, we ablate three design choices of IQS-BO, namely the composition and the size of the candidate set and the sequential selection of batch points (Table 2). The composition of the candidate set matters most. The network can only refine the incumbent if candidates lie near it, so replacing the perturbations of the best observations by space-filling candidates degrades performance in most runs. These perturbations also explain why EI-CAND outperforms GP-MAP in Section 4.3. Regarding the size of the candidate set, fewer candidates hurt, whereas more candidates yield only a modest improvement in R^{final} over our default, and none in R^{AUC}. Finally, selecting batch points sequentially, with earlier picks as pending points, outperforms taking the best candidates in a single forward pass, so the network relies on the pending points to improve quality and diversify the batch.

Table 2: Win rate (%) on R^{final} of IQS-BO against each variant, means $\pm 95\%$ CI over ten 3D problems and 900 paired runs.

Variant	n_{init}=10	20	40
space-filling	84.2±2.2	87.9±2.0	88.4±1.8
M=128	57.9±3.1	56.7±3.2	56.6±3.1
M=2048	48.3±3.2	46.4±3.2	50.3±3.2
top-q batch	59.2±3.2	57.9±3.2	55.6±3.2

5 Conclusion

We introduced IQS-BO, a Bayesian optimisation method that selects the next query in-context. Instead of fitting a surrogate and maximising an acquisition function, IQS-BO predicts the posterior probability that each candidate is the best in a set and selects the candidate with the highest probability. It outperforms in-context methods pre-trained on the same prior, matches or outperforms GP-based BO, and in batch BO reaches the highest performance while proposing queries more than an order of magnitude faster than GP-based BO. Finally, we showed that enriching the pre-training prior with functions that stationary GPs model poorly can further improve performance, especially at early evaluations.

Several directions for future work remain. The models are pre-trained for a fixed input dimension and batch size, and our experiments focus on three and four dimensions. Queries are limited to a finite candidate set whose design affects performance, and the objective is myopic, as it optimizes a one-step decision rule. Natural extensions are to train a single model applicable across dimensions and batch sizes, to learn the candidate set jointly with the selection, and to extend the objective beyond one-step selection.

AI USE STATEMENT

No generative AI tools were used as part of the core methodology of this work. The method, the experimental design, the implementation of all models and baselines, the data and the interpretation of results are entirely the work of the authors. Generative AI tools were used as writing assistance, to improve the readability of the text and to format references. This use does not affect the scientific content, methodology or originality of the research. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

All information needed to reproduce our results is given in the paper. Appendix B gives the full architecture and batch-selection rule, Appendix C the pre-training priors, candidate sampler and all training hyperparameters, and Appendix D the configuration of every baseline, including our reimplementation of FIBO. The benchmark problems and their global optima are defined in Appendix E, the evaluation protocol in Section 4.1, and the complete results behind every figure and table in Appendix F. Code is included in the supplementary material.

REFERENCES

Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Alexander I. Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan-Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, and Haitham Bou-Ammar. HEBO: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349, 2022.

Gustavo Sutter Pessurno de Carvalho, Mohammed Abdulrahman, Hao Wang, Sriram Ganapathi Subramanian, Marc St-Aubin, Sharon O'Sullivan, Lawrence Wan, Luis Ricardez-Sandoval, Pascal Poupart, and Agustinus Kristiadi. Simplifying Bayesian optimization via in-context direct optimum sampling. *arXiv preprint arXiv:2505.23913*, 2025.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to parallelize optimization. In Yoel Tenne and Chi-Keong Goh (eds.), *Computational Intelligence in Expensive Optimization Problems*, pp. 131–162. Springer, 2010.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

John Burdon Sanderson Haldane. *Enzymes*. Longmans, Green and Co., London, 1930.

Florian Häse, Matteo Aldeghi, Riley J. Hickman, Loïc M. Roch, Melodie Christensen, Elena Liles, Jason E. Hein, and Alán Aspuru-Guzik. Olympus: A benchmarking framework for noisy optimization and experiment planning. *Machine Learning: Science and Technology*, 2(3):035021, 2021.

Bing-Jing Hsieh, Ping-Chun Hsieh, and Xi Liu. Reinforced few-shot acquisition function learning for Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Daolang Huang, Zhuoyue Huang, Conor Hassan, Luigi Acerbi, Samuel Kaski, and Tom Rainforth. Efficient adaptive data acquisition via pretrained belief representations. *arXiv preprint arXiv:2606.25197*, 2026.

Deng Huang, Theodore T. Allen, William I. Notz, and Ning Zeng. Global optimization of stochastic black-box systems via sequential Kriging meta-models. *Journal of Global Optimization*, 34(3): 441–466, 2006.

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. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Ponnambalam Kumaraswamy. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 46(1–2):79–88, 1980.

Alexandre Maraval, Matthieu Zimmer, Antoine Grosnit, and Haitham Bou-Ammar. End-to-end meta-Bayesian optimisation with transformer neural processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In *International Conference on Machine Learning (ICML)*, 2023.

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.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016.

Joshua W. Sin, Siu Lun Chau, Ryan P. Burwood, Kurt Püntener, Raphael Bigler, and Philippe Schwaller. Highly parallel optimisation of chemical reactions through automation and machine intelligence. *Nature Communications*, 16:6464, 2025.

Jasper Snoek, Kevin Swersky, Richard S. Zemel, and Ryan P. Adams. Input warping for Bayesian optimization of non-stationary functions. In *International Conference on Machine Learning (ICML)*, 2014.

Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets. `https://www.sfu.ca/~ssurjano`, 2013.

Michael Volpp, Lukas P. Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-learning acquisition functions for transfer learning in Bayesian optimization. In *International Conference on Learning Representations (ICLR)*, 2020.

Barnaby E. Walker, James H. Bannock, Adrian M. Nightingale, and John C. deMello. Tuning reaction products by constrained optimisation. *Reaction Chemistry & Engineering*, 2(5):785–798, 2017.

James T. Wilson, Frank Hutter, and Marc Peter Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Xinyu Zhang, Conor Hassan, Julien Martinelli, Daolang Huang, and Samuel Kaski. In-context multi-objective optimization. In *International Conference on Learning Representations (ICLR)*, 2026.

A Comparison to In-Context and Learned Acquisition Methods

This appendix expands the discussion of in-context BO and learned acquisition functions in Section 1. We call a method in-context if, at deployment, the network reads the observations directly and nothing is fitted or updated. Appendix A.1 reviews the in-context methods, Appendix A.2 the learned acquisition methods, which read a refitted GP posterior and are therefore not in-context, and Appendix A.3 relates both to IQS-BO. Table 3 summarises what each method learns, from which signal, and how it proposes the next query.

Table 3: What each method learns, from which signal, and how it proposes the next query.

Method	Learned object	Training signal	Surrogate	Next query
PFNs4BO	posterior predictive	function values, prior	is one	maximise acquisition
FIBO	density over optimum	global optimiser, prior	none	sample
MetaBO, FSAF	candidate score	RL reward, solved tasks	GP, refit	argmax
NAP	candidate score	RL reward, solved tasks	learned jointly	argmax
POLAR	candidate score	RL reward, solved tasks	pretrained	argmax
IQS-BO	posterior over set	best candidate, prior	none or frozen	argmax

A.1 In-Context Methods

Amortised surrogate (PFNs4BO) PFNs4BO (Müller et al., 2023) trains a PFN on synthetic priors to return the posterior predictive $p(y \mid x, \mathcal{D})$ in one forward pass. The surrogate is in-context, but the acquisition function is chosen and maximised numerically at every round, and it is restricted to those with a closed form on the PFN's discrete Riemann distribution (Figure 1).

Amortised acquisition values (PFNs4BO-KG, α-PFN) Two methods train a PFN to predict the value of a specified acquisition function. PFNs4BO reports, in its appendix, a PFN trained to approximate the knowledge-gradient acquisition directly. α-PFN (Rakotoarison et al., 2026) trains a first PFN conditioned on information about the optimum and a second PFN to predict the expected information gain of entropy search from values measured with the first, replacing the Monte Carlo approximations of entropy search by one forward pass per candidate. Both approaches are in-context. In both cases, however, the acquisition rule is chosen in advance, the network regresses its value, and the query is still obtained by maximising that value over $\mathcal{X}$.

Amortised optimiser location (FIBO) FIBO (de Carvalho et al., 2025) trains a normalising flow on pairs $(\mathcal{D}, x^*)$ to model the distribution $p(x^* \mid \mathcal{D})$ of the optimiser location, and proposes queries by sampling from it. This amounts to Thompson sampling under the pre-training prior (Figure 1). The method is fully in-context, but its decision rule is fixed, it cannot be combined with a surrogate, and its batches consist of independent draws. Pre-training is also costly, since every training pair requires the global optimiser of a prior draw, and the rejection sampling used to make x^* uniform over the domain discards most draws (Appendix D.3).

Amortised policies trained by reinforcement learning (NAP, TAMO) In NAP (Maraval et al., 2023), a transformer neural process reads the observations and a candidate set and queries the highest-scoring candidate. It is trained on solved tasks of one family, such as hyperparameter-optimisation datasets for one search space, and transfers only within it. Lacking a label for the query, it learns the score with proximal policy optimisation (PPO) (Schulman et al., 2017), a policy-gradient reinforcement learning method, using the regret as the reward, and compensates for the sparsity of this reward with an auxiliary likelihood head. TAMO (Zhang et al., 2026), a dimension-agnostic transformer for multi-objective optimisation, maps the observations to the next design and is trained by reinforcement learning on hypervolume improvement. Like FIBO, both amortise the BO fully in-context and cannot take a surrogate's information as input.

Learned decision on a pretrained representation (POLAR) POLAR (Huang et al., 2026) trains a small head that scores the candidates from the hidden representations of a pretrained tabular foundation model, TabICLv2 (Qu et al., 2026) by default and TabPFN-2.5 (Grinsztajn et al., 2025) in

an ablation. As in IQS-BO with a fixed surrogate, the representation and the decision are separate modules, but the two methods differ in three respects. First, the head is trained by policy gradient on simulated rollouts with one-step improvements as rewards, so its score approximates no stated quantity, whereas our network approximates the posterior probability of Proposition 1. Second, POLAR is trained on the meta-training tasks of each benchmark search space rather than on a synthetic prior. Third, the backbone is not kept fixed. By default, it is fine-tuned on these tasks with an auxiliary prediction loss, so the gains of POLAR over non-adapted baselines combine the learned decision with a task-specific adaptation of the surrogate. Our network, in contrast, is pre-trained only on synthetic priors and either keeps the surrogate frozen or can be trained without one. We do not include POLAR as a baseline, since no public implementation was available at the time of writing.

A.2 Learned Acquisition Methods

MetaBO, FSAF and BOFormer also learn the query decision, but by reinforcement learning on top of a Gaussian process. At every round, a GP is refitted to the observations, and a network selects a point from a finite candidate set using only summaries of the GP posterior at each candidate, such as its mean and variance. The network is trained to maximise a reward based on the regret or, for multi-objective problems, the hypervolume. Since it never reads the observations, all information about a new task reaches it through the GP fit, so none of these methods is in-context.

