Black-Box Optimization From Small Offline Datasets via Meta Learning with Synthetic Tasks

Azza Fadhel **The Hung Tran** **Trong Nghia Hoang** **Jana Doppa**
School of EECS, Washington State University, Pullman, WA, USA

Abstract

We consider the problem of offline black-box optimization, where the goal is to discover optimal designs (e.g., molecules or materials) from past experimental data. A key challenge in this setting is data scarcity: in many scientific applications, only small or poor-quality datasets are available, which severely limits the effectiveness of existing algorithms. Prior work has theoretically and empirically shown that performance of offline optimization algorithms depends on how well the surrogate model captures the optimization bias (i.e., ability to rank input designs correctly), which is challenging to accomplish with limited experimental data. This paper proposes *Surrogate Learning with Optimization Bias via Synthetic Task Generation* (OptBias), a meta-learning framework that directly tackles data scarcity. OptBias learns a reusable optimization bias by training on synthetic tasks generated from a Gaussian process, and then fine-tunes the surrogate model on the small data for the target task. Across diverse continuous and discrete offline optimization benchmarks, OptBias consistently outperforms state-of-the-art baselines in small data regimes. These results highlight OptBias as a robust and practical solution for offline optimization in realistic small data settings.

1 INTRODUCTION

Many science and engineering applications involve optimizing complex design spaces (e.g., materials, hardware, molecules) using data from expensive lab experiments or computational simulations. Some examples include discovering new materials [Butler et al., 2018, Deshwal et al., 2021, Gantzler et al., 2023, Fadhel et al., 2026a], designing protein and drug molecules [Vamathevan et al., 2019, Dara et al., 2021], designing hardware [Deshwal et al., 2022], and optimizing parameters of additive manufacturing processes [Dara et al., 2021, Fadhel et al., 2026b]. To avoid the prohibitively high cost of direct experimentation, a more affordable strategy is to explore designs *in silico* by learning a surrogate model from experimental data and subsequently optimizing it. Existing research in this area follows two directions: (1) online optimization, where new experiments are iteratively conducted to refine the surrogate model Snoek et al. [2012a, 2015]; and (2) offline optimization, where the surrogate model is learned using only past experimental data [Trabucco et al., 2021a, 2022a].

Among these approaches, offline optimization is an emerging paradigm that has become increasingly preferable in situations where the high overhead and limited throughput of physical experimentation often makes online optimization impractical [Trabucco et al., 2021a]. The key challenge in offline optimization lies in the surrogate model's unreliable predictions for out-of-distribution (OOD) inputs which are far from the offline data regime. As a result, iterative extrapolation of the learned surrogate during the search process (e.g., gradient-based search) suffers from error propagation as the search moves away from the offline dataset, leading to suboptimal designs.

Existing research in offline optimization has largely focused on addressing this challenge using various conservative learning and optimization strategies which discourage the surrogate model from overestimating the output at unseen inputs from OOD regimes [Trabucco et al., 2021b]. However, most of this work is empirical and often lack a principled foundation. The more recent works of Hoang et al. [2025], Tan et al. [2024] have also pointed out that prior methods have inadvertently encoded an unnecessary value-

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

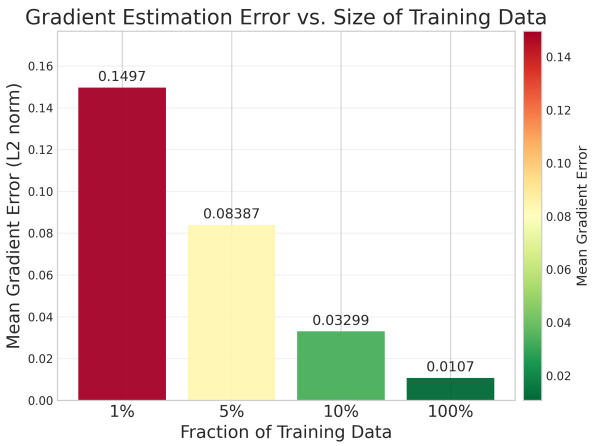

Figure 1: Conceptual illustration of gradient estimation error vs. fraction of training data on the Shekel-4D synthetic task. The surrogate is trained on 8,000 samples and evaluated on 2,000 test points; bars are colored red→green for higher→lower error. The gradient estimation error increases significantly in the small training data regime relative to the full (100 %) data.

matching bias in their surrogate modeling despite the simple intuition that shifting a function by a constant does not alter its optima. This observation suggests a key insight: rather than emphasizing value matching, optimization should emphasize gradient matching, which is more robust. The work of Hoang et al. [2025] has in particular established that the optimization performance is bounded by the gradient gap between the surrogate and the unknown oracle black-box functions. This reveals and emphasizes on an important optimization bias which prefers surrogates that match the oracle's gradient field better when gradient based search is used for optimization. Hoang et al. [2025] also introduced MATCH-OPT which is a gradient matching algorithm: by observing empirical differences between offline data points, interpolates the oracle's gradient field. This method is effective in the large offline data regime but struggles when the offline experimental data is limited, a common bottleneck in many real-world applications (e.g., drug design and materials discovery). For a conceptual illustration, we perform evaluation on the 4D Shekel synthetic function and observe that the gradient estimation error increases significantly as the the number of offline training examples reduce. This raises the main research question of this paper:

How to learn a surrogate to effectively capture optimization bias when offline data are limited?

