PABBO: PREFERENTIAL AMORTIZED BLACK-BOX OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Preferential Bayesian Optimization (PBO) is a sample-efficient method to learn latent user utilities from preferential feedback over a pair of designs. It relies on a statistical surrogate model for the latent function, usually a Gaussian process, and an acquisition strategy to select the next candidate pair to get user feedback on. Due to the non-conjugacy of the associated likelihood, every PBO step requires a significant amount of computations with various approximate inference techniques. This computational overhead is incompatible with the way humans interact with computers, hindering the use of PBO in real-world cases. Building on the recent advances of amortized BO, we propose to circumvent this issue by fully amortizing PBO, meta-learning both the surrogate and the acquisition function. Our method comprises a novel transformer neural process architecture, trained using reinforcement learning and tailored auxiliary losses. On a benchmark composed of synthetic and real-world datasets, our method is several orders of magnitude faster than the usual Gaussian process-based strategies and often outperforms them in accuracy.

1 INTRODUCTION

Learning the latent utility function of a given user from its feedback is a task that is becoming increasingly important in the era of personalization, with applications ranging from the optimization of visual designs (Koyama et al., 2020), thermal confort (Abdelrahman & Miller, 2022), proportional integral controller (Coutinho et al., 2024), or robotic gait (Li et al., 2021). As humans are typically better at comparing two options rather than assessing their absolute value (Kahneman & Tversky, 2013), current solutions often leverage data based on *preferential* feedback, the outcome of a pairwise comparison also referred to as a *duel* (Chu & Ghahramani, 2005).

Preferential Bayesian Optimization (PBO, González et al. (2017)) is the gold standard method for optimizing black-box functions from sequential duel feedback. To learn the latent utility of a human subject, PBO operates in a Bayesian framework and classically relies on a Gaussian Process (GP) prior (Rasmussen & Williams, 2006) and a probit or logit likelihood (Brochu et al., 2010). Due to the non-conjugacy of the preferential likelihood, approximation techniques are required for posterior inference (Takeno et al., 2023). The obtained posterior is then plugged into an *acquisition function*, which selects the next pair of candidate designs to display to the user, in a way that balances exploration and exploitation (Garnett, 2023). Approximating the posterior and maximizing the acquisition function results in a significant computational overhead, which may hinder the use of PBO in real-world cases, as the user would have to wait an extended period of time before each iteration.

To circumvent this issue, amortization (Amos, 2023), or pre-computing solutions and learning an ML model to approximate them, has emerged as a promising solution, effectively enabling rapid and still accurate on-line computation with sophisticated models, trading off the speed to off-line computations. Amortization has been successfully applied to Simulation-based inference (Cranmer et al., 2020; Gloeckler et al., 2024), sequential Bayesian experimental design (Foster et al., 2021), and more relevantly, vanilla Bayesian Optimization (Maraval et al., 2023; Song et al., 2024).

Beyond speed gains, an interesting characteristic of amortization with regards to BO and PBO is that it can jointly model the probabilistic surrogate and the acquisition function, two modules that are usually treated independently in BO. This provides two powerful advantages: rather than guessing which combination of GP kernel and acquisition function works best for a given task through various heuristics, these are now learned from data. In the end, the model will directly predict the acquisition

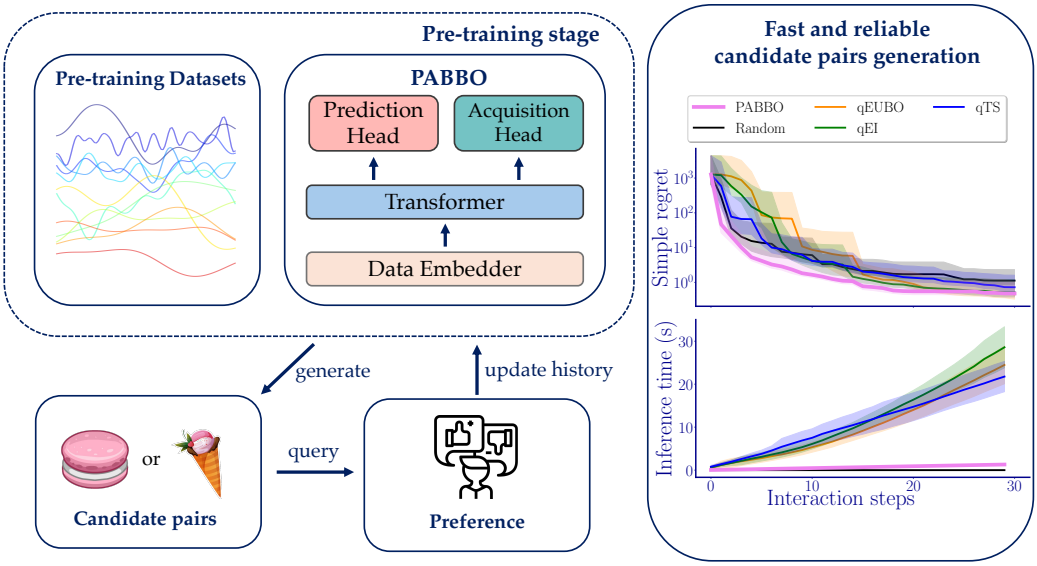

Figure 1: Leveraging synthetic and existing BO datasets, PABBO learns an acquisition policy in an end-to-end manner, thus directly amortizing the design proposal step. Contrary to existing methods, at inference time, PABBO only relies on preference data. Our method outperforms GP-based strategies and dramatically speeds up the optimization.

function values. However, current end-to-end black-box optimization methods predict acquisition function values for single designs. Translating this task in the preferential framework requires predicting acquisition values for pairs of designs, suggesting novel network architectures and training procedures. To the best of our knowledge, amortization has not yet been employed in a PBO setting.

Contributions.

- We introduce PABBO, the first end-to-end framework for Preferential Amortized Black-Box Optimization (Table 1). PABBO directly outputs acquisition function values for candidate pairs, conditioned on observed data, and handles preferential feedback through a novel transformer-based architecture.
- We present a reinforcement learning-based pre-training scheme to directly learn acquisition function values for pairs. Due to the sparsity of the rewards, in a similar spirit as Maraval et al. (2023), we add an auxiliary loss, but based on binary classification instead of regression, due to the preferential nature of the data. Figure 1 displays the PABBO workflow.
- On a set of synthetic and real-world examples, PABBO provides speedups of several orders of magnitude against GP-based methods, often outperforming them, even on unseen examples during pre-training, thus demonstrating its impressive in-context optimization capability. We conclude by illustrating the robustness of PABBO through a series of ablation studies.

Method	Preference Feedback	Amortization	End-to-End training	Reference
BO	✗	✗	✗	(Garnett, 2023)
qEUBO	✓	✗	✗	(Astudillo et al., 2023)
qEI	✓	✗	✗	(Siivola et al., 2021)
qTS	✓	✗	✗	(Hernández-Lobato et al., 2017)
PFN	✗	✓	✗	(Müller et al., 2023)
NAP	✗	✓	✓	(Maraval et al., 2023)
PABBO	✓	✓	✓	This work

Table 1: PABBO is the first end-to-end framework for Preferential Amortized Black-Box Optimization.

2 PRELIMINARIES

2.1 PREFERENTIAL BLACK-BOX OPTIMIZATION

The goal of preferential black-box optimization (PBBO) is to determine the optimal solution $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\mathbb{X}} f(\boldsymbol{x})$ using the duel feedback $\boldsymbol{x} \succ \boldsymbol{x}'$, signifying a preference of $\boldsymbol{x}$ over $\boldsymbol{x}'$. The preference structure is governed by the latent function $f : \mathbb{X} \to \mathbb{R}$ defined over the subset $\mathbb{X} \subset \mathbb{R}^d$:

$$\boldsymbol{x} \succ \boldsymbol{x}' \iff f(\boldsymbol{x}) + \varepsilon > f(\boldsymbol{x}') + \varepsilon', \tag{1}$$

where ε and ε' are i.i.d. realizations of the normal distribution $\mathcal{N}(0, \sigma^2_{\text{noise}})$.

By definition of PBBO, we only have access to a dataset $\mathcal{D}$ made of the human observations of m duel feedbacks $\{l_i\}_{i=1}^m = \{\boldsymbol{x}_i \succ \boldsymbol{x}'_i\}_{i=1}^m$, with $\boldsymbol{x}_i$ and $\boldsymbol{x}'_i$ being the winner and the loser of the duel, respectively, and l_i a binary variable recording the location of preferred instance in a query. No information about the latent function is provided, e.g., its analytic form, convexity, or regularity.

2.2 AMORTIZATION

Amortization is a data-driven approach that involves training a model, often a neural network, on a large set of pre-collected tasks. The goal is to enable the model to capture and leverage shared information across these tasks, thus learning how to make predictions efficiently on similar tasks in the future. This process allows the model to perform approximate inference rapidly during the inference stage, since knowledge from related tasks encountered during training has already been internalized.

Recently, amortization has been introduced to BO, either amortizing the surrogate model (Müller et al., 2023), the acquisition function (Swersky et al., 2020; Volpp et al., 2020), or directly end-to-end training system (Yang et al., 2024; Maraval et al., 2023; Chen et al., 2022). Our work aims to employ a fully end-to-end trained system that learns how to optimize functions directly from preference data.

Designing a task-specific architecture is crucial to efficiently utilize data in amortized learning. Conditional neural processes (CNPs; Garnelo et al. 2018), as a meta-learning framework built on Deep Sets (Zaheer et al., 2017), are particularly well-suited for tasks like BO (Maraval et al., 2023; Müller et al., 2023), where historical queries are permutation-invariant. Transformer-based neural processes (Nguyen & Grover, 2022; Müller et al., 2022) extend this idea by leveraging the attention mechanism to more effectively capture interactions between points.

2.3 REINFORCEMENT LEARNING

Reinforcement Learning (RL) has recently found success in the Black-Box Optimization realm (Volpp et al., 2020; Hsieh et al., 2021; Maraval et al., 2023), due to its ability to yield *non-myopic* acquisition strategies. This leads to better performances since these strategies consider the influence of future evaluations up to a given horizon determined by the available budget T. An RL problem is defined as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, $\mathcal{S}$ and $\mathcal{A}$ stand for the state and action spaces, respectively, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition model, and $\mathcal{R}$ is the reward function which encompasses a given optimization goal. The goal is to learn a probability distribution over state-action pairs $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, referred to as a policy, such that π maximizes the discounted expected returns with discount factor $\gamma \in [0, 1]$, which defines the degree of myopia of the policy.

3 PREFERENTIAL AMORTIZED BLACK-BOX OPTIMIZATION (PABBO)

To solve the problem introduced in Section 2.1 in an amortized manner, PABBO proceeds as illustrated in Figure 2. In brief, PABBO directly outputs acquisition values for candidate pairs through the policy π_θ, which is parameterized by a transformer backbone and an *acquisition head*. To stabilize the training, we additionally employ a *prediction head* for an auxiliary preference prediction task. Section 3.1 describes the MDP we employ for learning the policy π_θ, Section 3.2 details the conception of the pre-training dataset accordingly to the neural process framework, and Section 3.3 focuses on the transformer architecture and the loss functions involved in its actual training.