MetaBO (Volpp et al., 2020) MetaBO replaces only the acquisition function. A neural network scores each candidate from its location, the GP posterior mean and variance and the remaining budget, and the policy is a softmax over these scores on Sobol points augmented with local maxima of the score. The network is meta-trained with PPO on related source tasks, so that it encodes the structure shared across this family.

FSAF (Hsieh et al., 2021) FSAF treats the acquisition function as a Q-function, parameterised by a Bayesian deep Q-network on features of the GP posterior. It is meta-trained with Bayesian model-agnostic meta-learning on a set of training functions, with a demonstration policy based on a standard acquisition function as a prior, and adapted to each new task by a few gradient steps on its first observations. FSAF therefore requires both a GP fit at every round and gradient updates on every new task.

BOFormer (Hung et al., 2025) BOFormer addresses multi-objective BO and learns the acquisition function as a Q-function over sequences by non-Markovian deep Q-learning, reading the GP posterior rather than the observations.

A.3 Relation to Our Method

Like the in-context methods of Appendix A.1, IQS-BO reads the observations directly, and like the learned acquisition methods of Appendix A.2, it selects the next query from a finite candidate set. It differs from all of them in three respects: how the decision is learned, what it is trained on, and how a surrogate can be used.

First, the decision is learned by supervised classification. Under a synthetic prior, every function can be evaluated at all candidates, so the index of the best candidate is available as a label for every task. The network is trained with cross-entropy on this label, whereas NAP, POLAR, MetaBO, FSAF and BOFormer learn from a reward, and PFNs4BO-KG and α-PFN regress a specified acquisition function. Second, training requires only a prior over functions, not a pool of solved tasks as in NAP and POLAR. A trained model therefore applies to any problem of its input dimension and batch size, without meta-training on the target family or adaptation at test time. Third, the same network and loss can be used with or without a fixed surrogate. With a surrogate, only the query selection is amortised, and without one, the whole BO loop runs in-context.

B Model Details

This appendix details the model of Section 3: the token inputs and attention structure, the network and deployment settings, and the sequential selection of batch points, with its related extended training loss and algorithms.

B.1 Tokens and Inputs

Surrogate-free model, single query Without a surrogate, IQS-BO reads one token per observation and one per candidate. Every token carries the point channels of Table 4, and observation tokens also carry the value channels. Each token s is embedded as $h_s^{(0)} = W_{\text{feat}}\, t_s$. Observation tokens attend only to each other, and candidate tokens only to the observations. The features of a candidate are carried forward by the residual connections of every layer ℓ,

$$\tilde{h}_s^{(\ell)} = \mathrm{LN}\big(h_s^{(\ell)} + \mathrm{Attn}(h^{(\ell)})_s\big), \qquad h_s^{(\ell+1)} = \mathrm{LN}\big(\tilde{h}_s^{(\ell)} + \mathrm{FF}(\tilde{h}_s^{(\ell)})\big),$$

and the head maps the final representation of c_j to z_j. The score therefore depends only on $(\mathcal{D}, c_j)$ and is unchanged by adding, removing or permuting other candidates.

Table 4: Token channels. Statistics are over the n observed values, and x^+ is the incumbent. Channels are zero on tokens that do not carry them. The noise channel holds the task's noise level in pre-training and a fixed σ_0 at deployment (Appendix B.2).

channel	width	carried by	definition
coordinates	d	all tokens	$x \in [0,1]^d$
distances	2	all tokens	$\min_i \lVert x - x_i \rVert_2$ and $\lVert x - x^+ \rVert_2$
role	4	all tokens	one-hot token type
state	2	all tokens	$1/\sqrt{n}$, $\log_{10} \sigma_{\text{noise}}$
value	1	observations, filled pending	$(y_i - \bar{y})/s_y$ with $s_y \geq 10^{-3}$, or fantasy $\hat{y}$
rank	1	observations	$\Phi^{-1}\big((r_i + \frac{1}{2})/n\big)$
gap	1	observations	$\tilde{y}_i - \max_k \tilde{y}_k$
incumbent	1	observations	$\mathbf{1}[x_i = x^+]$

With a surrogate IQS-BO-PFN and IQS-BO-GP use the same tokens, channels and attention mask of IQS-BO, and only the embedding changes to $h_s^{(0)} = W_{\text{feat}}\, t_s + W_\phi\, m_s$, where m_s holds the surrogate features of token s (Table 5). The surrogate receives only the coordinates and values of the observations and is evaluated before the forward pass of the network. The PFN is run on the context-standardised values, without the power transform that PFNs4BO applies to y, and its predictive summaries are not standardised across candidates, since this would make each score depend on the other candidates.

Table 5: Surrogate features m_s. The surrogate is conditioned on the observations and on the filled pending points at their fantasy values. Empty pending tokens receive zeros.

model	feature	width	carried by	definition
IQS-BO-PFN	layer states	512	all tokens	$\sum_\ell \mathrm{softmax}(\alpha)_\ell\, \mathrm{state}_{\ell,s}$, learned $\alpha \in \mathbb{R}^{12}$
	summaries	5	candidates	mean, 68% half-width, 10/50/90% quantiles
IQS-BO-GP	mean	1	all tokens	posterior mean $\mu(x)$
	std. dev.	1	all tokens	latent posterior $\sigma(x)$
	correlation	1	all tokens	$\mathrm{corr}\,\big(f(x), f(x^+)\big)$
	hyperparameters	$d+2$	all tokens	log-lengthscales, log-noise, constant mean

Batch selection For a batch of q points, every model also carries $P = q - 1$ pending tokens, which hold the points already selected in the current round (Appendix B.3). An empty pending token holds the coordinates of the incumbent and no value. Once a point is selected, the next pending token takes its coordinates and, in its standardised-value channel, a fantasy value $\hat{y}$. For IQS-BO, $\hat{y} = 0$, the mean of the standardised context, and for IQS-BO-PFN and IQS-BO-GP, $\hat{y}$ is the surrogate's

posterior mean. Observation and pending tokens attend only to each other, and candidate tokens attend to both, so the score of a candidate becomes $z_j(\mathcal{D}, \mathcal{P}, c_j)$, with $\mathcal{P}$ the pending points.

With a surrogate, the filled pending points are also passed to the surrogate at their approximated values, and empty pending tokens receive zero surrogate features. The GP is conditioned on the observations and the filled pending points, with the hyperparameters of the round and without refitting. The PFN receives the filled pending points as additional context points.

B.2 Network and Deployment

The encoder of Section 3.3 is a post-norm transformer with 4 layers of width 512, 8 attention heads, feed-forward width 2048, GELU activations and no dropout. The scoring head consists of Linear$(512, 512)$, GELU and Linear$(512, 1)$. The network has 12.88M trainable parameters for IQS-BO, 12.89M for IQS-BO-GP and 13.15M for IQS-BO-PFN, the latter excluding the 26.8M parameters of the frozen PFN. At deployment, every run uses $M = 512$ candidates. The noise channel is set to the low end of each prior's noise range, $\sigma_0 = 10^{-3}$ for models pre-trained on the mixture prior and $\sigma_0 = 0.025$ for models pre-trained on the HEBO prior.

B.3 Batch Selection

The points of a batch are selected one at a time through the pending tokens of Appendix B.1. For $q = 2$, the loss of one task is

$$\mathcal{L} = \mathrm{CE}\big(z^{(0)}, J^*\big) + \mathbf{1}[j_0 \neq J^*]\ \mathrm{CE}\big(z^{(1)}, J^*\big), \qquad j_0 \sim \mathrm{softmax}(z^{(0)}), \tag{7}$$

where $z^{(k)}$ are the scores at slot k. The first pick j_0 is sampled without gradient and written into a pending token, and the same candidates are scored again. The second term applies only if j_0 is not the best candidate, so by Proposition 1 it is minimised by the posterior probability that each candidate is the best, given that j_0 is not. This mask is what teaches the network to diversify the batch. For $q > 2$, the loss of each slot is counted only if no earlier pick is the best candidate, and the losses of $\min(q, 4)$ random slots are averaged (Algorithm 3). At deployment, a fresh candidate set is drawn for every slot (Algorithm 4). Both algorithms reduce to Algorithms 1 and 2 for $q = 1$. The models of the batch study in Section 4.4 are pre-trained on the HEBO prior.

Algorithm 3 Pre-training on one task (surrogate ϕ or $\varnothing$, batch size q)

Require: prior $p(f)$, surrogate ϕ or $\varnothing$, network θ, candidate-set size M, batch size q

Sample $f \sim p(f)$ and a context $\mathcal{D}$ from f
Draw $\mathcal{C} = \textsc{CandidateSet}(\mathcal{D}, M)$ ▷ Appendix C.2
Compute the label $J^* = \arg\max_{j \le M} f(c_j)$ ▷ noise-free f
If ϕ is a GP, fit its hyperparameters to $\mathcal{D}$ by MAP
Draw $G \subseteq \{0, \dots, q-1\}$ with $|G| = \min(q, 4)$ ▷ slots with a gradient
$\mathcal{P} \leftarrow \emptyset$, $\mathcal{L} \leftarrow 0$, $m \leftarrow 1$
for $k = 0, \dots, \max G$ **do**
    Compute $p_\theta^{(k)}(\cdot \mid \mathcal{D}, \mathcal{P}, \mathcal{C})$ in one forward pass
    if $k \in G$ **then**
        $\mathcal{L} \leftarrow \mathcal{L} - m \log p_\theta^{(k)}(J^* \mid \mathcal{D}, \mathcal{P}, \mathcal{C})$
    end if
    Sample $j_k \sim p_\theta^{(k)}(\cdot \mid \mathcal{D}, \mathcal{P}, \mathcal{C})$ without gradient, and set $m \leftarrow m\, \mathbf{1}[j_k \neq J^*]$
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{(c_{j_k}, \hat{y}(\phi, c_{j_k}))\}$ ▷ kriging believer $\hat{y}$
end for
return the task loss $\mathcal{L}/|G|$ ▷ averaged over a mini-batch of tasks

C Pre-Training Details

This appendix shows how the pre-training tasks of Section 3.1 are generated, and provides details for the two pre-training priors (Appendix C.1), the candidate set sampler (Appendix C.2), pre-training settings and computational costs (Appendix C.3).