To answer this question, we develop a new framework referred to as *Learning Optimization Bias via Synthetic Task Generation (*OPTBIAS*)*. A key novelty of this work is the idea that offline optimizers can benefit from synthetic data generation, enabling training over a population of related functions rather than relying solely on the limited data available from the target task. This perspective introduces a new paradigm for offline optimization where augmented experience from related functions helps the optimizer navigate challenging OOD regions of the search space. Matching the surrogate's gradient field to those of multiple related functions reinforces the recurring, shared structures while filtering out function-specific nuances. If the oracle function is indeed similar to this population, these shared structures are likely to align with its own, thus implicitly aligning the gradient field of the surrogate with that of the oracle. To substantiate this intuition, we need to address two technical challenges: **(1)** How to generate auxiliary task functions which are similar to the oracle and synthesize strong training signals for gradient matching from them; and **(2)** How to learn the optimization bias effectively from the synthetic training data?

Contributions. We address the above two challenges via the following technical contributions:

1. We develop a robust procedure SIM4OPT to generate synthetic, closed-form functions which are similar to the oracle via fitting Gaussian process priors [Rasmussen and Williams, 2006] on offline data and simulating input sequences with increasing output as learning feedback for gradient matching (Section 3.1).

2. We develop a generalization of MATCH-OPT [Hoang et al., 2025], named OPTBIAS, which leverages meta learning techniques to effectively incorporate such simulated gradient-matching feedback to synthesize a good approximation of the (unobserved) oracle's gradient field (Section 3.2).

3. We conduct extensive experiments on real-world design optimization benchmarks to demonstrate the effectiveness of OPTBIAS. Our results show that it produces significantly better designs than baseline methods with small datasets. Our ablation experiments demonstrate that meta learning is a better approach than pretraining to capture optimization bias; and OPTBIAS performs best when the simulated data are well-aligned with the optimization bias (Section 4).

2 BACKGROUND

Let $\mathcal{X}$ denote the input design space and $g : \mathcal{X} \mapsto \Re$ be an unknown, expensive real-valued objective function that evaluates output $z = g(\mathbf{x})$ for any input $\mathbf{x} \in \mathcal{X}$. The goal is to find an optimal design $\mathbf{x}_* = \arg\max_{\mathbf{x}} g(\mathbf{x})$. Since the gradient of $g(\mathbf{x})$ is typically not accessible, previous approaches use derivative-free methods, such as random gradient es-

timation [Wang et al., 2018], or Bayesian optimization [Snoek et al., 2015], which requires online active sampling of experimental evaluations to either approximate the target function's derivative or learn its surrogate model. However, online data collection can be expensive or impractical. For example, lab-tests of candidate drugs are time-consuming and preferably done in high throughput setting, whereas test-driving an automatic controller for self-driving vehicles in real environments can be hazardous. Offline optimization has recently been proposed to avoid online data collection via fitting a parameterized surrogate model $g_\phi(\mathbf{x})$ for $g(\mathbf{x})$ on an existing dataset of past input designs and their evaluations, i.e., $\mathcal{D} = \{(\mathbf{x}_1, z_1), (\mathbf{x}_2, z_2), \cdots, (\mathbf{x}_n, z_T)\}$. This is achieved via solving the following:

$$\phi \triangleq \arg\min_{\phi'} \sum_{i=1}^{n} \ell\Big(g_{\phi'}(\mathbf{x}_i),\ z_i\Big)\ , \qquad (1)$$

where ϕ denotes the parameters of surrogate model, e.g., $g_\phi(\mathbf{x}) = \phi^\top \mathbf{x}$, and ℓ denotes a supervised learning loss function, e.g., $\ell(z, z') = \|z - z'\|^2$. Once learned, g_ϕ is fixed and used as a surrogate to find the optimized designs via gradient search. This implicitly assumes that the target function is continuous. For design tasks on discrete input spaces, a continuous embedding of the data is first learned and offline optimization is performed on the embedding space. The focus of this paper is on real-world settings (e.g., drug design and materials discovery) where the size of offline experimental data is small (T is a small number).

Since we employ Gaussian processes (GPs) to generate synthetic optimization tasks, we provide some background on GPs to make the paper self-contained.