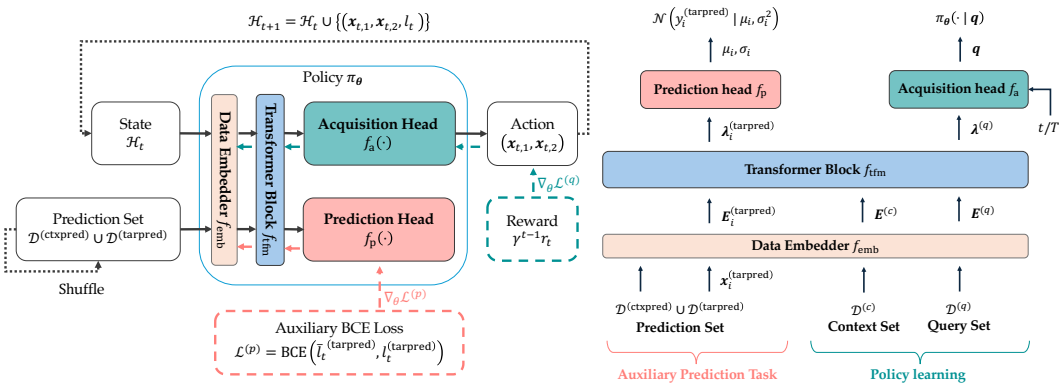

Figure 2: Flowchart of PABBO (left), together with a zoom on our proposed transformer block (right). We use reinforcement learning to guide the acquisition head in proposing valuable query pairs, and apply an auxiliary BCE loss to the prediction head to stabilize the training of the transformer.

3.1 POLICY LEARNING

Unlike current PBO methods, we leverage RL to learn a non-myopic policy π_θ. For a single task, our RL problem is formulated as follows:

- **States**: $s_t = [\mathcal{H}_t, t, T]$, where $\mathcal{H}_t = \{(\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, l_i)\}_{i=1}^{t-1}$ contains the optimization history made of $(t-1)$-step queries and preferences.
- **Actions**: $a_t = (\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2})$, the next query pair.
- **Rewards**: $r_t = \max_{1 \le i \le t} \max\{y_{i,1}, y_{i,2}\}$, the best objective function values observed so far with $y_{i,1}$ and $y_{i,2}$ being associated to inputs $\boldsymbol{x}_{i,1}$ and $\boldsymbol{x}_{i,2}$, respectively.

State transitions consist in updating the optimization history with the selected next query and preference feedback, $\mathcal{H}_{t+1} = \mathcal{H}_t \cup \{(\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, l_t)\}$. We seek a policy maximizing the cumulative rewards along the entire optimization trajectory while performing well on average across tasks sampled from p_{tasks}. Rewards r_t denote the maximal objective function values observed at step t. Such values are always available in synthetic pre-training datasets, like GP draws. If the latent function values are unavailable during pre-training, ranking data can be used to obtain a score for each point as the reward. We show in Section 5.4 that this substitution does not greatly alter performance.

3.2 PRE-TRAINING DATA ARCHITECTURE

The PABBO workflow involves a pre-training phase, where a policy is learned on all meta-tasks, and an inference phase, where the trained model is applied to predict acquisition values on a new task directly. Our approach effectively amortizes the optimization process over multiple tasks, allowing rapid inference by leveraging the shared structure learned during pre-training. The pre-training dataset for each meta-task features three parts:

- A **context set** $\mathcal{D}^{(c)} = \{(\boldsymbol{x}_{i,1}^{(c)}, \boldsymbol{x}_{i,2}^{(c)}, l_i^{(c)})\}_{i=1}^{t-1}$ containing all past queries and preferences.
- A **query set** $\mathcal{D}^{(q)} = \{(\boldsymbol{x}_i^{(q)})\}_{i=1}^{S}$ consisting of S locations that the model considers for the next step, which will be further combined into a candidate query pair set $Q = \{(\boldsymbol{x}_i^{(q)}, \boldsymbol{x}_j^{(q)}) \mid \boldsymbol{x}_i^{(q)}, \boldsymbol{x}_j^{(q)} \in \mathcal{D}^{(q)}, i \neq j\}$ of size M. In a training iteration, we randomly sample S query points, and candidate query pairs are chosen from all possible pairwise combinations within this set. At inference time, query points are sampled across the search space from a quasi-random Sobol sequence, and all possible pairs from the full set of combinations are considered. S should be set to a sufficiently large value and scaled with data dimension to ensure adequate exploration.
- A **prediction set** $\mathcal{D}^{(p)} = \{(\boldsymbol{x}_{i,1}^{(p)}, \boldsymbol{x}_{i,2}^{(p)}, l_i^{(p)})\}_{i=1}^{N}$ containing N queries and preferences. Similarly, we generate $\mathcal{D}^{(p)}$ by sampling pairwise combinations of a set of random points. This prediction set is used to compute an auxiliary binary classification loss: for each op-

timization step, we randomly sample a subset of $\mathcal{D}^{(p)}$ which refers as *prediction context set* $\mathcal{D}^{(\text{ctxpred})} = \{(\boldsymbol{x}_{i,1}^{(\text{ctxpred})}, \boldsymbol{x}_{i,2}^{(\text{ctxpred})}, l_i^{(\text{ctxpred})})\}_{i=1}^{N^{(\text{ctxpred})}}$, and all the points from the rest of P queries will be considered as a *prediction target set* $\mathcal{D}^{(\text{tarpred})} = \{(\boldsymbol{x}_i^{(\text{tarpred})})\}_{i=1}^{N^{(\text{tarpred})}}$. The model is required to estimate the latent function values for each point in $\mathcal{D}^{(\text{tarpred})}$ conditioned on $\mathcal{D}^{(\text{ctxpred})}$, and further recover the preference relationship of a possible pair.

It is worth mentioning that no reward information can leak from the query set to the prediction set as entirely different instances are assigned to each set.

3.3 Model Architecture

PBBO differs from standard BO in two aspects: (1) preferential outcomes instead of exact latent function values are observed, and (2) a pair of points is expected as the next query instead of a single point. To address these challenges, PABBO leverages novel data encoding and decoding structures, efficiently handling pairwise preference data. At the center of PABBO's architecture lie conditional neural processes, a class of models quite amenable for handling the sets that naturally arise due to the permutation invariance of the optimization history. Specifically, the PABBO model architecture, presented in Figure 2, consists of four main components: the data embedder f_{emb}, transformer block f_{tfm}, prediction head f_{p}, and the acquisition head f_{a}.

Firstly, inputs from all parts of the dataset are mapped into a shared embedding space via the **data embedder** f_{emb}. The first and second instance within a pair $\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}$ and their corresponding preference l_i are encoded separately with individual MLPs into embeddings of the same size and added up as $\mathbf{E} = \mathbf{E}^{(\boldsymbol{x}_{i,1})} \oplus \mathbf{E}^{(\boldsymbol{x}_{i,2})} \oplus \mathbf{E}^{(l_i)}$. For parts $\mathcal{D}^{(q)}$ and $\mathcal{D}^{(\text{tarpred})}$ containing only a single instance $\boldsymbol{x}_i$, $\mathbf{E} = \mathbf{E}^{\boldsymbol{x}_i}$. This gives us embeddings $\mathbf{E}^{(c)}, \mathbf{E}^{(p)}, \mathbf{E}^{(\text{ctxpred})}$ and $\mathbf{E}^{(\text{tarpred})}$ corresponding to $\mathcal{D}^{(c)}, \mathcal{D}^{(q)}, \mathcal{D}^{(\text{ctxpred})}$ and $\mathcal{D}^{(\text{tarpred})}$, respectively.

Next, embeddings are passed through the **transformer block** f_{tfm}, which captures the interactions between different parts of a meta-dataset. Dependencies between elements are controlled with masking in self-attention layers of f_{tfm}. Precisely, $\mathcal{D}^{(c)}$, the observations collected by the current step, can be attended by other elements in $\mathcal{D}^{(c)}$ and all the query points in $\mathcal{D}^{(q)}$, while each query point can only be attended by itself. Our interest is the Transformer outputs for query set $\boldsymbol{\lambda}^{(q)} = f_{\text{ftm}}(\mathbf{E}^{(q)})$. The same rule applies for the prediction context set $\mathcal{D}^{(\text{ctxpred})}$ and prediction target set $\mathcal{D}^{(\text{tarpred})}$, leading to the transformer outputs $\boldsymbol{\lambda}^{(\text{tarpred})} = f_{\text{ftm}}(\mathbf{E}^{(\text{tarpred})})$. These outputs $\boldsymbol{\lambda} = [\boldsymbol{\lambda}^{(q)}, \boldsymbol{\lambda}^{(\text{tarpred})}]$ are then processed with different decoders according to their specified task needs.

Subsequently, $\boldsymbol{\lambda}^{(q)}$ are combined into pairs $\{(\boldsymbol{\lambda}_i^{(q)}, \boldsymbol{\lambda}_j^{(q)}) \mid (\boldsymbol{x}_i^{(q)}, \boldsymbol{x}_j^{(q)}) \in Q)\}$ according to the candidate query pair set Q[1]. Each pair $(\boldsymbol{\lambda}_i^{(q)}, \boldsymbol{\lambda}_j^{(q)})$ passes through the **acquisition head** f_{a} along with current step t and budget T to predict the acquisition function value $\mathbf{q}_{i,j} = f_a(\boldsymbol{\lambda}_i^{(q)}, \boldsymbol{\lambda}_j^{(q)}, t, T)$ for associated query pair. This allows us to form a policy for proposing the next-step query $(\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2})$ from Q by mapping the outputs of f_a, $\mathbf{q} = \{\mathbf{q}_{i,j} \mid (\boldsymbol{x}_i^{(q)}, \boldsymbol{x}_j^{(q)}) \in Q\}$, to a categorical distribution using the SoftMax function:

$$\pi_\theta((\boldsymbol{x}_i^{(q)}, \boldsymbol{x}_j^{(q)}) \mid \mathcal{H}_t, t, T) = \frac{\exp \mathbf{q}_{i,j}}{\sum_{\mathbf{q}_m \in \mathbf{q}} \exp \mathbf{q}_m}. \tag{2}$$

As defined in Section 3.1, the best objective function value obtained so far is chosen as the immediate reward when training this policy, since the goal is to find the global optimum as efficiently as possible. To further pursue a non-myopic solution, we consider the cumulative reward across the entire query trajectory, incorporating a discount factor γ, which controls the rate at which future rewards are discounted. The loss function is then written as the negative expected reward over the entire optimization trajectory:

$$\mathcal{L}^{(q)} = -\sum_{t=1}^{T} \gamma^{t-1} r_t \log \pi_\theta((\boldsymbol{x}_{t,1}^{(q)}, \boldsymbol{x}_{t,2}^{(q)}) \mid \mathcal{H}_t, t, T). \tag{3}$$

[1] To reduce computational costs from evaluating all the possible pairwise combinations between query points, which are of size $O(S^2)$, we randomly sample $M \ll S^2$ pairs as Q for each meta-task during pre-training.