Algorithm 4 One optimisation round (surrogate ϕ or $\varnothing$, batch size q)

Require: observations $\mathcal{D}$, surrogate ϕ or $\varnothing$, network θ, candidate-set size M, batch size q

If ϕ is a GP, fit its hyperparameters to $\mathcal{D}$ by MAP
$\mathcal{P} \leftarrow \emptyset$
for $k = 0, \dots, q-1$ **do**
  Draw $\mathcal{C}_k = \textsc{CandidateSet}(\mathcal{D}, M)$
  Compute $p_\theta^{(k)}(\cdot \mid \mathcal{D}, \mathcal{P}, \mathcal{C}_k)$ in one forward pass
  $x_k \leftarrow c_{\hat{\jmath}} \in \mathcal{C}_k$ with $\hat{\jmath} = \arg\max_{j \le M} p_\theta^{(k)}(j \mid \mathcal{D}, \mathcal{P}, \mathcal{C}_k)$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(x_k, \hat{y}(\phi, x_k))\}$ ▷ kriging believer $\hat{y}$
end for
Evaluate f at $x_0, \dots, x_{q-1}$, observe $y_0, \dots, y_{q-1}$, and add them to $\mathcal{D}$

C.1 Priors

HEBO prior Müller et al. (2023) propose several pre-training priors for PFNs4BO, among them a GP-based prior inspired by the surrogate of HEBO (Cowen-Rivers et al., 2022). We pre-train on the HEBO prior, with the kernel and hyperpriors of the released PFNs4BO-HEBO model (Table 6). A GP path cannot be drawn on a fixed grid in advance, since the context points are chosen adaptively from earlier values. The value of f at each new point, including the candidates for the label, is therefore drawn from the GP conditioned on all values drawn so far. This gives the same joint distribution as a single draw of f at all these points.

Table 6: HEBO prior. Functions are drawn as $f \sim \mathcal{GP}(0, s^2 k_{3/2})$, with $k_{3/2}$ the Matérn-3/2 ARD kernel with lengthscales ℓ. Gamma distributions are given by shape and rate.

lengthscales	$\ell_i \sim \mathrm{Gamma}(1.2107, 1.5212)$
output scale	$s^2 \sim \mathrm{Gamma}(0.8452, 0.3994)$
noise	$\sigma^2 \sim \mathrm{LogNormal}(-4.63, 0.5)$, with relative noise $\sqrt{\sigma^2/s^2}$ clamped to $[10^{-3}, 10]$
active dimensions	each dimension dropped with probability 0.3, keeping at least one

Mixture prior The mixture prior extends pre-training to functions that stationary GP surrogates model poorly. Each function is drawn from one of four families, with the weights of Table 7. The stationary family covers smooth functions, on which GP surrogates are well specified. The warped family applies a monotone transformation to each input, so that the effective lengthscale varies across the domain (Snoek et al., 2014). The bumps family sums a few Gaussian basins, which produces narrow optima within nearly flat regions. The plateau family applies steep sigmoids to a stationary path, which produces piecewise-flat functions with sharp steps, such as those caused by thresholds or regime changes in physical processes.

The stationary, warped and plateau families are built on the random-Fourier-feature approximation of a GP path (Rahimi & Recht, 2007), as in the pre-training prior of FIBO (de Carvalho et al., 2025), which we extend from the squared-exponential kernel to Matérn-ν kernels,

$$\mathrm{rff}(x) = \sqrt{2/m} \sum_{k=1}^{m} a_k \cos(\omega_k^\top x + b_k), \qquad m = 256,$$

where $a_k \sim \mathcal{N}(0,1)$, $b_k \sim \mathcal{U}(0, 2\pi)$, ω_k is drawn from the Matérn-ν spectral density with $\nu \sim \mathcal{U}\{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \infty\}$, and the lengthscales follow the HEBO prior, floored at 0.02. Every function is standardised over a 1024-point Sobol grid, and observations carry Gaussian noise with $\sigma \sim \mathrm{LogU}(10^{-3}, 0.1)$.

C.2 Candidate Sampling

Candidates are drawn by Algorithm 5, identically in pre-training and at deployment. They consist of 40% scrambled Sobol points, 30% uniform points and 30% Gaussian perturbations of the three best observations, clipped to $[0,1]^d$ by the projection Π. The space-filling points support exploration, and the perturbations allow the network to refine the incumbent. Their scale is drawn log-uniformly over an order of magnitude, which covers both coarse and fine moves without a fixed trust-region radius.

Table 7: Definition of the mixture prior. The warped family uses Kumaraswamy CDFs κ_i (Kumaraswamy, 1980; Snoek et al., 2014), with α_i, β_i clamped to $[0.2, 5]$. In the bumps family, the division by w_k is elementwise, and sigm denotes the logistic function.

family	weight	definition
stationary	0.3	rff
warped	≈ 0.233	$\text{rff} \circ \kappa,\ \kappa_i(x) = 1 - (1 - x^{\beta_i})^{\alpha_i}$
		$\alpha_i, \beta_i \sim \text{LogNormal}(0, 0.75)$
bumps	≈ 0.233	$\sum_{k \le K} h_k \exp\big(-\frac{1}{2}\|(x - c_k)/w_k\|^2\big)$
		$K \sim \mathcal{U}\{2, \dots, 10\},\ c_k \sim \mathcal{U}[0,1]^d,\ w_{ki} \sim \text{LogU}(0.03, 2)$
plateau	≈ 0.233	$\sum_{j \le L} \text{sigm}\big((\text{rff} - t_j)/\tau\big) + 10^{-3}\,\text{rff}$
		$L \sim \mathcal{U}\{2, \dots, 7\},\ \tau \sim \text{LogU}(0.01, 0.1)$

Pre-training draws $M \in \{64, 128, \dots, 384\}$, while all experiments use $M = 512$. This is possible because each score is independent of the other candidates (Appendix B.1).

Algorithm 5 CANDIDATESET$(\mathcal{D}, M)$

$S \leftarrow 0.4M$ scrambled-Sobol points in $[0,1]^d$
$U \leftarrow 0.3M$ uniform points in $[0,1]^d$
$A \leftarrow$ the inputs of the three observations in $\mathcal{D}$ with the largest y
for $t = 1, \dots, 0.3M$ **do**
    Draw $a \sim \mathcal{U}(A)$, $s \sim \text{LogU}(0.02, 0.2)$ and $\xi \sim \mathcal{N}(0, I_d)$, and set $l_t \leftarrow \Pi(a + s\,\xi)$
end for
return $S \cup U \cup \{l_t\}_{t \le 0.3M}$

C.3 Training Configurations and Cost

All IQS-BO models share the settings of Table 8, and all models were trained on a single NVIDIA H100 GPU. PFNs4BO, FIBO and NAP use their own recipes (Appendix D). Table 9 reports the training time as the median time per epoch, excluding the first epoch, multiplied by the number of epochs, without evaluation and data generation. For NAP, which is trained by PPO rather than in epochs, we report the time of the full training run, and its 3D and 4D models use different configurations (Appendix D.4), which explains the difference in their training times.

Table 8: Pre-training settings.

setting	value
optimiser	AdamW
learning rate	10^{-4}, cosine, 5 warm-up epochs
weight decay	10^{-2}
gradient clipping	1.0
epochs	100
steps per epoch	1024
batch size	128 (2 micro-batches of 64)
optimiser steps	51,200

Table 9: Training time (H100 hours), $q = 2$.

model	prior	$d = 3$	$d = 4$
IQS-BO	HEBO	2.73	2.74
	mixture	4.37	4.28
IQS-BO-PFN	HEBO	4.63	4.59
	mixture	6.06	6.10
IQS-BO-GP	HEBO	2.65	2.66
	mixture	2.65	2.67
PFNs4BO	mixture	0.74	0.74
FIBO	GP prior	1.78	1.94
	mixture	1.94	2.00
NAP	GP prior	6.77	11.1

D Baselines

D.1 GP-MAP and GP-FB

Both GP baselines are implemented in BoTorch 0.18.1 (Balandat et al., 2020), in double precision. They use a Matérn-5/2 ARD kernel and a homoscedastic Gaussian likelihood, with inputs in the unit cube and standardised outputs. At every round, the next batch is chosen by maximising

qLogNoisyExpectedImprovement (Ament et al., 2023), estimated with 256 quasi-Monte Carlo samples, jointly over the q points with optimize_acqf, using 512 raw samples and 10 L-BFGS-B restarts. The two baselines differ only in how they handle the hyperparameters. GP-MAP uses SingleTaskGP with BoTorch's dimension-scaled log-normal hyperpriors and fits the hyperparameters by MAP at every round. GP-FB uses FullyBayesianSingleTaskGP, samples the hyperparameter posterior with NUTS (256 warm-up steps, 8 retained samples), and averages the acquisition function over these samples.

D.2 PFNs4BO

We use the two checkpoints released by Müller et al. (2023).[1] PFNs4BO-HEBO (12 layers) is pre-trained on the HEBO prior (Appendix C.1), and PFNs4BO-BNN (6 layers) on a prior of random Bayesian neural networks with warped inputs. At every round, the observed values are Yeo–Johnson transformed, and expected improvement is computed in closed form from the predictive distribution and maximised with the authors' optimiser, which refines the best 100 of $100{,}000$ uniform samples by L-BFGS-B. The released code proposes one point per round and does not support batch optimisation, so we build batches with the kriging-believer rule (Ginsbourger et al., 2010). Each selected point is added to the observations with its predictive mean as value, and the next point maximises expected improvement on the augmented observations.

D.3 FIBO

FIBO (de Carvalho et al., 2025) pre-trains a conditional normalising flow $p_\vartheta(x^* \mid \mathcal{D})$ of the optimiser location on pairs $(\mathcal{D}, x^*)$ from a GP prior, and proposes a batch as q independent samples. No implementation is public, so we reimplemented it following the specifications mentioned in the paper. One model is trained per dimension. The encoder is the PFNs4BO transformer pre-trained on the BNN prior, whose outputs are averaged over the data points and projected to a context vector of dimension 256 ($d = 3$) or 512 ($d = 4$). The generative head is a normalising flow, which maps a simple base distribution through a sequence of invertible transformations to a density over the domain, conditioned on the context vector. Following the paper, it is an autoregressive neural spline flow (Durkan et al., 2019), in which each block transforms one coordinate at a time with a monotone rational-quadratic spline whose parameters depend on the previous coordinates and on the context vector. It has 6 ($d = 3$) or 8 ($d = 4$) blocks, each with 6 layers of 256 hidden units. Functions are random-Fourier-feature paths of a squared-exponential GP with the paper's hyperpriors, the optimum is located by multi-start L-BFGS-B, and rejection sampling makes the marginal of x^* uniform. Training uses $70{,}000$ pairs, Adam with learning rate 10^{-4}, a cosine schedule, 200 epochs and batch size 128. In our runs, rejection sampling discards about 99% of the draws, with 5.6M optimisations at $d = 3$ and 18.2M at $d = 4$.

Three choices depart from the paper, each intended to favour FIBO. The base distribution of the flow is uniform on the domain, so that samples are always feasible, the observed values are standardised over the context instead of Yeo–Johnson transformed, and the context sizes cover those of our protocols. On functions drawn from its prior at $d = 3$, our implementation reaches a GAP of 0.90 after 58 evaluations, against 0.93 after 200 in the paper, and it beats random search on every test function.

D.4 NAP

NAP (Maraval et al., 2023) is a transformer neural process that reads the observations and a candidate set, scores every candidate and queries the one with the highest score. We use the authors' code,[2] with 6 layers of width 512 and a bar-distribution head with 1000 buckets, and, since it provides no training script for continuous functions, we add one that changes only the task environment. NAP is meta-trained with PPO for 2000 iterations on the GP task family of MetaBO (Volpp et al., 2020), that is, on Matérn-5/2 GP draws with lengthscales in $[0.05, 0.5]$, unit signal variance and noise variance 0.1, with a regret-based reward and an auxiliary loss that fits the predictive head to the observed values. In 4D, we use the published configuration, with episodes of 30 steps without initial points and

[1] https://github.com/automl/PFNs4BO

[2] https://github.com/huawei-noah/HEBO/tree/master/NAP

2000 Sobol candidates per query. In 3D, we adapt it to our protocol, with episodes of 90 steps from 10 initial points, observed values standardised within the context and the candidate sets of the other candidate-based methods (Appendix C.2). NAP skips candidates that coincide with an observed or already selected point, and for $q = 2$ it imputes the value of its first pick with the mean of its own predictive head, adds it to the context and then selects the second point.

E Benchmark Problems

E.1 Problems

Table 10 lists the ten 3D and five 4D problems. Every problem is mapped to the unit cube and maximised, with minimisation problems negated.

Analytic functions The analytic functions use the implementations of BoTorch 0.18.1 (Balandat et al., 2020),[3] whose definitions follow the Virtual Library of Simulation Experiments (Surjanovic & Bingham, 2013). They cover needle-in-a-haystack landscapes (Ackley), dense ripples on a global trend (Levy, Rastrigin), narrow ridges on a plateau (Michalewicz), an ill-conditioned valley (Rosenbrock) and smooth surfaces with few optima (Hartmann, Six-Hump Camel). Rosenbrock uses the box $[-2.048, 2.048]^d$ instead of BoTorch's default $[-5, 10]^d$. The two-dimensional Six-Hump Camel function is embedded in 3D through a third coordinate that does not affect f.

Bioprocess surface This surface, constructed for this work, models the tuning of a microbial fuel cell over three concentrations with Haldane substrate-inhibition kinetics (Haldane, 1930),

$$f(x) = h(x_1; p_1)\, h\big(x_2; p_2(x_3)\big)\, h(x_3; p_3), \qquad p_2(x_3) = p_2\,\big(1 + 0.8\,(x_3 - 0.5)\big),$$

where $h(\cdot; p)$ is the Haldane rate $v(s) = s/(K + s + s^2/K_i)$ normalised to one at its peak p, with $(p_1, p_2, p_3) = (0.10, 0.20, 0.35)$ and $\sqrt{K/K_i} = 6$. The dependence of p_2 on x_3 makes it non-separable. It is unimodal but strongly non-stationary, and about 0.3% of the domain lies within 10% of the optimum $f^* = 1$ at $x^* = (0.10, 0.176, 0.35)$.

Olympus campaigns The real-world campaigns come from Olympus (Häse et al., 2021),[4] whose neural-network emulators, trained on data from automated experiments, serve as oracles. We use the deterministic `NeuralNet` emulators, so the optimum can be pinned and regret computed exactly. Since the package requires TensorFlow 1.15, we extracted the emulator weights from `olymp` 0.0.1b0 and reimplemented the forward pass in PyTorch, which reproduces the stored test R^2 of the 3D emulators to seven significant figures. *Fullerenes* maximises the mole fraction of a bis-adduct in a flow reactor, on data from Walker et al. (2017), and *colours* (`colors_n9`) minimises the distance of a mixed colour to a target green (Table 12). *Benzylation* minimises the impurity of an N-benzylation reaction in 4D. Its response surface is smooth, which makes it well suited to GP modelling.

GP-prior draws The 4D GP-prior problem consists of ten functions drawn from our reimplementation of the GP prior of FIBO, a random-Fourier-feature squared-exponential GP (Appendix D.3). Each is optimised as a separate problem, and their results are averaged into one. They are problems on which GP surrogates are well specified by construction.

Table 10: Benchmark problems. Minimisation problems are negated.

problem	d	native domain	source	problem	d	native domain	source
Ackley	3, 4	$[-32.768, 32.768]^d$	BoTorch	Six-Hump Camel	3	$[-3, 3] \times [-2, 2]$	BoTorch
Levy	3, 4	$[-10, 10]^d$	BoTorch	bioprocess	3	$[0, 1]^3$	this work
Rosenbrock	3, 4	$[-2.048, 2.048]^d$	BoTorch	fullerenes	3	emulator	Olympus
Rastrigin	3	$[-5.12, 5.12]^3$	BoTorch	colours	3	emulator	Olympus
Michalewicz	3	$[0, \pi]^3$	BoTorch	benzylation	4	emulator	Olympus
Hartmann	3	$[0, 1]^3$	BoTorch	GP-prior draws	4	$[0, 1]^4$	FIBO prior

[3] `https://github.com/pytorch/botorch`, module `botorch.test_functions`.

[4] `https://github.com/aspuru-guzik-group/olympus`, package `olymp` on PyPI.

E.2 Optima

Regret is computed against recomputed global optima, since rounded tabulated constants, such as that of Hartmann in BoTorch, which lies 2×10^{-6} below the true maximum, would put a floor under the regret. Closed-form optima are used exactly, and all others were found by multi-start L-BFGS-B and confirmed by an independent second search (Tables 11 and 12, on the unit cube). In 4D, Ackley, Levy and Rosenbrock have their optima at the same coordinates as in 3D, and benzylation at the corner $x^* = (1, 0, 0, 0)$, with $f^* = -2.33504673$. The GP-prior draws are filtered as in the pre-training of FIBO, so that their optima lie inside the cube and are spread uniformly, use seeds outside those of FIBO's training data, and have their optima checked against 2×10^6 Sobol points.

Table 11: Analytic functions in 3D, with $\cdot$ any value of the inert coordinate.

problem	x^*	f^*
Ackley	$(0.5, 0.5, 0.5)$	0
Levy	$(0.55, 0.55, 0.55)$	0
Rastrigin	$(0.5, 0.5, 0.5)$	0
Rosenbrock	$(0.7441, 0.7441, 0.7441)$	0
Michalewicz	$(0.7012, 0.5, 0.4090)$	2.76039468
Hartmann	$(0.1146, 0.5556, 0.8525)$	3.86278
Six-Hump Camel	$(0.515, 0.322, \cdot)$ $(0.485, 0.678, \cdot)$	1.03162845

Table 12: Olympus campaigns in 3D, with n measurements and test R^2 per emulator.

	fullerenes	colours
n	246	102
R^2	0.990	0.940
x^*	$(0.2693, 0.4361, 0.0513)$	$(0, 1, 0.8075)$
f^*	0.95243518	-0.06969733

F Additional Results and Ablations

This appendix reports more detailed results behind Section 4, including per-problem results, pairwise win rates and results at every context size, and adds analyses not shown in the main text. It covers the 3D and 4D problems (Appendix F.1), the comparison with expected improvement (Appendix F.2), the effect of the pre-training prior (Appendix F.3), batch optimisation (Appendix F.4) and additional ablations (Appendix F.5).

F.1 Additional Results on the 3D and 4D Problems

This section is an extension of Section 4.2. For the 3D problems, Figures 5 and 6 repeat Figure 2 at $n_{\text{init}} = 10$ and 20, and Table 13 complements the ranks of Table 1 with the paired win rate of every IQS-BO variant against every baseline. For the five 4D problems (Appendix E), whose ranks appear in Table 1, Figure 8 shows the regret and rank curves. For both dimensions, Figures 7 and 9 and Table 14 give the results per problem.

Smaller context sizes and paired win rates At $n_{\text{init}} = 10$ and 20, the ordering of the methods is the same as at $n_{\text{init}} = 40$. Pre-trained on the mixture prior, each IQS-BO variant beats GP-MAP and GP-FB in about 80% of runs and every in-context baseline in more than 83%. Pre-trained on the HEBO prior, IQS-BO and IQS-BO-GP still beat GP-MAP in 65 to 75% of runs, whereas IQS-BO-PFN ties with it at $n_{\text{init}} = 10$ and only pulls ahead with more context.

Per-problem results In 3D, the gains of IQS-BO are largest on Ackley, Camel, Levy and the Olympus colours campaign, where it reaches a lower regret than all baselines early in the run and keeps it until the end (Figure 7). The exception is Rosenbrock, a smooth unimodal valley on which a Matérn GP is well specified, and where the GP baselines end with lower regret. In 4D, the IQS-BO variants pre-trained on the mixture prior again reach a lower final regret than GP-MAP on Ackley, Levy and Rosenbrock (Table 14, Figure 9). The GP-prior draws and benzylation, which were included as problems on which a GP is well specified, favour the GP baselines, for two different reasons. On the GP-prior draws, the objective is a GP sample, so a GP surrogate can approximate it almost exactly by construction. IQS-BO-GP, which conditions on the same GP, comes closest to GP-MAP and improves on it in 55 to 66% of paired runs, whereas the variants without a GP must infer this structure from the observations alone, and perform worse. On benzylation, the response surface is smooth, and gradient-based maximisation of the acquisition function lets GP-MAP and GP-FB place a query

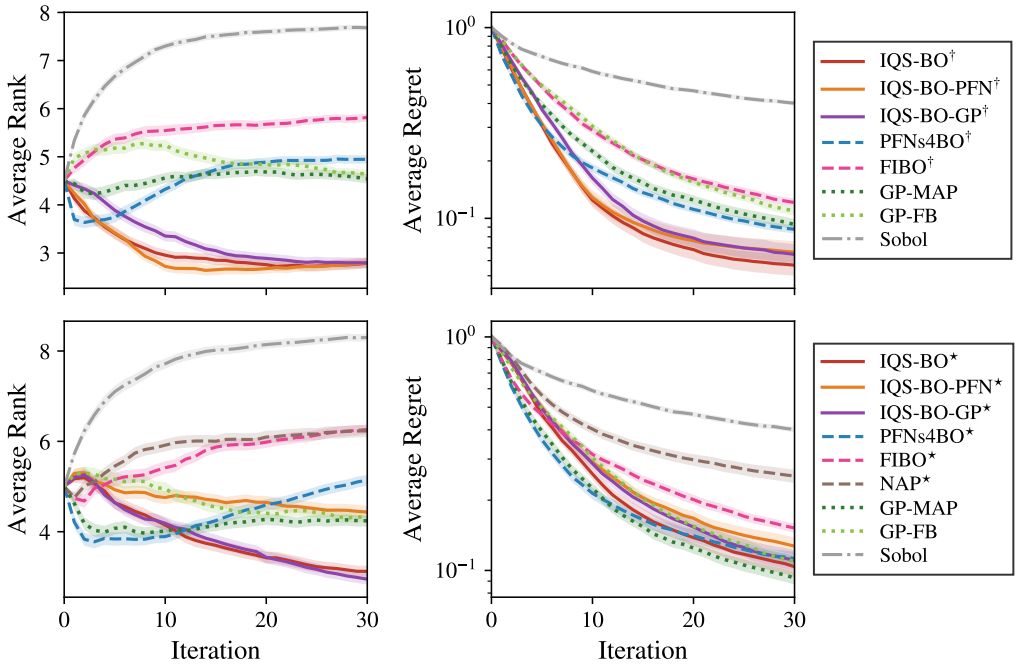


Figure 5: As Figure 2, at $n_{\text{init}} = 10$.