In addition to the forward process through f_a, we handle the sparsity of rewards by introducing an auxiliary path leading to the **prediction head** f_p to encourage the learning of latent function shape from preferential data. For each prediction target point $\boldsymbol{x}_i^{(\text{tarpred})} \in \mathcal{D}^{(\text{tarpred})}$, f_p parameterizes a Gaussian distribution:

$$p\left(y_i|\boldsymbol{x}_i^{(\text{tarpred})}\right) = \mathcal{N}\left(y_i^{(\text{tarpred})}; (\mu_i, \sigma_i^2) := f_p\left(\boldsymbol{\lambda}_i^{(\text{tarpred})}\right)\right). \tag{4}$$

Using independent Gaussian likelihoods is common in the Neural Processes community (Garnelo et al., 2018). Correlations can be accounted for if the downstream task requires it (Markou et al., 2022). A similar choice is often found in PBO, using the probit likelihood (Brochu et al., 2010).

At any two target points $\boldsymbol{x}_i^{(\text{tarpred})}$ and $\boldsymbol{x}_j^{(\text{tarpred})}$ that constitute a query pair in $\mathcal{D}^{(p)}$, the predicted latent function values $\bar{y}_i^{(\text{tarpred})}$ and $\bar{y}_j^{(\text{tarpred})}$ are obtained by sampling from p, thus capturing the randomness in the process as described by Equation 1. The final preference is $\bar{l}^{(\text{tarpred})} := \sigma(\bar{y}_j^{(\text{tarpred})} - \bar{y}_i^{(\text{tarpred})})$ with σ the Sigmoid function. As mentioned, at each optimization step, we generate a prediction task with different contexts by shuffling the prediction set $\mathcal{D}^{(p)}$, creating different splits of $\mathcal{D}^{(\text{ctxpred})}$ and $\mathcal{D}^{(\text{tarpred})}$ to fully utilize the data. f_p is trained by minimizing the binary cross entropy (BCE) loss between predicted preferred location $\bar{l}^{(\text{tarpred})}$ and the ground truth $l^{(\text{tarpred})}$ on all the prediction tasks:

$$\mathcal{L}^{(p)} = -\sum_{t=1}^{T}\sum_{i=1}^{P}\left[l_{t,i}^{(\text{tarpred})}\log(\bar{l}_{t,i}^{(\text{tarpred})}) + (1 - l_{t,i}^{(\text{tarpred})})\log(1 - \bar{l}_{t,i}^{(\text{tarpred})})\right], \tag{5}$$

where $l_{t,i}^{(\text{tarpred})}$ and $\bar{l}_{t,i}^{(\text{tarpred})}$ represent the ground truth and predicted preference between a possible pair of prediction target points at step t, respectively.

The final objective $\mathcal{L}$ for policy learning combines the two losses: $\mathcal{L} := \mathcal{L}^{(q)} + \lambda\mathcal{L}^{(p)}$. We fix $\lambda = 1$ in all the experiments. In practice, we warm up the model with only the prediction task by setting $\mathcal{L} = \mathcal{L}^{(p)}$ in the initial training steps, the rationale being that the policy can only perform well once the shape of the latent function has been learned. Algorithm 1 describes the complete meta-learning training procedure, and Algorithm 2 describes how PABBO operates at test-time on a new task.

Algorithm 1 Preferential Amortized Black-box Optimization (PABBO)

Require: K meta-tasks, candidate query pair set size M, discount factor γ, query budget T
  for each training iteration **do**
    initialize $\mathcal{D}^{(p)}, \mathcal{D}^{(q)}$, sample Q of size M, set $\mathcal{H}_0 \leftarrow \{\emptyset\}$
    for $t = 1, \ldots, T$ **do** ▷ start policy learning
      $(\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}) \sim \pi_\theta(\cdot \mid \mathcal{H}_t, t, T)$ ▷ sample next query pair from policy
      $l_t = \boldsymbol{x}_{t,1} \succ \boldsymbol{x}_{t,2}$ ▷ comparison of $y_{t,1}$ and $y_{t,2}$ sampled *via* (Equation 4)
      $r_t = \max_{1\le i\le t}\max\{y_{t,1}, y_{t,2}\}$ ▷ collect immediate reward
      $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \cup \{(\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, l_t)\}$ ▷ update history
      $Q \leftarrow Q \backslash \{(\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2})\}$ ▷ update query set
      infer prediction target preference $\bar{l}^{(\text{tarpred})}$ ▷ auxiliary prediction task
      $\mathcal{D}^{(p)} \rightarrow \mathcal{D}^{(\text{ctxpred})} \cup \mathcal{D}^{(\text{tarpred})}$ ▷ update prediction dataset
    end for
    Update PABBO using $\mathcal{L} := \mathcal{L}^{(q)} + \lambda\mathcal{L}^{(p)}$ (Equations 3 and 5)
  end for

4 Related Work

Preferential Black-Box Optimization. While PBBO has been tackled using techniques like dueling bandits (Bengs et al., 2021) or Reinforcement Learning (Myers et al., 2023; Rafailov et al., 2024), the bulk of the work points to the Preferential Bayesian Optimization field. The latter mostly hinges on an approximate GP surrogate, with vastly different results depending on the approximate inference scheme (Takeno et al., 2023), and several dedicated acquisition functions have been proposed. These include dueling Thompson sampling, or EUBO, a decision-theoretic AF (Lin et al., 2022). Unlike

standard BO, deriving regret bounds for PBO is cumbersome. Only one work has done so (Xu et al., 2024), although convergence can be demonstrated, e.g. for dueling Thompson sampling (Astudillo et al., 2024). Lastly, Bayesian Neural Networks were also trialed (Huang et al., 2023a).

PBO has been extended to various settings: handling a batch of queries $q > 2$ (Siivola et al., 2021; Astudillo et al., 2023), heteroscedastic noise (Sinaga et al., 2024) and multi-objective optimization (Astudillo et al., 2024). On another note, preferential relations $\boldsymbol{x}_1 \succ \boldsymbol{x}_2$ have been leveraged to enhance vanilla BO loops: Hvarfner et al. (2024) investigate a user-defined prior that is integrated to the GP surrogate, whereas Adachi et al. (2024) allow the prior to be iteratively updated.

Amortization and Meta-Learning. Our architecture is closely related to Conditional Neural Processes (CNPs) (Garnelo et al., 2018; Kim et al., 2019; Huang et al., 2023b; Jha et al., 2022), and its more recent variant, Transformer-based Neural Processes (Nguyen & Grover, 2022; Müller et al., 2022). CNPs have primarily focused on amortizing the predictive posterior distribution via supervised learning on a collection of datasets, whilst our approach employs reinforcement learning to train an optimization policy.

In recent years, multiple works have explored training neural networks to directly amortize black-box optimization (Chen et al., 2017; Yang et al., 2024; Maraval et al., 2023; Amos, 2023; Song et al., 2024; Müller et al., 2023; Chen et al., 2022). These methods typically rely on direct function utilities for training. Recent advancements also include the use of large language models to assist in Bayesian Optimization (Liu et al., 2024; Chen et al., 2024). However, to the best of our knowledge, no existing work has yet attempted to directly amortize optimization based on preferential feedback data.

Besides, since PBBO can be viewed as a sequential experimental design (SED) problem (Rainforth et al., 2024), our work is also related to research on amortized SED. This line of work primarily revolves around Deep Adaptive Design (DAD; Foster et al. 2021), whose key contribution is the development of a comprehensive framework to amortize experimental design. Subsequent work has extended this approach to settings with unknown likelihoods (Ivanova et al., 2021) and has leveraged reinforcement learning to further improve performance (Blau et al., 2022; Lim et al., 2022). Our work can be seen as a specialized application of SED with specific downstream objectives.

Preference tuning for language models. Finally, recent advancements in preference learning have further expanded its application in tuning the language models, particularly in tasks such as reinforcement learning with human feedback (RLHF) to align large-scale models with user preferences. Christiano et al. (2017) proposes a framework for guiding policy optimization in RL with human preference feedback, which has been applied to several natural language tasks (Zaheer et al., 2017; Stiennon et al., 2020). While those works focus on relatively small models and specific tasks, InstructGPT (Ouyang et al., 2022) extends it to large-scale models like GPT-3, allowing alignment with user instructions across a wide range of tasks. Recently, Rafailov et al. (2024) proposes Direct Preference Optimization (DPO), which shows that language models can be directly trained on preference data without explicit reward modeling or RL optimization, significantly simplifying the preference learning pipeline. While PABBO and these works both leverage preference feedback, they serve fundamentally different purposes: language model alignment aims to steer model behavior towards desired outputs in text spaces, whereas PABBO focuses on efficient optimization of black-box functions, requiring different architectural designs and optimization strategies.

5 EXPERIMENTS

We evaluate PABBO on synthetic functions, described in Section 5.1, and against real-world examples from the BO literature: hyperparameter optimization (Section 5.2) and human preferences datasets (Section 5.3). Subsequently, Section 5.4 presents a series of ablation studies: replacing the latent values in the rewards by pure rankings, examining different discount factor values γ in Equation 3, and varying the number of query locations on which the acquisition function values are predicted.

Baselines. Our approach is benchmarked against three GP-based acquisition strategies from the PBO realm: qEUBO (Astudillo et al., 2023), qEI (Siivola et al., 2021) and qTS (Hernández-Lobato et al., 2017). We also include a strategy where candidate pairs are acquired at random. While qEUBO is a principled preferential acquisition strategy, qEI and qTS are effectively batch BO strategies used in a PBO setting, a solution commonly employed in the field. For instance, qTS samples two draws from the posterior, maximizes each draw, and provides each maximizer as a candidate pair for comparison.

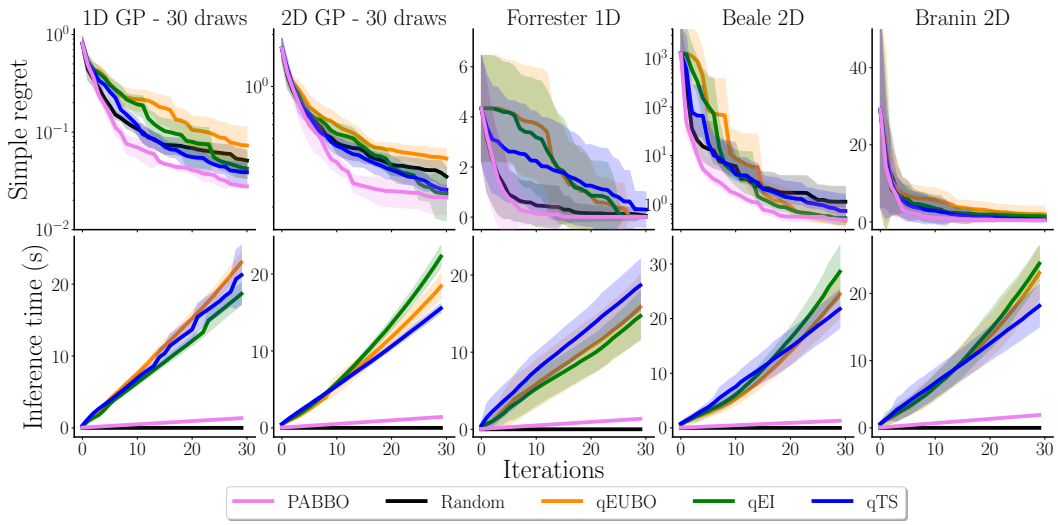

Figure 3: Simple regret and inference time on synthetic examples. Mean $\pm$ std computed across 5 runs with random starting pairs for GP examples, 30 runs for the others. **PABBO consistently achieved the lowest simple regret across all tasks while offering a 10$\times$ speedup.**

As performance metric, we consider the simple regret $f(\boldsymbol{x}^*) - f(\boldsymbol{x}_T^{\text{best}})$ where $\boldsymbol{x}_T^{\text{best}} = \arg\max_{t \in \{1,\dots,T\}} f(\boldsymbol{x}_t)$. For pre-training, the (discounted) cumulative simple regret was used (Equation 3), sending a stronger and less sparse signal than simple regret. But because our goal is human preferences optimization, e.g., finding the *best* sushi product for a given user, we report simple regret at inference time. However, for completeness, Figure 7 reports cumulative simple regret.