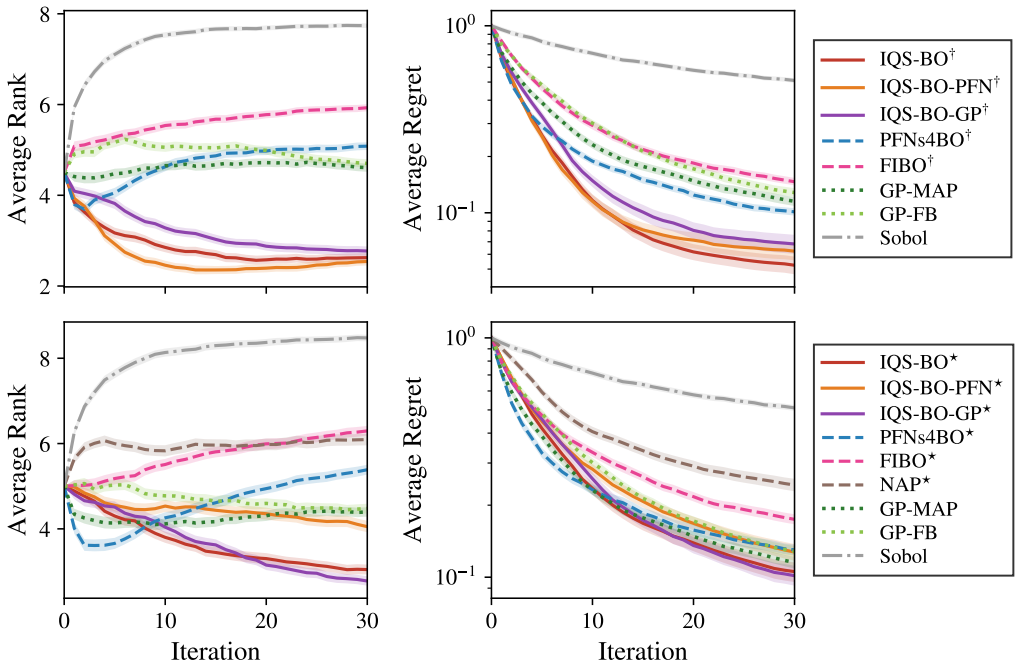


Figure 6: As Figure 2, at $n_{\text{init}} = 20$.

exactly at the optimum, whereas the regret of IQS-BO and all other in baselines stalls at a small non-zero value once it has found the optimum region.

F.2 Learned Query Selection Against Expected Improvement

Table 15 provides the paired win rates behind Section 4.3 and adds two comparisons.

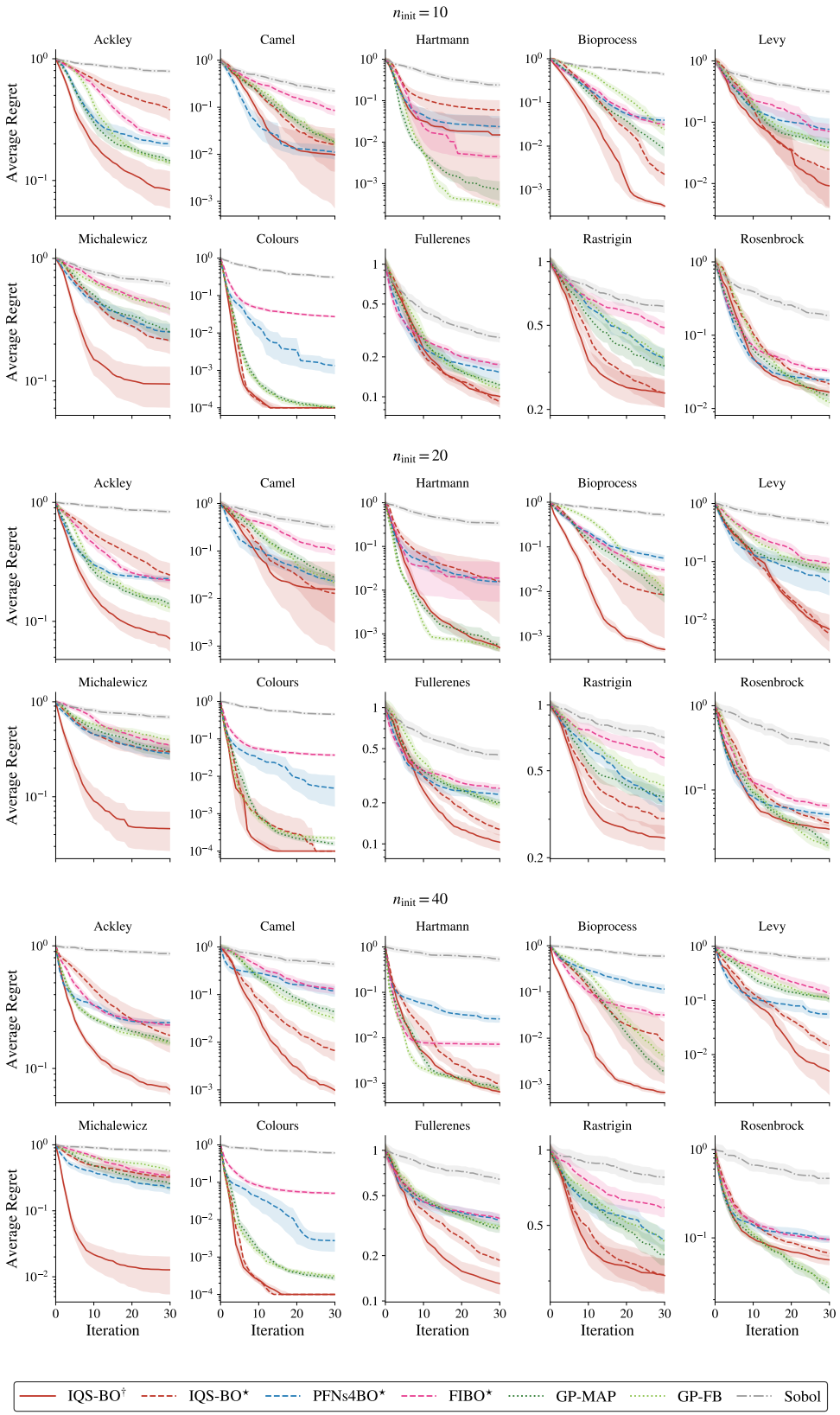


Figure 7: Average regret per iteration on each 3D problem, $n_{\text{init}} = 10$ (top), 20 (middle) and 40 (bottom), 90 seeds, with 95% confidence intervals.

Table 13: Paired win rate (%) on R^{final} of each IQS-BO variant against each baseline, ten 3D problems, 900 paired runs per cell, $\pm$ 95% bootstrap CI half-width. † mixture prior, * HEBO prior.

	model	GP-MAP	GP-FB	PFNs4BO-HEBO	PFNs4BO-BNN	FIBO	Sobol
$n_{\text{init}} = 10$	IQS-BO†	77.4$\pm$2.4	77.3$\pm$2.4	86.4$\pm$2.1	85.1$\pm$2.2	92.7$\pm$1.7	96.4$\pm$1.2
	IQS-BO-PFN†	76.1$\pm$2.6	77.3$\pm$2.4	86.3$\pm$2.2	85.7$\pm$2.2	90.9$\pm$1.9	95.4$\pm$1.4
	IQS-BO-GP†	77.6$\pm$2.6	78.8$\pm$2.4	85.7$\pm$2.1	83.9$\pm$2.2	91.0$\pm$1.8	96.3$\pm$1.2
	IQS-BO*	65.2$\pm$2.7	67.4$\pm$2.7	75.3$\pm$2.7	75.4$\pm$2.8	83.6$\pm$2.2	94.2$\pm$1.5
	IQS-BO-PFN*	49.7$\pm$3.1	51.0$\pm$3.0	60.4$\pm$2.9	61.8$\pm$3.1	72.0$\pm$2.8	90.9$\pm$1.8
	IQS-BO-GP*	71.0$\pm$2.8	71.6$\pm$2.7	77.0$\pm$2.6	76.7$\pm$2.6	85.3$\pm$2.1	93.7$\pm$1.6
$n_{\text{init}} = 20$	IQS-BO†	79.7$\pm$2.2	80.3$\pm$2.2	90.1$\pm$1.8	86.4$\pm$2.2	94.4$\pm$1.4	97.7$\pm$0.9
	IQS-BO-PFN†	81.6$\pm$2.3	81.6$\pm$2.2	90.7$\pm$1.7	86.1$\pm$2.1	94.6$\pm$1.4	97.6$\pm$0.9
	IQS-BO-GP†	78.3$\pm$2.4	79.9$\pm$2.3	86.7$\pm$2.1	83.1$\pm$2.2	92.4$\pm$1.7	98.0$\pm$0.9
	IQS-BO*	69.6$\pm$2.6	70.1$\pm$2.6	79.5$\pm$2.5	75.6$\pm$2.7	85.2$\pm$2.2	95.8$\pm$1.3
	IQS-BO-PFN*	57.3$\pm$3.2	57.1$\pm$3.1	67.2$\pm$2.8	62.6$\pm$2.9	76.5$\pm$2.5	94.3$\pm$1.5
	IQS-BO-GP*	75.0$\pm$2.8	76.7$\pm$2.6	80.7$\pm$2.3	74.7$\pm$2.6	87.6$\pm$2.0	96.1$\pm$1.2
$n_{\text{init}} = 40$	IQS-BO†	79.6$\pm$2.2	80.6$\pm$2.1	92.4$\pm$1.6	91.6$\pm$1.7	94.2$\pm$1.5	98.8$\pm$0.7
	IQS-BO-PFN†	81.2$\pm$2.1	81.7$\pm$2.1	92.4$\pm$1.6	90.0$\pm$1.8	94.7$\pm$1.3	98.3$\pm$0.8
	IQS-BO-GP†	81.1$\pm$2.3	82.7$\pm$2.3	91.6$\pm$1.7	87.9$\pm$1.8	94.3$\pm$1.4	98.3$\pm$0.8
	IQS-BO*	68.6$\pm$2.6	71.9$\pm$2.5	82.5$\pm$2.4	80.3$\pm$2.4	87.8$\pm$2.0	96.2$\pm$1.2
	IQS-BO-PFN*	58.6$\pm$2.9	62.6$\pm$2.9	75.2$\pm$2.5	68.2$\pm$2.6	80.6$\pm$2.3	95.6$\pm$1.3
	IQS-BO-GP*	70.1$\pm$2.9	72.6$\pm$2.8	79.9$\pm$2.4	77.5$\pm$2.2	87.3$\pm$2.0	96.8$\pm$1.1