Implementation. PABBO is implemented using `PyTorch` (Paszke et al., 2019). Hyperparameter settings can be found in Supplementary Section A.5. For qEUBO, qEI and qTS, we used the implementation from the `BoTorch` library (Balandat et al., 2020). Code will be available upon acceptance.

5.1 SYNTHETIC FUNCTIONS

We begin by benchmarking PABBO on two types of synthetic functions. The first type involves 30 draws from the same GPs that were used to build the pre-training dataset (See Supplementary Section A.3 for the data generation process). We refer to this case as the *in-distribution*. The second type refers to classical test functions: the Forrester, Branin, and Beale functions. These functions were not part of the pre-training dataset used by PABBO, hence we refer to this case as *out-of-distribution*.

At each training epoch, we generate B different GP samples, to which a quadratic bowl is added to increase the chances of a draw exhibiting a global optimum. Then, we draw $200D$ points, where D is the dimension of $\boldsymbol{x}$, half of which for the auxiliary prediction task and the rest for the optimization task. For synthetic functions, the query set size was set to $S = 256$.

Results are displayed in Figure 3 and clearly illustrate the superiority of PABBO, against GP-based alternatives in terms of simple regret. While this is expected for the *in-distribution* case, given the low dimensionality of the problems compared to our model architecture, in the *out-of-distribution* setting, PABBO also achieves the lowest simple regret across all three examples. Of note, the random strategy often outperforms certain GP baselines. Regarding inference speed, PABBO offers an average 12-fold speed-up over the fastest GP-based strategy across 5 problems, consistently over the whole trial.

Additionally, PABBO successfully recovers at least one of the three global optima of the Branin function (Figure 17, left panel), and correctly identifies the global optimum of the Forrester function (right panel). Beyond convergence to an optimum, PABBO also accurately learned the shape of each function from preference data, a modality that only allows identification up to a monotonous transformation.

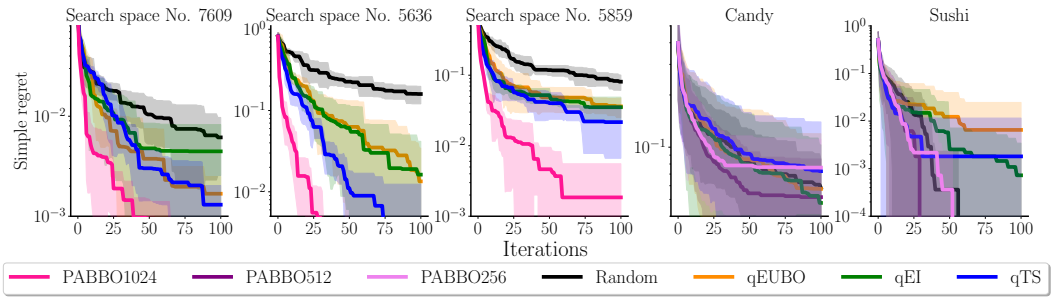

Figure 4: Simple regret on different search spaces from the HPO-B benchmark and human preferences datasets. Mean $\pm$ std computed across 30 runs with random starting pairs. Attached to each PABBO baseline is a number corresponding to S, the size of the query set. **PABBO significantly outperformed all the baselines on HPO-B tasks and ranks first for human preferences datasets.**

5.2 HYPERPARAMETER OPTIMIZATION

Next, we consider hyperparameter optimization (Pineda Arango et al., 2021). This challenge emulates the task an ML expert would face when optimizing the hyperparameter of a given model, performing comparisons rather than assigning a scalar utility to each hyperparameter set. The HPO-B benchmark contains multiple search spaces, each corresponding to the optimization of different ML models.

Each search space contains multiple meta-datasets from different meta-tasks, and each meta-dataset collects a list of hyperparameter configurations with their response values, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, allowing us to simulate preference for any two configurations given their responses. All the meta-datasets within a search space have been categorized into three splits: meta-train, meta-validation, and meta-test.

We choose three representative search spaces and pre-train our model on the meta-train split associated with each space, further splitting each set into two equal-sized, non-overlapping parts beforehand for the auxiliary prediction and optimization tasks (see Supplementary Section A.7). For evaluation, we draw $S = 1024$ points from each meta-datasets of the meta-test split of the search space, allowing us to assess the average performance of our model and the baselines across all meta-datasets.

In all examples, PABBO improves over GP-based methods by a significant margin (Figure 4, first three panels), even reaching convergence three times as fast than qTS, the best GP-based strategy, for search spaces No. 7609 and 5636. It is worth noticing that on these higher-dimensional problems, all methods now clearly outperform the random acquisition strategy.

5.3 HUMAN PREFERENCES

We experiment with our model pre-trained with synthetic GP samples on two real-world human preference datasets: the Candy[2] and Sushi[3] datasets. The Candy dataset provides the full ranking of 85 different candies from pairwise preference via online studies. Each candy object contains 2 continuous features, `sugarpercent` and `pricepercent`. The Sushi dataset collects a preference score by asking users to rate 100 different types of sushi on a five-point-scale. The overall score for each sushi is obtained by averaging across users. Each sushi contains 4 continuous features. Following Siivola et al. (2021), we generalize both datasets to continuous space by performing linear extrapolation, and the out-of-bound values are filled with their nearest neighbors in the original dataset.

For both examples, we report the simple regret achieved by our model when using a query set of size $S \in \{256, 512\}$. For the Candy dataset, even though the problem is only 2-dimensional, we observe a nonnegligible difference for different values of S, switching from worst (PABBO 256) to best (PABBO 512) performance. (Figure 4, fourth panel). Results are more consistent for the Sushi dataset. Both problems are characterized by a large variance in the results for all baselines.

5.4 ABLATION STUDY

This last experimental section investigate whether certain modeling choices and hyperparameter values may have a significant impact on the performance of PABBO .

[2] Available at data/candy-power-ranking/ [3] Available at https://www.kamishima.net/sushi/

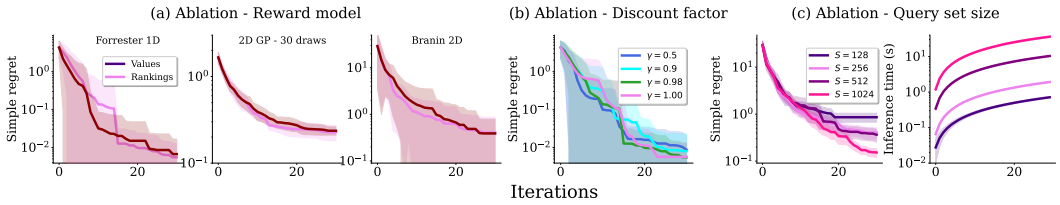

Figure 5: Ablation studies. mean $\pm$ std averaged over 30 runs with random starting pairs, except for the bottom second panel, where for visibility concerns, we show mean± 0.25std. **On low dimensional examples, PABBO maintains similar performance when substituting latent function values for rankings, and performs as expected when varying the discount factor γ and query set size S.**

Access to latent values during pre-training. Even though latent values are always accessible for synthetic datasets used for pre-training, this might not be the case for real-world preferential meta datasets. A simple solution for proposing rewards without access to the true utility is to use rankings from a fully ranked set to approximate the latent function values. We trained a 1-and-2-dimensional models using a pre-training dataset GP samples, substituting classical rewards for each meta-task by the rankings of corresponding samples. On the 1-dimensional Forrester function, this alternative version of PABBO still yields decent performances (Figure 5(a)).

Discount factor γ. PABBO learns a policy π_θ by taking the future discounted rewards into account across a time horizon T. A smaller discount factor prioritizes immediate queries, giving more weight to the early steps in the optimization process. In contrast, as γ approaches 1, the future queries become increasingly important, reflecting a more balanced consideration of both immediate and future rewards. We test with different discount factors on the 1-dimensional Forrester problem. The results in Figure 5(b) align with our expectation. When γ is small, we observe faster convergence in the early stages. However, since the future rewards are not sufficiently considered, the average performance over larger steps is less favorable compared to using a larger γ, which balances both immediate and future rewards, resulting in better long-term optimization outcomes.

Query set size S. At inference time, a continuous search space is discretized into S query locations sampled from a quasi-random Sobol sequence, used to build a set of candidate query pairs consisting of all possible pairwise combinations, resulting in $O(S^2)$ query pairs. S introduces a trade-off between performance and inference time: a denser grid guarantees adequate exploration of the search space, but longer inference times. For the 2-dimensional Branin problem, increasing the grid size $S \in [128, 256, 512, 1024]$ leads to increased inference time, a consequence of the need to embed an increasing number of candidate pairs (Figure 5(c)). In terms of simple regret, a grid of size $S = 1024$ achieves the best result, whereas $S = 256$ offers the best speed/accuracy tradeoff.

6 DISCUSSION

We introduced PABBO, a method to conduct preferential black-box optimization in a fully amortized manner. Leveraging an offline training phase on both synthetic and existing datasets, PABBO achieves considerably reduced inference time jointly with top-ranking performances compared to GP alternatives, based on our benchmarks. One way to explain this gap is that our pre-training dataset contains latent function values, which are always accessible for synthetic datasets, thus giving PABBO a significant advantage over preferential GPs. This being said, a dedicated ablation study demonstrated that replacing latent function values with pure rankings did not significantly affect PABBO's results.

Due to the model architecture and the difficulty of the task, learning to optimize from preferential data, PABBO requires large amounts of pre-collected data. While synthetic datasets might be expressive enough, specific applications falling *out-of-distribution* might pose an issue. More generally, this raises the question of how the pre-training dataset should be constructed. Moreover, PABBO is currently limited to handling tasks with the same input dimensionality. For tasks with varying dimensions, separate models would need to be trained for each case. A promising direction for future work would be developing a dimension-agnostic architecture capable of accepting inputs of different dimensions, thus enhancing PABBO 's generalizability. Finally, PABBO directly outputs acquisition values for a candidate pair, modeling the user's latent goal, but does not account for the fact that ultimately, a human is performing the comparison. Introducing dedicated user models like the one presented by Sinaga et al. (2024) could be an interesting extension to this work.

REFERENCES

Mahmoud M Abdelrahman and Clayton Miller. Targeting occupant feedback using digital twins: Adaptive spatial–temporal thermal preference sampling to optimize personal comfort models. *Building and Environment*, 2022.

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

Brandon Amos. Tutorial on amortized optimization. *arXiv preprint arXiv:2202.00665*, 2023.