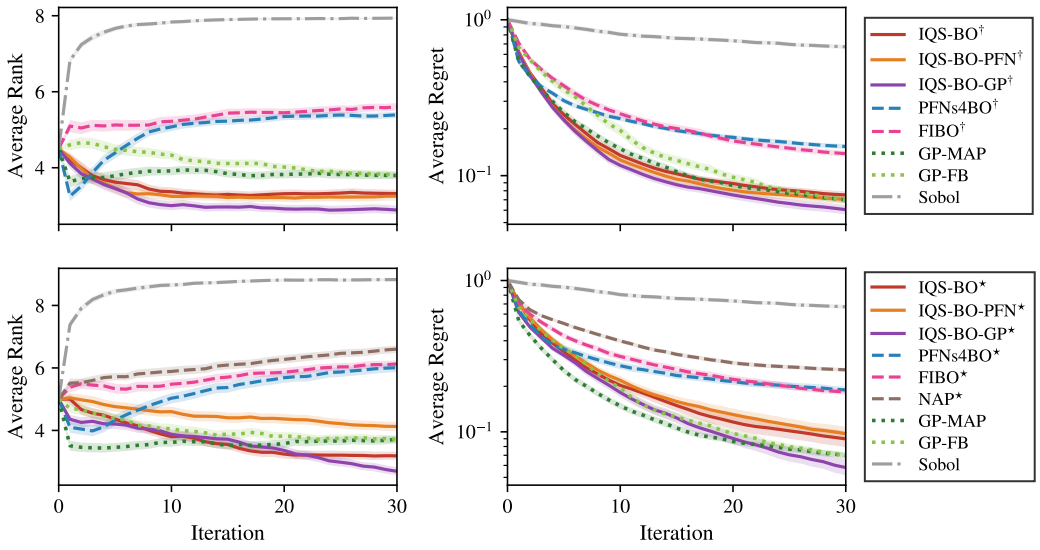


Figure 8: As Figure 2, on the five 4D problems, $n_{\text{init}} = 40$. Curves for $n_{\text{init}} = 10$ and 20 show the same pattern.

Dependence on the prior Pre-trained on the HEBO prior instead of the mixture prior, IQS-BO-GP beats EI-CAND in 54 to 58% of runs on R^{final} and loses to it on R^{AUC}, winning 30 to 47% (bottom block of Table 15). The learned rule still reaches a better final regret than expected improvement but is slower early in the run, and the clear advantage of Section 4.3 comes from the mixture prior.

PFN surrogate The same comparison can be read from existing results with a PFN surrogate, since IQS-BO-PFN reads the released PFNs4BO-HEBO model unchanged and differs from it only in how the query is chosen. IQS-BO-PFN ends with lower regret than PFNs4BO-HEBO in 60, 67

Table 14: Mean normalised final regret per 4D problem, $n_{\text{init}} = 40$, 90 seeds, $\pm$ 95% CI. The GP-prior column averages ten instances. Bold marks the best value per problem.

Method	Ackley	Levy	Rosenbrock	Benzylation	GP prior
IQS-BO†	$\mathbf{0.111}_{\pm 0.006}$	$0.016_{\pm 0.008}$	$0.028_{\pm 0.002}$	$0.141_{\pm 0.016}$	$0.079_{\pm 0.011}$
IQS-BO-PFN†	$0.118_{\pm 0.006}$	$0.031_{\pm 0.013}$	$0.026_{\pm 0.002}$	$0.135_{\pm 0.016}$	$0.043_{\pm 0.007}$
IQS-BO-GP†	$0.123_{\pm 0.006}$	$0.024_{\pm 0.009}$	$\mathbf{0.025}_{\pm 0.002}$	$0.113_{\pm 0.018}$	$0.018_{\pm 0.004}$
PFNs4BO†	$0.432_{\pm 0.020}$	$0.116_{\pm 0.016}$	$0.080_{\pm 0.008}$	$\mathbf{0.000}_{\pm 0.000}$	$0.142_{\pm 0.008}$
FIBO†	$0.295_{\pm 0.014}$	$0.138_{\pm 0.025}$	$0.059_{\pm 0.006}$	$0.049_{\pm 0.005}$	$0.152_{\pm 0.009}$
IQS-BO*	$0.228_{\pm 0.048}$	$\mathbf{0.006}_{\pm 0.003}$	$0.038_{\pm 0.003}$	$0.102_{\pm 0.018}$	$0.076_{\pm 0.010}$
IQS-BO-PFN*	$0.203_{\pm 0.047}$	$0.053_{\pm 0.009}$	$0.059_{\pm 0.005}$	$0.119_{\pm 0.017}$	$0.053_{\pm 0.007}$
IQS-BO-GP*	$0.150_{\pm 0.029}$	$0.047_{\pm 0.008}$	$0.033_{\pm 0.003}$	$0.048_{\pm 0.016}$	$0.013_{\pm 0.003}$
PFNs4BO*	$0.358_{\pm 0.018}$	$0.119_{\pm 0.035}$	$0.078_{\pm 0.011}$	$0.179_{\pm 0.016}$	$0.213_{\pm 0.014}$
FIBO*	$0.274_{\pm 0.010}$	$0.240_{\pm 0.038}$	$0.056_{\pm 0.004}$	$0.276_{\pm 0.006}$	$0.070_{\pm 0.009}$
NAP*	$0.335_{\pm 0.022}$	$0.065_{\pm 0.010}$	$0.097_{\pm 0.012}$	$0.480_{\pm 0.018}$	$0.307_{\pm 0.018}$
GP-MAP	$0.208_{\pm 0.010}$	$0.093_{\pm 0.012}$	$0.042_{\pm 0.004}$	$\mathbf{0.000}_{\pm 0.000}$	$0.006_{\pm 0.001}$
GP-FB	$0.218_{\pm 0.008}$	$0.087_{\pm 0.011}$	$0.039_{\pm 0.004}$	$\mathbf{0.000}_{\pm 0.000}$	$\mathbf{0.005}_{\pm 0.001}$
Sobol	$0.906_{\pm 0.028}$	$0.627_{\pm 0.059}$	$0.516_{\pm 0.063}$	$0.748_{\pm 0.038}$	$0.557_{\pm 0.018}$

Table 15: Paired win rate (%) of the first method against the second, ten 3D problems, 900 paired runs per cell, $\pm$ 95% bootstrap CI half-width. † mixture prior, * HEBO prior. EI-CAND requires no pre-training, so its comparison with GP-MAP is the same under both priors and is shown once.

	R^{final}			R^{AUC}		
n_{init}	10	20	40	10	20	40
IQS-BO-GP† vs EI-CAND	$66.1_{\pm 3.1}$	$68.2_{\pm 3.1}$	$70.3_{\pm 2.9}$	$53.3_{\pm 3.2}$	$61.6_{\pm 3.2}$	$69.6_{\pm 3.0}$
EI-CAND vs GP-MAP	$70.7_{\pm 3.1}$	$71.2_{\pm 2.9}$	$67.3_{\pm 3.1}$	$61.7_{\pm 3.2}$	$61.9_{\pm 3.2}$	$62.5_{\pm 3.1}$
IQS-BO-GP† vs GP-MAP	$77.6_{\pm 2.7}$	$78.3_{\pm 2.7}$	$81.1_{\pm 2.6}$	$62.0_{\pm 3.2}$	$70.4_{\pm 3.0}$	$75.1_{\pm 2.8}$
IQS-BO-GP* vs EI-CAND	$54.3_{\pm 3.3}$	$57.7_{\pm 3.3}$	$56.2_{\pm 3.3}$	$30.1_{\pm 3.0}$	$37.2_{\pm 3.2}$	$46.8_{\pm 3.2}$
IQS-BO-GP* vs GP-MAP	$71.0_{\pm 2.9}$	$75.0_{\pm 2.8}$	$70.1_{\pm 3.0}$	$38.8_{\pm 3.2}$	$49.0_{\pm 3.3}$	$58.7_{\pm 3.2}$

and 75% of runs at $n_{\text{init}} = 10$, 20 and 40 (Table 13), and its gain is largest on the problems where expected improvement on the PFN posterior does worst against GP-MAP.

F.3 Effect of the Pre-Training Prior

Table 16 provides the paired win rates behind the prior comparison of Section 4.3 and adds two analyses.

Effect over the run For all three IQS-BO variants, the win rates of the mixture prior on R^{AUC} exceed those on R^{final} (Table 16), so the prior helps most early in the run. The regret curves of Figure 10 show the same, as the gap between the two priors opens in the first rounds, when the model relies more on what it learned in pre-training than on the observations, and then persists.

Problems with well-specified GP surrogates Table 17 tests whether adding non-GP functions to the prior hurts on problems where a GP is already the right model. For IQS-BO and IQS-BO-PFN, the mixture-prior versions win about half of the runs or more on Rosenbrock and the GP-prior draws, so the broader prior costs them nothing there. On benzylation, both priors often reach the optimum, so ties pull the win rates towards 50% at small context sizes, but at $n_{\text{init}} = 40$ the HEBO-prior versions of IQS-BO and IQS-BO-GP win 61% and 68% of paired runs. Elsewhere, the mixture prior lowers performance only for IQS-BO-GP on the 3D Rosenbrock, PFNs4BO on the 4D Rosenbrock and FIBO on the GP-prior draws.