Raul Astudillo, Zhiyuan Jerry Lin, Eytan Bakshy, and Peter Frazier. qeubo: A decision-theoretic acquisition function for preferential bayesian optimization. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, 2023.

Raul Astudillo, Kejun Li, Maegan Tucker, Chu Xin Cheng, Aaron D. Ames, and Yisong Yue. Preferential multi-objective bayesian optimization. *arXiv preprint arXiv:2406.14699*, 2024.

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BoTorch: A framework for efficient monte-carlo bayesian optimization. In *Advances in Neural Information Processing Systems*, 2020.

Viktor Bengs, Robert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: A survey. *Journal of Machine Learning Research*, 2021.

Tom Blau, Edwin V Bonilla, Iadine Chades, and Amir Dezfouli. Optimizing sequential experimental design with deep reinforcement learning. In *International conference on machine learning*, 2022.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Guojin Chen, Keren Zhu, Seunggeun Kim, Hanqing Zhu, Yao Lai, Bei Yu, and David Z Pan. Llm-enhanced bayesian optimization for efficient analog layout constraint generation. *arXiv preprint arXiv:2406.05250*, 2024.

Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando Freitas. Learning to learn without gradient descent by gradient descent. In *International Conference on Machine Learning*, 2017.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'aurelio Ranzato, et al. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems*, 2022.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Wei Chu and Zoubin Ghahramani. Preference learning with Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, 2005.

João P.L. Coutinho, Ivan Castillo, and Marco S. Reis. Human-in-the-loop controller tuning using preferential bayesian optimization. *IFAC-PapersOnLine*, 2024.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 2020.

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

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International conference on machine learning*, 2018.

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

Manuel Gloeckler, Michael Deistler, Christian Dietrich Weilbach, Frank Wood, and Jakob H. Macke. All-in-one simulation-based inference. In *Forty-first International Conference on Machine Learning*, 2024.

Javier González, Zhenwen Dai, Andreas Damianou, and Neil D Lawrence. Preferential bayesian optimization. In *International Conference on Machine Learning*, 2017.

José Miguel Hernández-Lobato, James Requeima, Edward O Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. In *International conference on machine learning*, 2017.

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*, 2021.

Daolang Huang, Louis Filstroff, Petrus Mikkola, Runkai Zheng, Milica Todorovic, and Samuel Kaski. Augmenting bayesian optimization with preference-based expert feedback. In *ICML 2023 Workshop The Many Facets of Preference-Based Learning*, 2023a.

Daolang Huang, Manuel Haussmann, Ulpu Remes, ST John, Grégoire Clarté, Kevin Luck, Samuel Kaski, and Luigi Acerbi. Practical equivariances via relational conditional neural processes. *Advances in Neural Information Processing Systems*, 2023b.

Carl Hvarfner, Frank Hutter, and Luigi Nardi. A general framework for user-guided bayesian optimization. In *The Twelfth International Conference on Learning Representations*, 2024.

Desi R Ivanova, Adam Foster, Steven Kleinegesse, Michael U Gutmann, and Thomas Rainforth. Implicit deep adaptive design: Policy-based experimental design without likelihoods. *Advances in neural information processing systems*, 2021.

Saurav Jha, Dong Gong, Xuesong Wang, Richard E Turner, and Lina Yao. The neural process family: Survey, applications and perspectives. *arXiv preprint arXiv:2209.00517*, 2022.

Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. In *Handbook of the fundamentals of financial decision making: Part I*. World Scientific, 2013.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019.

Yuki Koyama, Issei Sato, and Masataka Goto. Sequential gallery for interactive visual design optimization. *ACM Transactions on Graphics (TOG)*, 2020.

Kejun Li, Maegan Tucker, Erdem Bıyık, Ellen Novoseller, Joel W Burdick, Yanan Sui, Dorsa Sadigh, Yisong Yue, and Aaron D Ames. Roial: Region of interest active learning for characterizing exoskeleton gait preference landscapes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Vincent Lim, Ellen Novoseller, Jeffrey Ichnowski, Huang Huang, and Ken Goldberg. Policy-based bayesian experimental design for non-differentiable implicit models. *arXiv preprint arXiv:2203.04272*, 2022.

Zhiyuan Jerry Lin, Raul Astudillo, Peter Frazier, and Eytan Bakshy. Preference exploration for efficient bayesian optimization with multiple outcomes. In *International Conference on Artificial Intelligence and Statistics*, 2022.

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

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*, 2023.

Stratis Markou, James Requeima, Wessel Bruinsma, Anna Vaughan, and Richard E Turner. Practical conditional neural process via tractable dependent predictions. In *International Conference on Learning Representations*, 2022.

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

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

Vivek Myers, Erdem Bıyık, and Dorsa Sadigh. Active reward learning from online preferences. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In *International Conference on Machine Learning*, 2022.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 2019.

Sebastian Pineda Arango, Hadi Jomaa, Martin Wistuba, and Josif Grabocka. Hpo-b: A large-scale reproducible benchmark for black-box hpo based on openml. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Tom Rainforth, Adam Foster, Desi R Ivanova, and Freddie Bickford Smith. Modern bayesian experimental design. *Statistical Science*, 2024.

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

Eero Siivola, Akash Kumar Dhaka, Michael Riis Andersen, Javier González, Pablo García Moreno, and Aki Vehtari. Preferential batch bayesian optimization. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, 2021.

Marshal Arijona Sinaga, Julien Martinelli, Vikas Garg, and Samuel Kaski. Heteroscedastic preferential bayesian optimization with informative noise distributions. *arXiv preprint arXiv:2405.14657*, 2024.

Lei Song, Chenxiao Gao, Ke Xue, Chenyang Wu, Dong Li, Jianye Hao, Zongzhang Zhang, and Chao Qian. Reinforced in-context black-box optimization. *arXiv preprint arXiv:2402.17423*, 2024.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Kevin Swersky, Yulia Rubanova, David Dohan, and Kevin Murphy. Amortized bayesian optimization over discrete spaces. In *Conference on Uncertainty in Artificial Intelligence*, pp. 769–778. PMLR, 2020.

Shion Takeno, Masahiro Nomura, and Masayuki Karasuyama. Towards practical preferential Bayesian optimization with skew Gaussian processes. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

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*, 2020.

Wenjie Xu, Wenbin Wang, Yuning Jiang, Bratislav Svetozarevic, and Colin Jones. Principled preferential bayesian optimization. In *Forty-first International Conference on Machine Learning*, 2024.

Adam X. Yang, Laurence Aitchison, and Henry Moss. MONGOOSE: Path-wise smooth bayesian optimisation via meta-learning. In *ICML 2024 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2024.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 2017.

A APPENDIX

A.1 FURTHER DETAILS ON PABBO

A.1.1 ALGORITHM FOR EMPLOYING PABBO AT TEST TIME

Algorithm 2 Executing PABBO at test time on a new task

Require: Query set size S, query budget T, initial observations $\mathcal{H}_0$ (possibly empty)
  Initialize $\mathcal{D}^{(q)}$ with S quasi-random samples
  $Q = \{(\boldsymbol{x}_i, \boldsymbol{x}_j)\ \forall (i,j) \in \{1,\dots,S\}^2, i \neq j\}$
  for $t = 1, \dots, T$ **do**
    $\mathcal{D}^{(c)} \leftarrow \mathcal{H}_t$ ▷ update context set
    $\mathbf{E}^{(c)} = f_{\text{emb}}(\mathcal{D}^{(c)}), \mathbf{E}^{(q)} = f_{\text{emb}}(\mathcal{D}^{(q)})$ ▷ embed context set and query set
    $\lambda^{(q)} = f_{\text{tfm}}(\mathbf{E}^{(q)}; \mathbf{E}^{(c)})$ ▷ Transformers output for query set conditioning on context set
    $\mathbf{q} = \{f_a(\lambda_i^{(q)}, \lambda_j^{(q)}, t, T)\ \forall (\boldsymbol{x}_i, \boldsymbol{x}_j) \in Q\}$ ▷ predict acquisition function values for all query pairs
    $\pi_\theta(Q \mid \mathcal{H}_t, t, T) = \text{Categorical}(\mathbf{q})$ ▷ form policy as a Categorical distribution
    $(\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}) \sim \pi_\theta(Q \mid \mathcal{H}_t, t, T)$ ▷ sample next query pair from policy
    $l_t = \boldsymbol{x}_{t,1} \succ \boldsymbol{x}_{t,2}$ ▷ observe binary preference
    $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \cup \{(\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, l_t)\}$ ▷ update history
    $Q \leftarrow Q \backslash \{(\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2})\}$ ▷ update query set
  end for

A.1.2 ON THE TRANSFORMER ARCHITECTURE

PABBO employs *separate* MLPs for $\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}$ and l_i, mapping each of them to *separate* embeddings of Transformer input dimension as $\mathbf{E}^{\boldsymbol{x}_{i,1}}$, $\mathbf{E}^{\boldsymbol{x}_{i,2}}$, and $\mathbf{E}^{(l_i)}$, with $\oplus$ denoting the component-wise addition. When working on the context set $\mathcal{D}^{(c)}$ and prediction context set $\mathcal{D}^{(\text{ctxpred})}$, which consists of duels $(\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, l_i)$, the token embedding is obtained by adding the individual embeddings up, i.e. $\mathbf{E} = \mathbf{E}^{\boldsymbol{x}_{i,1}} \oplus \mathbf{E}^{\boldsymbol{x}_{i,2}} \oplus \mathbf{E}^{(l_i)}$. When processing a single design from the query set $\mathcal{D}^{(q)}$ or prediction target set $\mathcal{D}^{(\text{tarpred})}$, we directly use the individual embedding for that design as the token embedding, i.e., $\mathbf{E} = \mathbf{E}^{\boldsymbol{x}_{i,1}}$. Because $(\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, l_i)$ forms a single atomic element in our context set, Transformer architecture requires all elements in its input sequence to live in the same embedding space: to pass these preference observations to the transformer, we need to make sure their dimensions are aligned.

Given that PABBO encodes elements separately, for the same type of elements, like $\boldsymbol{x}$, a shared embedder can be used across all three sets. This enables parameter sharing for the embedders, as the same embedder processes all $\boldsymbol{x}$-type input, regardless of which set they belong to. Model complexity benefits from this choice as the need to train separate embedding functions for each set is avoided. Besides, separate embeddings confer distinct semantic meanings to features $\boldsymbol{x}$ and preference labels l. Thus, separate MLPs can specialize in their respective tasks: feature embedders can focus on capturing spatial relationships, and preference embedders can focus on binary relationship encoding. This leads to an increased representation power.

On a similar note, one could have settled for using the same embedding for $\boldsymbol{x}_{i,1}$ and $\boldsymbol{x}_{i,2}$. However, in the particular case of preferential feedback, upon summing $\mathbf{E}^{\boldsymbol{x}_{i,1}}$ and $\mathbf{E}^{\boldsymbol{x}_{i,2}}$ the order information would have been lost: the network would not know whether $\boldsymbol{x}_{i,1} > \boldsymbol{x}_{i,2}$, or the opposite. Therefore, we opted for separate embedders for $\mathbf{E}^{\boldsymbol{x}_{i,1}}$ and $\mathbf{E}^{\boldsymbol{x}_{i,2}}$.

A.2 Additional Experiments

A.2.1 Ablation on the pre-training synthetic dataset composition

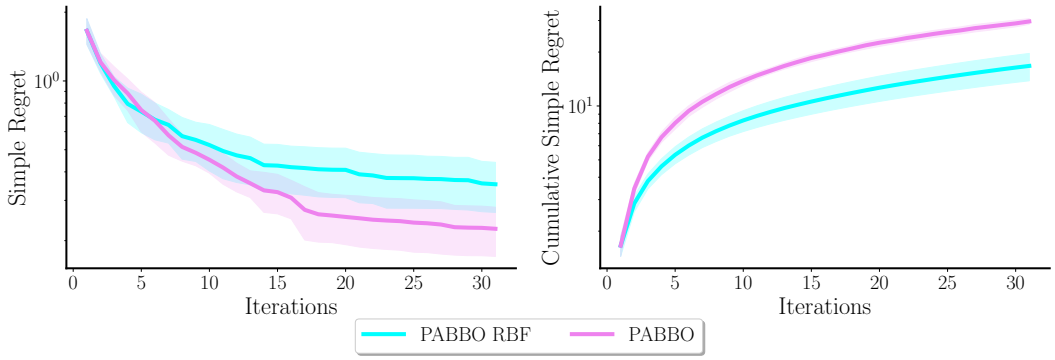

Figure 6: Ablation study on the composition of the pre-training synthetic dataset on 30 2D GP draws, with a query set size $S = 256$. Mean $\pm$ std computed across 5 runs with random starting pairs. PABBO RBF is pre-trained on GP draws sampled using an RBF kernel, whereas PABBO uses the procedure described in Section A.3. Both baselines use lengthscales $l \sim \mathcal{U}([1.5, 2])$. **As expected, pre-training on a dataset coming from a mixture of kernels rather than only a RBF kernel yields better results, due to a more pronounced diversity of the synthetic samples.**

A.2.2 Cumulative simple regret

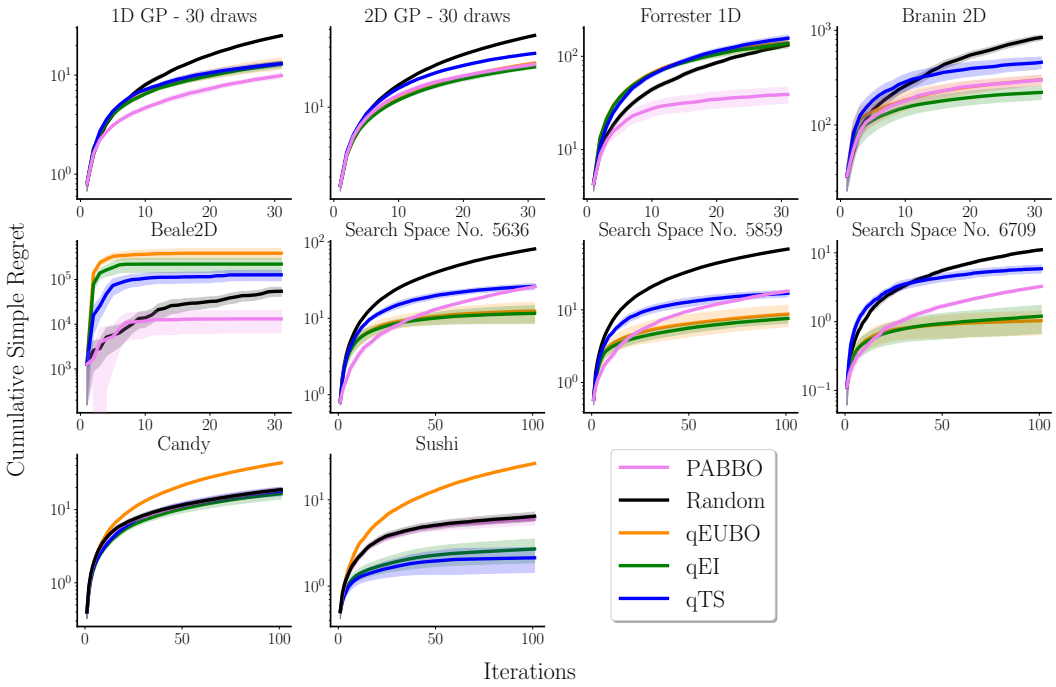

Figure 7: Cumulative simple regret plots for all experiments carried out in the main text. **When averaged over all tasks, PABBO ranks close to the second position.**

A.2.3 ABLATION STUDY ON LOSS WEIGHT HYPERPARAMETER λ

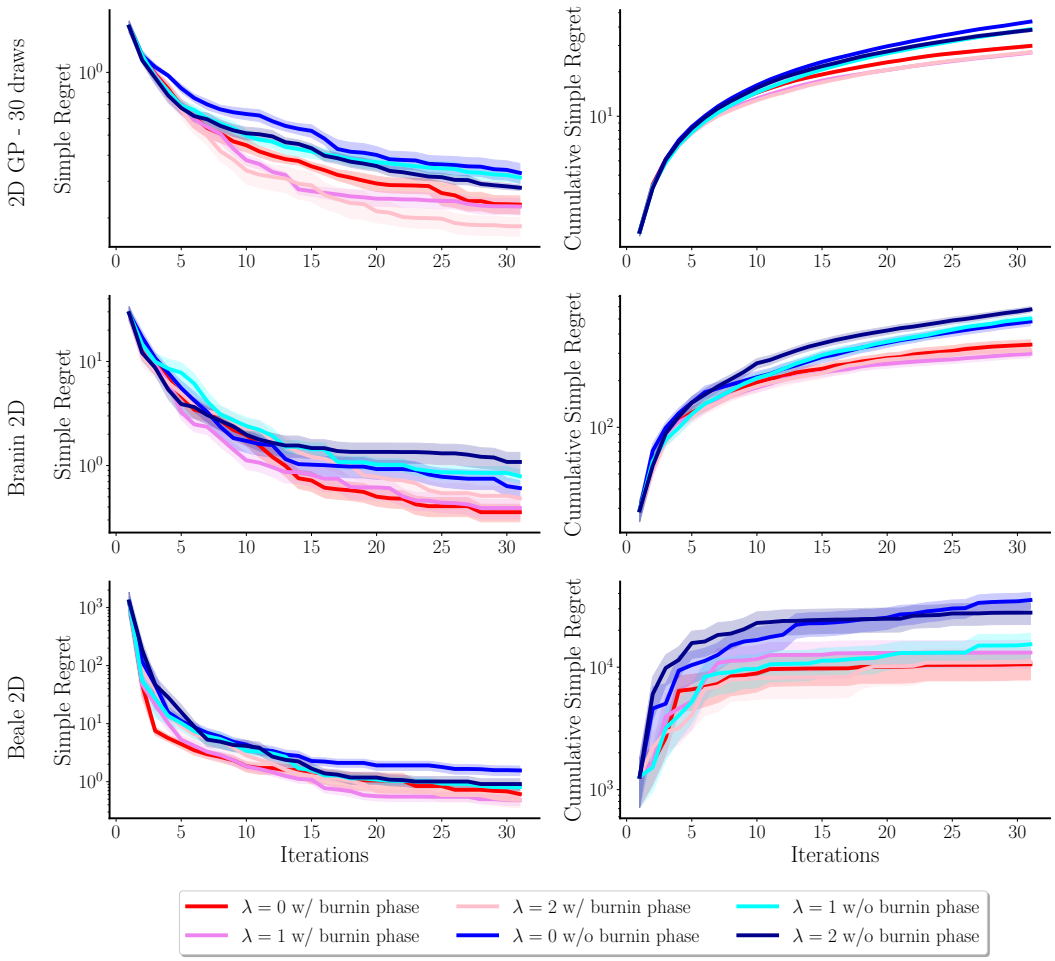

Figure 8: Ablation study on the loss weight hyperparameter λ in $\mathcal{L} = \mathcal{L}^{(q)} + \lambda\mathcal{L}^{(p)}$ (Equations 3 and 5) on three 2D functions, with a query set size $S = 256$. Mean $\pm$ std computed across 5 runs with random starting pairs for GP draws, 30 for the others. During the warm-up phase, PABBO is trained using only the prediction task $\mathcal{L}^{(p)}$. **We observe a stark contrast depending on the presence or not of the warm-up phase, which validates the usefulness of the prediction task, mainly as a tool to stabilize training. Results are mildly affected by different values of λ.**

A.2.4 ABLATION STUDY ON THE DISCOUNT FACTOR γ WITH A LARGER GRID

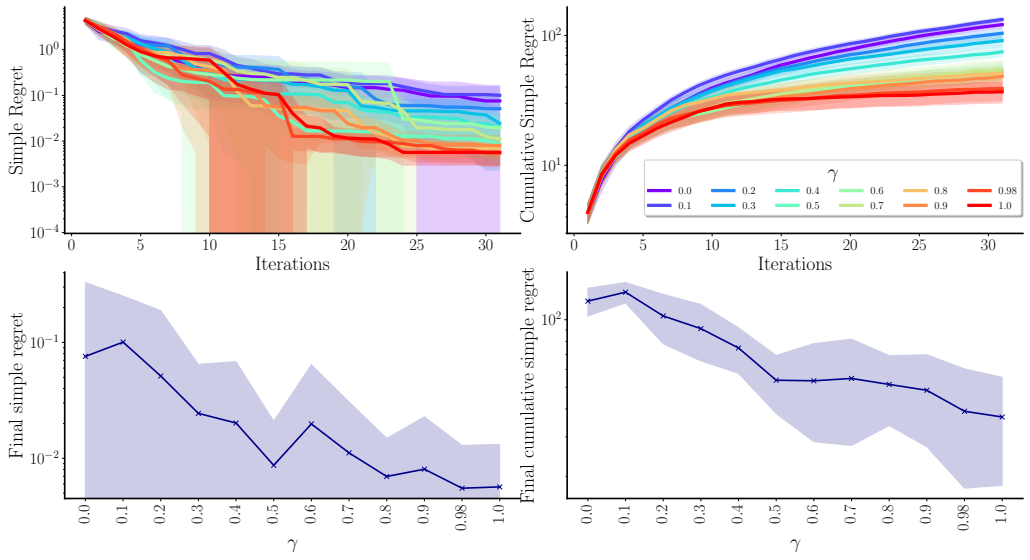

Figure 9: Ablation Study on the discount factor γ (Equation 3) on the 1D Forrester function, with a query set size $S = 256$. Mean $\pm$ std computed across 30 runs with random starting pairs. **As expected, an higher value of γ leads to a lower cumulative simple regret. The picture is more nuanced for simple regret, even though as we progress towards a large number of iterations, the conclusion appears to be similar.**