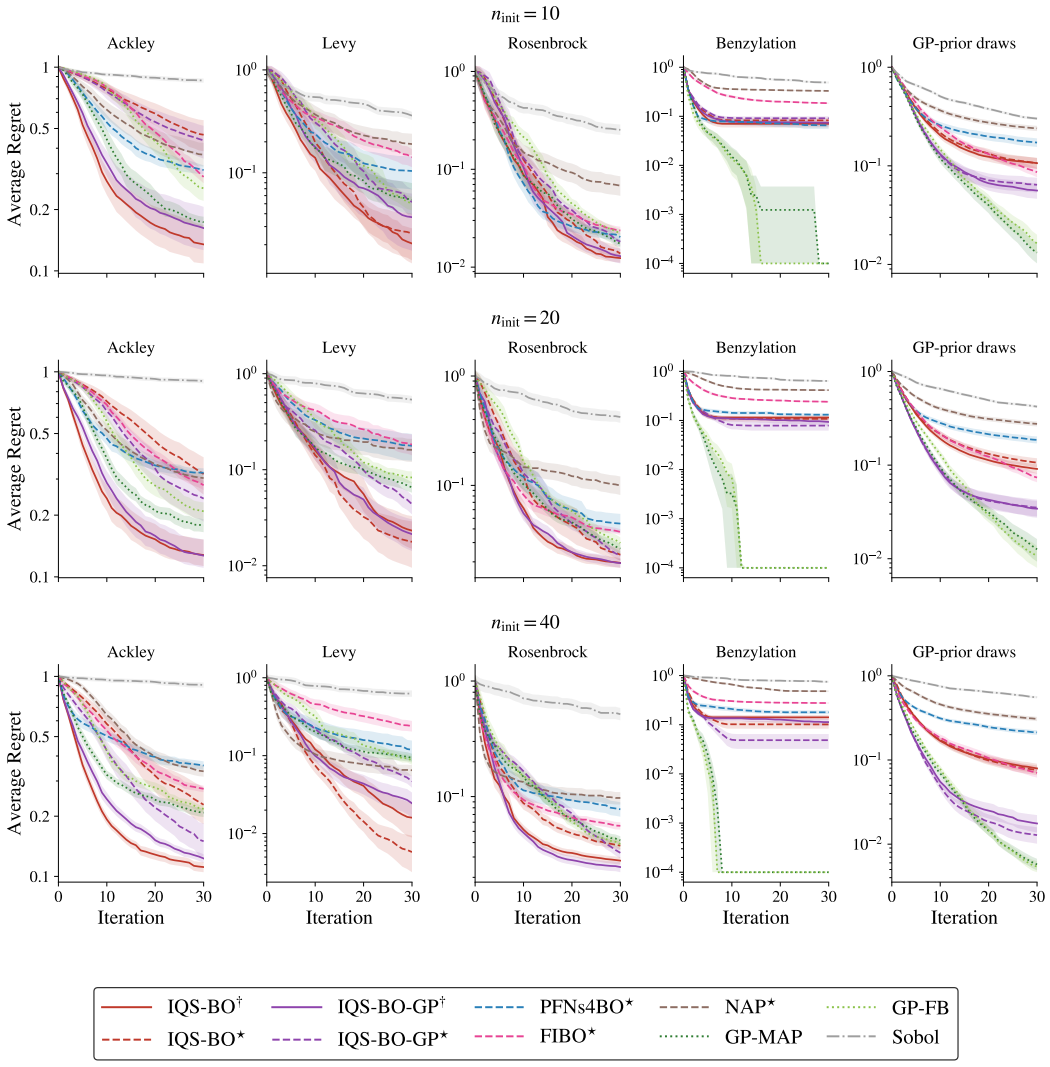


Figure 9: Average regret per iteration on each 4D problem (columns), $n_{\text{init}} = 10$, 20 and 40 (rows), 90 seeds (900 runs for the ten GP-prior draws), with 95% confidence intervals.

Table 16: Paired win rate (%) of each model pre-trained on the mixture prior against the same model pre-trained on its original prior, ten 3D problems, 900 paired runs per cell, ± 95% bootstrap CI half-width.

	R^{final}			R^{AUC}		
n_{init}	10	20	40	10	20	40
IQS-BO	66.7±3.1	68.3±3.1	68.8±3.1	76.1±2.8	75.7±2.8	76.7±2.7
IQS-BO-PFN	77.5±2.8	76.0±2.8	76.2±2.8	80.7±2.6	82.6±2.5	83.4±2.3
IQS-BO-GP	62.2±3.2	61.8±3.1	65.8±3.1	70.3±3.0	70.9±2.9	68.7±3.1
PFNs4BO	53.0±3.3	55.5±3.2	62.9±3.2	56.7±3.2	54.3±3.2	63.0±3.2
FIBO	51.1±3.3	47.9±3.3	50.9±3.3	50.9±3.2	53.2±3.3	54.2±3.2

Table 17: Paired win rate (%) on R^{final} of each model pre-trained on the mixture prior against the same model pre-trained on its original prior, on the problems for which GP surrogates are well specified, $\pm$ 95% CI. 90 paired runs per cell (900 for the GP draws).

	Rosenbrock (3D)			Rosenbrock (4D)			GP draws (4D)			Benzylation (4D)		
n_{init}	10	20	40	10	20	40	10	20	40	10	20	40
IQS-BO	$62_{\pm 10}$	$58_{\pm 10}$	$64_{\pm 10}$	$58_{\pm 10}$	$67_{\pm 9}$	$78_{\pm 8}$	$55_{\pm 3}$	$53_{\pm 3}$	$46_{\pm 3}$	$56_{\pm 6}$	$48_{\pm 6}$	$39_{\pm 7}$
IQS-BO-PFN	$72_{\pm 9}$	$64_{\pm 10}$	$61_{\pm 10}$	$80_{\pm 8}$	$92_{\pm 5}$	$91_{\pm 6}$	$63_{\pm 3}$	$65_{\pm 3}$	$65_{\pm 3}$	$45_{\pm 6}$	$45_{\pm 6}$	$45_{\pm 7}$
IQS-BO-GP	$44_{\pm 10}$	$26_{\pm 9}$	$51_{\pm 10}$	$69_{\pm 9}$	$66_{\pm 10}$	$61_{\pm 10}$	$46_{\pm 3}$	$44_{\pm 3}$	$48_{\pm 3}$	$58_{\pm 6}$	$45_{\pm 7}$	$32_{\pm 6}$
PFNs4BO	$49_{\pm 10}$	$56_{\pm 10}$	$74_{\pm 9}$	$26_{\pm 9}$	$28_{\pm 9}$	$40_{\pm 10}$	$49_{\pm 3}$	$58_{\pm 3}$	$59_{\pm 3}$	$78_{\pm 6}$	$95_{\pm 3}$	$97_{\pm 2}$
FIBO	$50_{\pm 10}$	$44_{\pm 10}$	$46_{\pm 10}$	$48_{\pm 10}$	$60_{\pm 10}$	$51_{\pm 11}$	$20_{\pm 2}$	$16_{\pm 2}$	$13_{\pm 2}$	$100_{\pm 0}$	$100_{\pm 0}$	$100_{\pm 0}$

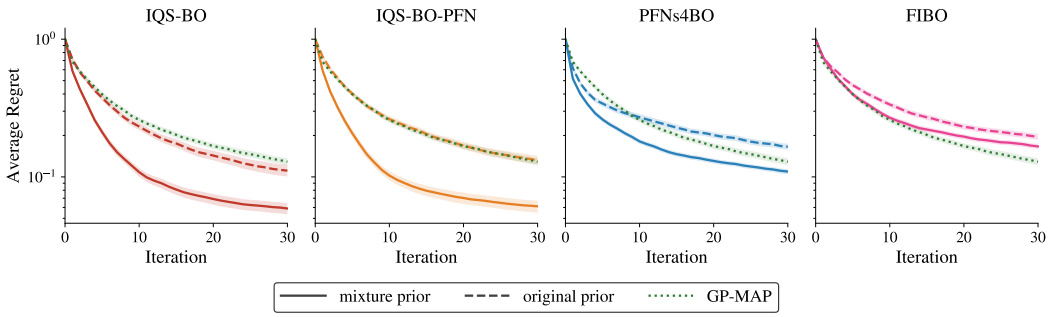


Figure 10: Average regret per iteration of IQS-BO, IQS-BO-PFN, PFNs4BO and FIBO pre-trained on the mixture prior (solid) and on their original prior (dashed), with GP-MAP as reference, ten 3D problems, $n_{\text{init}} = 40$, 90 seeds, with 95% confidence intervals.

F.4 Batch Bayesian Optimisation

Table 18 provides the per-problem GAP values behind Figure 4, for the seven 3D problems of the batch study, all but Camel and the two Olympus campaigns (Appendix E). We provide two complementary analyses.

Per-problem values Figure 11 shows how the GAP of IQS-BO and IQS-BO-PFN evolves over the evaluations on each problem. The mean GAP of IQS-BO stays between 0.91 and 0.93 across batch sizes, and its margin to the best baseline widens from 0.05 at $q = 10$ to 0.08 at $q = 50$. The batch-size ablation attributes this robustness to pre-training at the deployed batch size (Appendix F.5).

Proposal time per batch Table 19 complements the run-level proposal times of Section 4.4 with the time of a single proposal on a GPU workstation, for batch sizes from 1 to 20. The cost of every IQS-BO variant grows slowly with q, as it needs one forward pass per point, whereas that of GP-MAP and PFNs4BO-HEBO grows steeply. At $q = 20$, IQS-BO and IQS-BO-PFN are more than an order of magnitude faster than GP-MAP, and only FIBO is faster, because it draws the points of a batch independently, without conditioning on earlier draws.

F.5 Additional Ablations

This section reports ablation studies not included in Section 4.5.

Candidate composition with other scoring rules We repeat the candidate-set ablation of Section 4.5 with IQS-BO-GP and with EI-CAND, and compare each against GP-MAP. With the default candidate set, which contains 30% perturbations of the best observations, IQS-BO, IQS-BO-GP and EI-CAND beat GP-MAP in 77 to 80%, 78 to 81% and 67 to 71% of paired runs. With a space-filling set of only Sobol and uniform candidates, all three lose to it, winning only 43 to 45%, 36 to 40% and 27 to 33% of runs. The perturbed candidates therefore matter regardless of the scoring rule, since they are the main way in which the models can exploit near the best observations.

Table 18: GAP after 200 evaluations per problem and mean over the seven problems, mean over 90 seeds, with the 95% CI half-width in units of 10^{-3} as subscript, and seconds spent proposing queries over a complete run. Bold marks the best value per problem and batch size.

	method	Ackley	Hartm.	Bioproc.	Levy	Michal.	Rastr.	Rosenb.	mean	s/run
$q = 10$	IQS-BO	$.940_{\pm 6}$	$\mathbf{.999_{\pm 0}}$	$\mathbf{.999_{\pm 0}}$	$\mathbf{.998_{\pm 1}}$	$.853_{\pm 51}$	$\mathbf{.750_{\pm 39}}$	$.909_{\pm 20}$	$\mathbf{.921_{\pm 10}}$	0.9
	IQS-BO-PFN	$\mathbf{.949_{\pm 6}}$	$.998_{\pm 1}$	$.998_{\pm 1}$	$.954_{\pm 13}$	$\mathbf{.874_{\pm 47}}$	$.741_{\pm 38}$	$.933_{\pm 30}$	$\mathbf{.921_{\pm 10}}$	3.2
	GP-MAP	$.839_{\pm 13}$	$\mathbf{.999_{\pm 0}}$	$.995_{\pm 2}$	$.857_{\pm 35}$	$.760_{\pm 57}$	$.682_{\pm 40}$	$\mathbf{.973_{\pm 6}}$	$.872_{\pm 11}$	54.5
	GP-FB	$.834_{\pm 14}$	$\mathbf{.999_{\pm 0}}$	$.994_{\pm 2}$	$.885_{\pm 29}$	$.719_{\pm 61}$	$.679_{\pm 39}$	$.971_{\pm 11}$	$.869_{\pm 11}$	350.3
	PFNs4BO-HEBO	$.773_{\pm 20}$	$.968_{\pm 10}$	$.879_{\pm 41}$	$.946_{\pm 16}$	$.797_{\pm 39}$	$.714_{\pm 31}$	$.855_{\pm 37}$	$.848_{\pm 11}$	182.6
	PFNs4BO-BNN	$.719_{\pm 18}$	$.975_{\pm 11}$	$.987_{\pm 8}$	$.949_{\pm 15}$	$.547_{\pm 59}$	$.678_{\pm 35}$	$.850_{\pm 39}$	$.815_{\pm 12}$	131.8
	FIBO	$.805_{\pm 15}$	$.994_{\pm 2}$	$.975_{\pm 14}$	$.941_{\pm 16}$	$.810_{\pm 39}$	$.568_{\pm 43}$	$.863_{\pm 35}$	$.851_{\pm 10}$	0.6
$q = 20$	IQS-BO	$.930_{\pm 10}$	$\mathbf{.998_{\pm 1}}$	$\mathbf{.999_{\pm 1}}$	$\mathbf{.995_{\pm 3}}$	$.871_{\pm 40}$	$.781_{\pm 37}$	$.931_{\pm 18}$	$.929_{\pm 8}$	0.9
	IQS-BO-PFN	$\mathbf{.947_{\pm 6}}$	$\mathbf{.998_{\pm 1}}$	$.998_{\pm 1}$	$.922_{\pm 25}$	$\mathbf{.953_{\pm 22}}$	$\mathbf{.790_{\pm 31}}$	$.937_{\pm 25}$	$\mathbf{.935_{\pm 7}}$	3.1
	GP-MAP	$.819_{\pm 13}$	$\mathbf{.998_{\pm 0}}$	$.983_{\pm 7}$	$.832_{\pm 41}$	$.680_{\pm 55}$	$.598_{\pm 49}$	$\mathbf{.964_{\pm 10}}$	$.839_{\pm 12}$	60.8
	GP-FB	$.816_{\pm 12}$	$\mathbf{.998_{\pm 0}}$	$.976_{\pm 9}$	$.808_{\pm 48}$	$.631_{\pm 64}$	$.617_{\pm 43}$	$.960_{\pm 14}$	$.829_{\pm 13}$	417.0
	PFNs4BO-HEBO	$.756_{\pm 15}$	$.967_{\pm 8}$	$.859_{\pm 47}$	$.943_{\pm 15}$	$.809_{\pm 38}$	$.715_{\pm 37}$	$.852_{\pm 33}$	$.843_{\pm 12}$	208.5
	PFNs4BO-BNN	$.710_{\pm 21}$	$.974_{\pm 9}$	$.989_{\pm 6}$	$.951_{\pm 13}$	$.589_{\pm 60}$	$.664_{\pm 37}$	$.878_{\pm 29}$	$.822_{\pm 12}$	135.8
	FIBO	$.806_{\pm 17}$	$.993_{\pm 2}$	$.965_{\pm 18}$	$.935_{\pm 23}$	$.768_{\pm 48}$	$.557_{\pm 46}$	$.864_{\pm 34}$	$.841_{\pm 12}$	0.4
$q = 50$	IQS-BO	$\mathbf{.860_{\pm 13}}$	$\mathbf{.998_{\pm 1}}$	$\mathbf{.989_{\pm 5}}$	$\mathbf{.975_{\pm 7}}$	$\mathbf{.853_{\pm 46}}$	$\mathbf{.796_{\pm 27}}$	$.892_{\pm 34}$	$\mathbf{.909_{\pm 9}}$	1.2
	GP-MAP	$.787_{\pm 13}$	$\mathbf{.998_{\pm 1}}$	$.945_{\pm 17}$	$.741_{\pm 52}$	$.619_{\pm 58}$	$.550_{\pm 50}$	$\mathbf{.931_{\pm 25}}$	$.796_{\pm 14}$	45.1
	GP-FB	$.799_{\pm 16}$	$\mathbf{.998_{\pm 0}}$	$.954_{\pm 11}$	$.703_{\pm 61}$	$.489_{\pm 65}$	$.599_{\pm 52}$	$.926_{\pm 27}$	$.781_{\pm 15}$	396.3
	PFNs4BO-HEBO	$.692_{\pm 21}$	$.953_{\pm 16}$	$.774_{\pm 52}$	$.908_{\pm 28}$	$.781_{\pm 44}$	$.604_{\pm 49}$	$.780_{\pm 51}$	$.784_{\pm 15}$	225.4
	PFNs4BO-BNN	$.667_{\pm 17}$	$.971_{\pm 9}$	$.986_{\pm 5}$	$.946_{\pm 14}$	$.657_{\pm 50}$	$.641_{\pm 39}$	$.858_{\pm 35}$	$.818_{\pm 11}$	98.7
	FIBO	$.806_{\pm 14}$	$.988_{\pm 4}$	$.955_{\pm 26}$	$.911_{\pm 32}$	$.755_{\pm 48}$	$.538_{\pm 48}$	$.852_{\pm 33}$	$.829_{\pm 12}$	0.3

Table 19: Median seconds per proposed batch on a GPU workstation (Quadro P2200, Xeon W-1350P), Ackley in 3D, $n = 30$ observations, ten repeats (three for PFNs4BO-HEBO), with the interquartile range in brackets. † mixture prior.

q	1	2	10	20
IQS-BO†	0.011 [0.011, 0.011]	0.020 [0.020, 0.022]	0.085 [0.085, 0.085]	0.170 [0.170, 0.171]
IQS-BO-PFN†	0.030 [0.030, 0.030]	0.060 [0.060, 0.060]	0.302 [0.301, 0.303]	0.610 [0.604, 0.612]
IQS-BO-GP†	0.096 [0.096, 0.096]	0.105 [0.105, 0.106]	0.185 [0.183, 0.186]	0.282 [0.280, 0.284]
FIBO	0.021 [0.021, 0.021]	0.022 [0.022, 0.022]	0.022 [0.022, 0.022]	0.022 [0.022, 0.023]
GP-MAP	0.233 [0.233, 0.233]	0.232 [0.232, 0.233]	3.47 [3.47, 3.47]	10.7 [10.7, 10.8]
PFNs4BO-HEBO	3.18 [3.17, 3.19]	6.78 [6.76, 6.79]	32.6 [32.6, 32.6]	64.9 [64.9, 65.0]

Batch size and batch construction Figure 11 compares each checkpoint pre-trained at the deployed batch size with the $q = 2$ checkpoint used at the same q. For IQS-BO, the pre-trained checkpoint beats the $q = 2$ checkpoint in 62, 91 and 98% of paired runs at $q = 10$, 20 and 50, and the mean GAP of the $q = 2$ checkpoint drops from 0.88 to 0.63 and 0.44, while that of the pre-trained checkpoint stays above 0.90. For IQS-BO-PFN, the $q = 2$ checkpoint degrades much less, with a mean GAP between 0.84 and 0.90, likely because the frozen PFN conditions on the pending points and so still separates them. Sequential selection also matters more as the batch grows. At $q = 10$, it outperforms top-q selection, which takes the q highest-scoring candidates of a single forward pass without pending points, in 68% of paired runs.

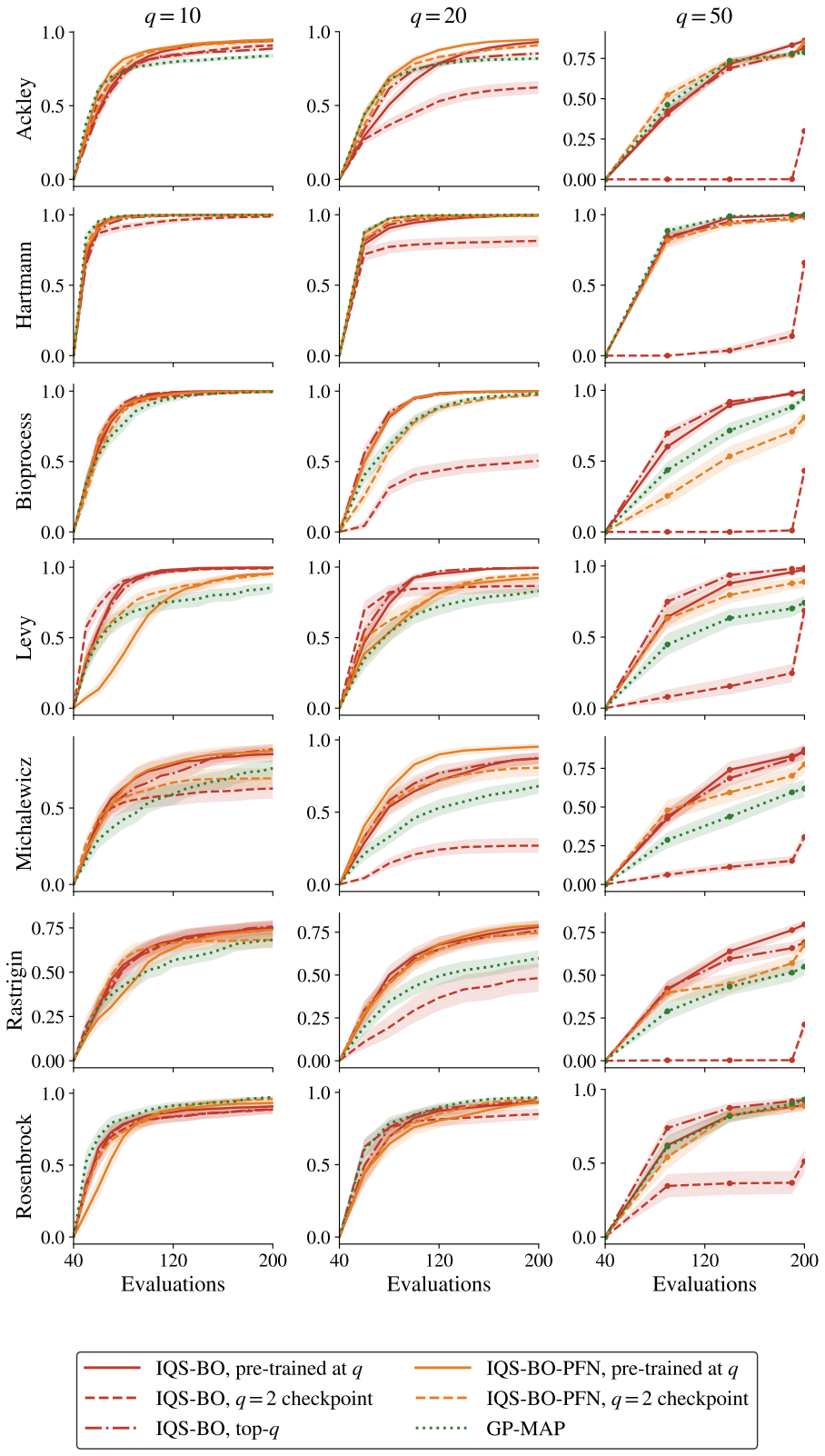


Figure 11: GAP per evaluation on each problem of the batch study (rows), for $q = 10$, 20 and 50 (columns), networks pre-trained on the HEBO prior, $n_{\text{init}} = 40$, 90 seeds, with 95% confidence intervals. The y-axis of every panel shows the GAP. Solid lines are checkpoints pre-trained at the deployed batch size, dashed lines the $q = 2$ checkpoints used at that batch size.