A.2.5 HIGH DIMENSIONAL SEARCH SPACE

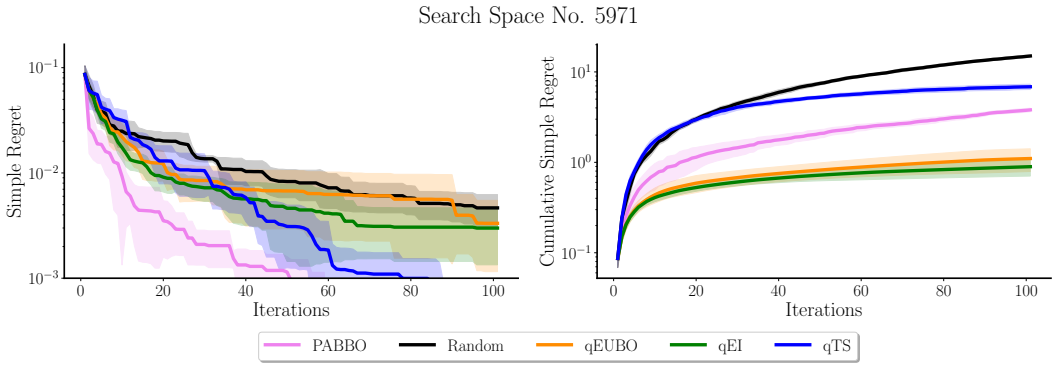

Figure 10: Simple regret and cumulative simple regret on a 16-dimensional search space from the HPO-B benchmark. Mean $\pm$ std computed across 30 runs with random starting pairs. We used a query set size $S = 1024$ for PABBO. **PABBO significantly outperforms all the baselines and ranks 3rd in terms of cumulative simple regret.**

A.2.6 MULTINOMIAL PREDICTIVE ENTROPY SEARCH BASELINE

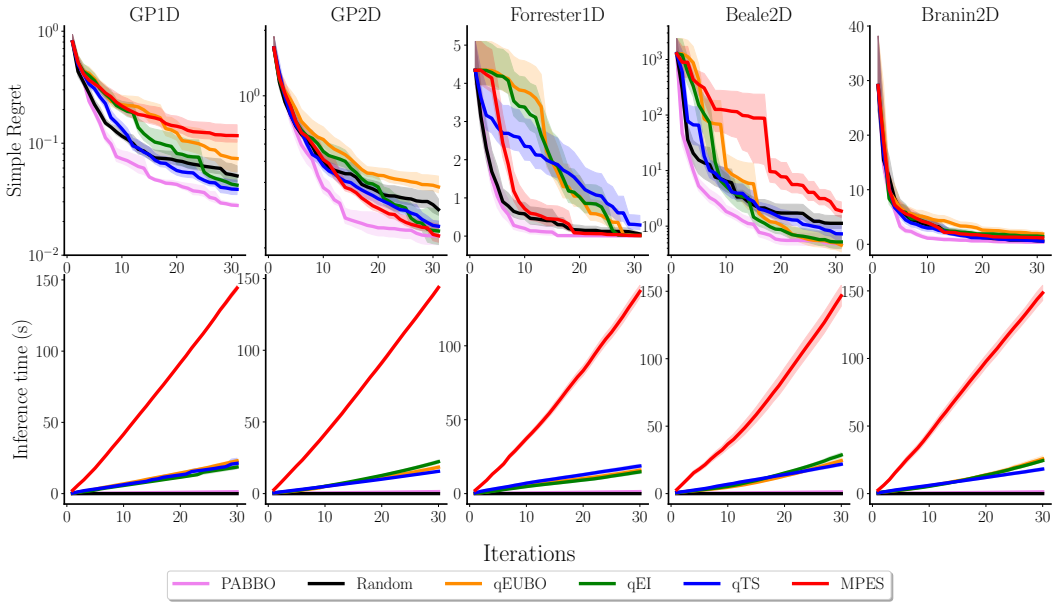

Figure 11: Simple regret and inference time on synthetic examples, with additional baseline Multinomial Predictive Entropy Search (MPES). Mean ± std computed across 5 runs with random starting pairs for GP examples, 30 runs for the others. **PABBO consistently achieved the lowest simple regret across all tasks while offering a 10× speedup. MPES is an order of magnitude slower than other GP-based strategies.**

A.2.7 ACKLEY 6D FUNCTION

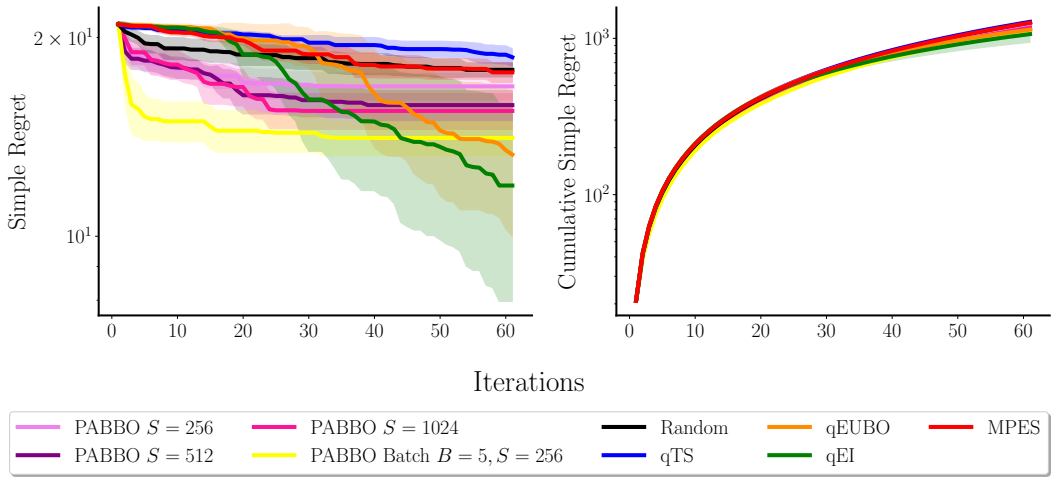

Figure 12: Simple regret and cumulative simple regret on the 6-dimensional Ackley function. Mean ± std computed across 10 runs with random starting pairs. **Using parallelization, we can mimic a larger query set size for PABBO , thus being able to closely follow the top ranking baselines qEUBO and qEI. Additionally, "batch" PABBO (13s) achieves significantly faster cumulative inference time compared to PABBO1024 (40s).**

A.2.8 ABLATION ON THE QUERY BUDGET T

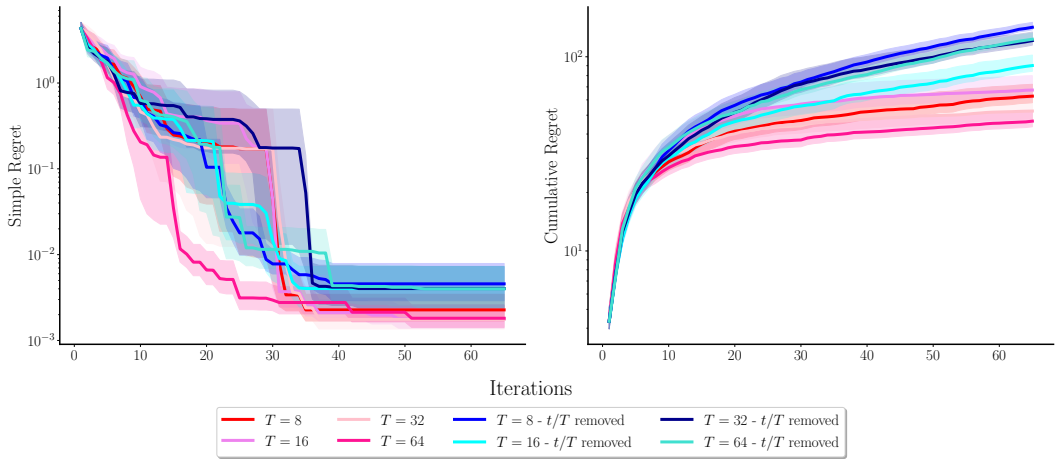

Figure 13: Simple regret and cumulative simple regret on the 1-dimensional Forrester function. Mean $\pm$ std computed across 30 runs with random starting pairs. **The variance is mostly driven by whether or not t/T is passed to the acquisition head or not (see Figure 2, right). When passing t/T, a large budget T is associated with the fastest convergence and the lowest simple cumulative regret.**

A.2.9 ABLATION ON THE PERCENTAGE OF CONTEXT AND TARGET PREDICTION SET

In the main experiments, we set a total number of samples for the prediction set and *randomly* sample $D^{(\text{ctxpred})}$ and assign the remaining pairs as $D^{(\text{tarpred})}$. To examine the effect of the percentage of context and target prediction set, we train different PABBO for prediction task, when setting $N^{(\text{ctxpred})} = 10$ with varying $N^{(\text{tarpred})} = [1, 10, 100]$.

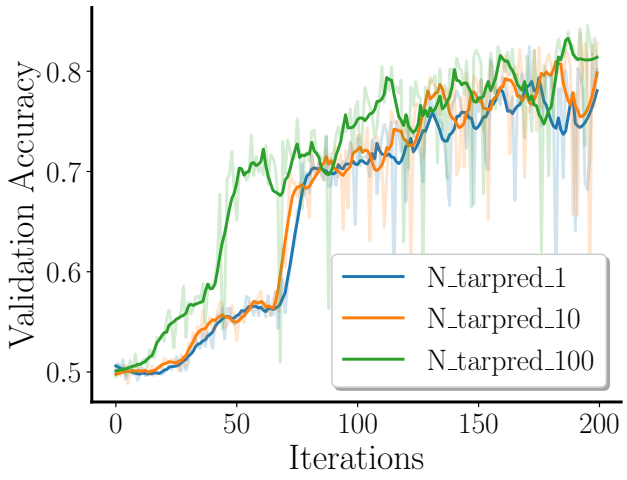

Figure 14: Validation accuracy on 128 1-dimensional GP samples. **Increasing the size of $D^{(\text{tarpred})}$ provides more loss terms during training, thereby enriching the training signal, and resulting in faster convergence of the network.**

A.3 DETAILS OF SYNTHETIC DATASET

To generate the $D-$dimensional synthetic data in Section 5.1, we first sample from a GP, where

- The kernel are equally sampled from the RBF, Matérn-$1/2$, Matérn-$3/2$ and Matérn-$5/2$ kernels, with the kernel standard deviation $\sigma \sim \mathcal{U}([0.1, 2])$ and lengthscale for each dimension $l^d \sim \mathcal{N}(1/3, 0.75)$ truncated to $[0.05, 2]$.
- Then we sample function mean as the maximal value of m observations from the Normal distribution $\mathcal{N}(0, \sigma^2)$. To account for very high optimum, $\exp(1)$ is added to the mean with a probability of 0.1.
- With the defined prior, we randomly sample an optimum $(\boldsymbol{x}^*, 0)$ inside $[-1, 1]^D$ and a total of $N-1$ context points, where $N = 100 * D$. The context set is sampled from GP posterior conditioned on the optimum by first sampling $M-1$ points conditioned on the optimum, and the rest of $N-M$ points conditioned on these points and the optimum. $M = 50$ for 1-dimensional data and 100 when $D = 2, 3, 4$.
- To ensure the existence of global optimum, we additionally add a quadratic bowl at $\boldsymbol{x}^*$.
- Finally, the function value is $y = -(|y| + \frac{1}{8^D}\|\boldsymbol{x}^* - \boldsymbol{x}\|^2 + \Delta y)$. The offset $\Delta y \sim \mathcal{U}([-5, 5])$, makes the maximal value $y^* = (0 + \Delta y) \in [-5, 5]$.

Figure 15 shows 16 randomly generated 1-dimensional synthetic functions.

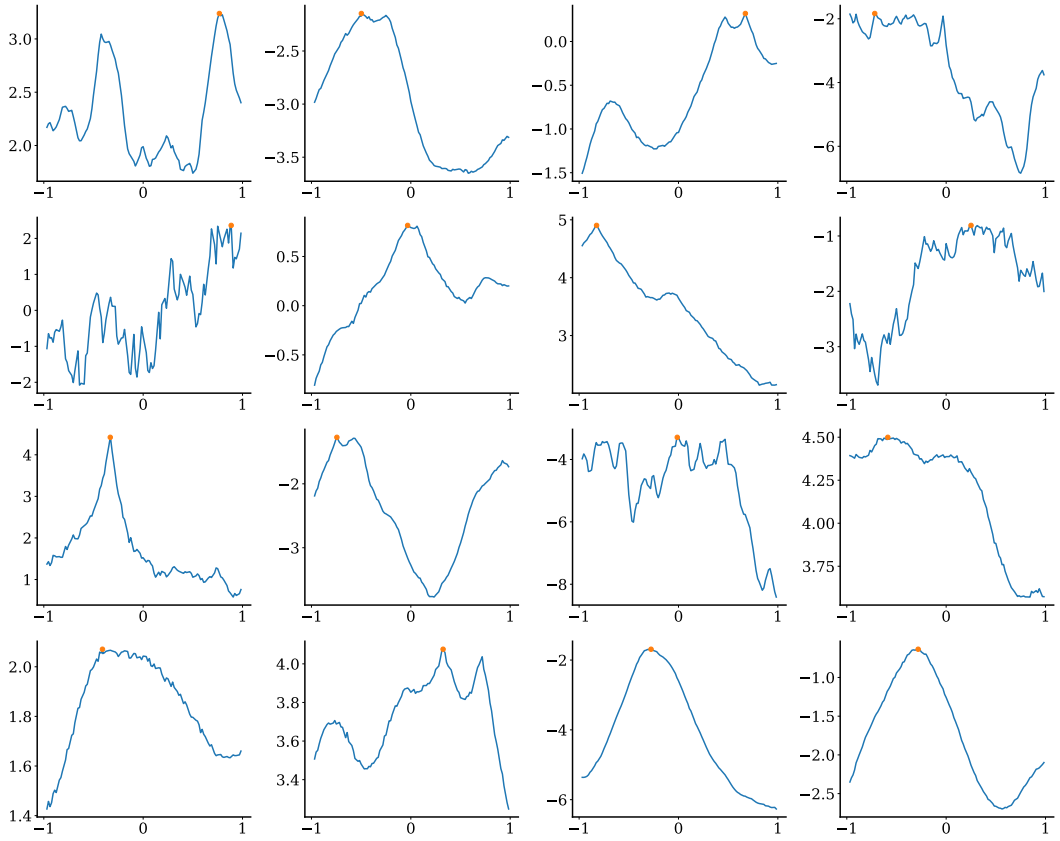

Figure 15: Example of 1-dimensional synthetic data generated using the protocol defined in Supplementary Section A.3. The red dot in each plot indicates the optimum of the sampled function.

A.4 EXAMPLE OF MASK IN SELF-ATTENTION LAYERS

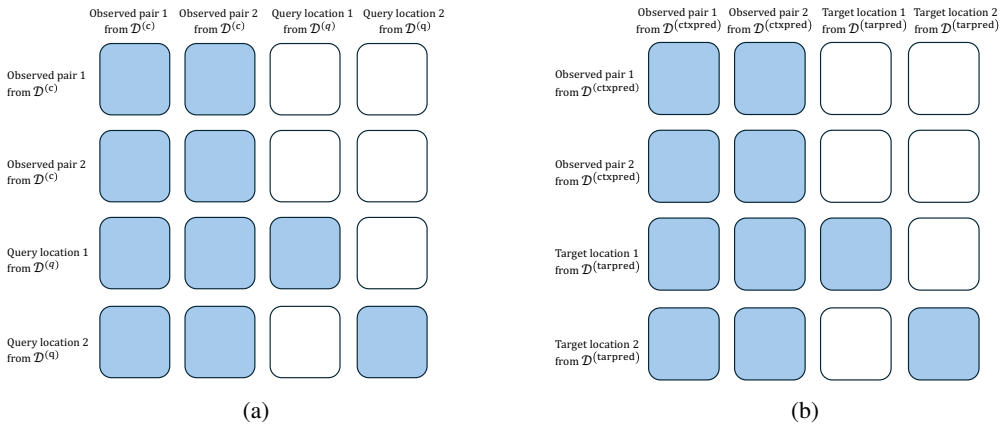

Figure 16: Examples of masks. Colored squares represent the elements from the left column can attend to the elements on the top in the self-attention layers of the Transformer block f_{ftm}. (a) An example of mask for the acquisition process, involving 2 samples from $\mathcal{D}^{(c)}$ and 2 points from $\mathcal{D}^{(q)}$. (b) Mask example for the auxiliary prediction process, with 2 samples from $\mathcal{D}^{(\text{ctxpred})}$ and 2 points from $\mathcal{D}^{(\text{tarpred})}$.

A.5 HYPERPARAMETERS

The hyperparameter setting we used for all the experiments are detailed in Table 2. Most of them remain consistent across all the tasks, and we scale up the size of meta-datasets according to the search space dimension.

A.6 HARDWARE

We train our model using up to 5 Tesla V100-SXM2-32GB GPUs, with training time of roughly 10, 25, and 29 hours for 1-, 2-, and 4-dimensional synthetic data respectively. There is significant potential to shorten this pre-training process by creating the training samples beforehand, rather than generating them online. For the HPO-B tasks on search spaces No. 5636 and No. 5859 (6 dimensions) and the search space No. 7609 (9 dimensions), training takes approximately 21.5 hours per task. We evaluate all the models on 2x64 core AMD EPYC 7713 @2.0 GHz.

A.7 HPO DATASET DESCRIPTION AND EXPERIMENTAL DETAILS

As mentioned in Section 5.2, we experimented on three high-dimensional search spaces from the HPO-B benchmarks, each corresponding to the hyperparamers of a certain model: No. 5636 for `rpart(29)`, No. 5859 for `rpart(31)`, and No. 7609 for `ranger(16)`. Both No. 5636 and No. 5859 are 6-dimensional spaces, while No. 7609 is a 9-dimensional space. Meta-datasets within each space are divided into three splits: meta-train, meta-validation, and meta-test.

An individual model is trained on the meta-train split of each search space. Each meta-dataset is equally divided beforehand into two parts from which we sample either prediction set $\mathcal{D}^{(p)}$ or query set $D^{(q)}$, so as to prevent any information leak from rewards. The only exception happens when a meta-dataset has too few data points: we fit a GP to the original data points and sample from the posterior to generate either $\mathcal{D}^{(p)}$ or $\mathcal{D}^{(q)}$. As described in Section 3.2, we sample $\mathcal{D}^{(p)}$ containing N queries and preferences and $\mathcal{D}^{(q)}$ with S query points from each meta-dataset at the beginning of each training iteration.

Model Architecture	
Embedding dimension of Transformer	64
Point-wise feed-forward dimension of Transformer	128
Number of self-attention layers in Transformer	6
Number of self-attention heads in Transformer	4
Number of hidden layers in data embedders	3
Number of hidden layers in decoder (f_p, f_a)	1
Hidden layer dimension of decoder (f_p, f_a)	128
Training	
Number of iterations for warm-up	3000
Total number of training iterations	8000
Horizon of episodes (T)	64
Learning rate for warm-up	$1 \cdot 10^{-3}$
Learning rate after warm-up	$3 \cdot 10^{-5}$
Learning rate decay	Cosine decay to 0 over 8000 iterations
Number of meta-tasks in a batch during warm-up (B)	128
Number of meta-tasks in a batch after warm-up (B)	16
Number of trajectories from one meta-task for policy learning	20
Weight on auxiliary loss (λ)	1.0
discount factor (γ)	0.98
Size of query set (S)	$\min(300, 100 \cdot D)$
Size of candidate query pair set (M)	$\min(300, 100 \cdot D)$
Size of prediction set (N)	$\min(300, 100 \cdot D)$
Maximal size of prediction context set ($N^{(\text{ctxpred})}$)	$50(D = 1), 100(1 < D \leq 4), 200(D > 4)$
Environment	
Number of initial pairs during training	0
Number of initial pairs during evaluation	1
Observation noise of duel feedback (σ_{noise})	0.0

Table 2: Hyperparameter settings

A.8 Details of baselines

The PBO baselines are implemented based on `PairwiseGP` model class from `BoTorch`. For qEUBO and qEI, `BoTorch` provides ready-to-use acquisition class `qExpectedUtilityOfBestOption` and `qExpectedImprovement`. For qTS, we sample two draws from the GP posterior at: (1) locations from quasi-random Sobol sequences for continuous search spaces, or (2) locations of all candidates for discrete search spaces. The maximum of each draw are chosen as the next query. Finally, we select a pair of random points for continuous spaces and a random pair of possible candidates for discrete spaces as a random strategy. Baseline inputs are all normalized to $(0, 1)$.

A.9 Example of inference function shape and optimum on test functions

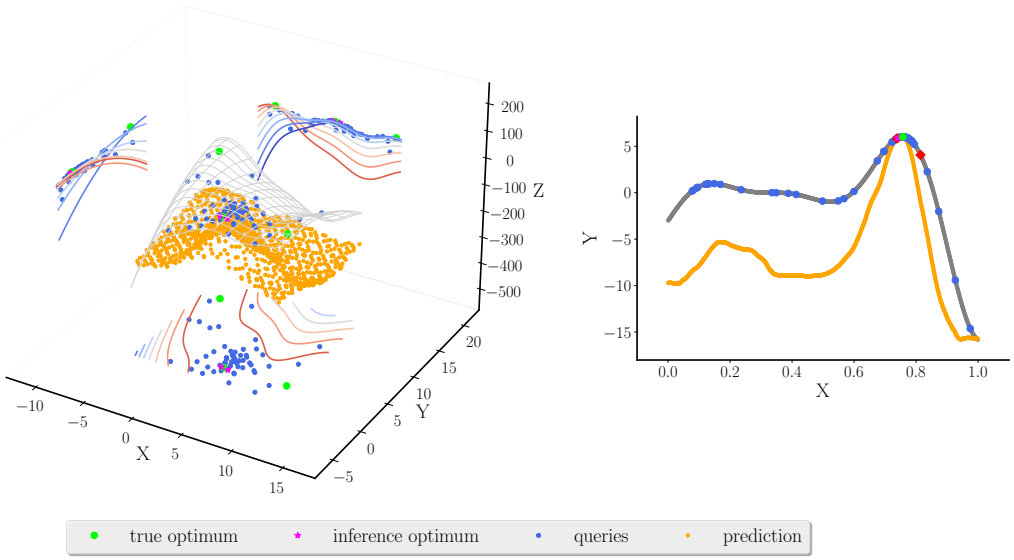

Figure 17: Optimum inference and function shape on test functions after 30 optimization steps for the $2D$ Branin function (left) and $1D$ Forrester function (right) trialed in Section 5.1.