Gaussian Processes. A Gaussian process (GP) [Rasmussen and Williams, 2006] defines a prior distribution over a random function $g(\mathbf{x})$, characterized by a mean function $m(\mathbf{x})$ and a kernel function $k(\mathbf{x}, \mathbf{x}')$. Under this prior, let $\mathbf{z} = \{z_1, z_2, \ldots, z_n\}$ denote the set of noisy output $z_i = g(\mathbf{x}_i) + \epsilon$ with $\epsilon \sim \mathbb{N}(0, \sigma^2)$ at any finite set of inputs $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$. We have $\mathbf{x}$ follows a multivariate Gaussian distribution $\mathbf{z} \sim \mathbb{N}(\mathbf{m}, \mathbf{K})$ where $\mathbf{m} = [m(\mathbf{x}_1), m(\mathbf{x}_2), \ldots m(\mathbf{x}_n)]$ denotes the mean vector, $\mathbf{K}$ denotes the covariance matrix with entries $\mathbf{K}^{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Given training data $D = (\boldsymbol{X}, \boldsymbol{z})$, the GP prediction z_* and unseen test input $\mathbf{x}_*$ is also distributed by a Gaussian, $z_* \sim \mathbb{N}(\mu_D(\mathbf{x}_*), \sigma_D^2(\mathbf{x}_*))$ with:

$$\mu_D(\mathbf{x}_*) = m(\mathbf{x}_*) + \mathbf{k}_*^\top \big(\mathbf{K} + \sigma^2\mathbf{I}\big)^{-1}\big(\mathbf{z} - \mathbf{m}\big) \qquad (2)$$

$$\sigma_D^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top \big(\mathbf{K} + \sigma^2\mathbf{I}\big)^{-1}\mathbf{k}_* \qquad (3)$$

where $\mathbf{k}_* = [k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_1)]$ denote the covariance vector between the test and training inputs.

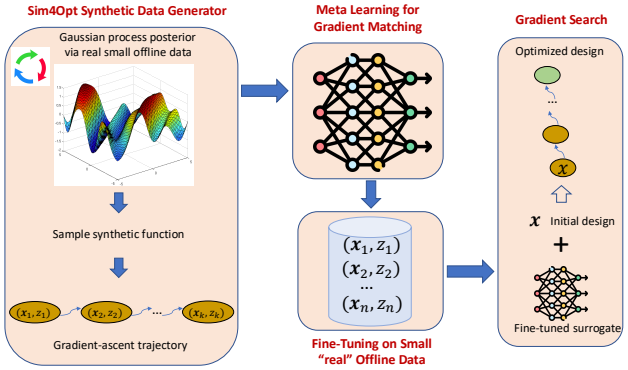

Figure 2: Workflow overview of OptBias. First, we use a novel Sim4Opt procedure to generate synthetic functions which are similar to the oracle function. Then, we combine meta-learning with gradient matching (MatchOpt) to distill the common parts across gradient fields of the synthetic functions into a surrogate, thus implicitly aligning it with the (unobserved) gradient field of the oracle function.

3 OPTBIAS FRAMEWORK

In this section, we first provide the intuition and overview of the proposed OptBias framework. Next, we explain our Sim4Opt approach to generate simulated training data using a Gaussian process to address the small data challenge. Finally, we describe our meta learning approach to effectively learn optimization bias from simulated data using the gradient-matching procedure of [Hoang et al., 2025].

Key Intuition and Overview of OptBias. Existing offline optimization methods often aim to learn surrogate models via value matching followed by a gradient based search for optimization. However, this is unnecessary for the purpose of optimization. For example, the maximizer of a function $g(\mathbf{x})$ is identical to that of $g(\mathbf{x}) + c$ for any constant c, which implies that preserving absolute function values is not required to recover the optimizer. Instead, it is more important that the surrogate preserves the gradient field of the oracle which leads to its optima. This motivates the use of gradient matching as a more appropriate optimization bias [Hoang et al., 2025, Tan et al., 2024] which focuses on learning the directional signals essential for optimization and avoids the unnecessary bias of matching absolute values.

However, learning to match gradient is bottlenecked by the lack of training data, which is often the case in real-world scenarios such as materials discovery and drug design. To alleviate this bottleneck, we propose to enrich the offline data via synthetic data generation. In particular, we aim to acquire additional learning feedback for gradient matching by generating and

simulating gradient-ascent trajectories from a collection of synthetic functions similar to the oracle (Section 3.1). Our intuition is that by constructing synthetic functions similar to the oracle, their gradient fields will naturally overlap with the oracle's.

Matching the surrogate's gradient field to this collection of functions amounts to learning from the common structures that consistently recur across functions while down-weighting nuances that are specific to individual functions (Section 3.2). In this way, the surrogate can recover shared patterns that are otherwise difficult to discern from limited offline data alone. Next, we fine-tune the learned meta-surrogate using the small (real) offline data for the target optimization task. Finally, we perform gradient search on the trained surrogate from high value inputs in the real offline data to find optimized designs. Notably, unlike prior work that avoids the oracle's out-of-distribution regime, OptBias introduces a new paradigm where synthetic data is used to navigate it more reliably. Figure 2 illustrates key steps of OptBias and Algorithm 2 gives the pseudo-code.

3.1 Sim4Opt: Simulated Data Generation

One possible approach is to construct a variety of Gaussian process (GP) priors [Rasmussen and Williams, 2006] over candidate functions and sample from them. This idea was previously developed in ExPT [Nguyen et al., 2023], which samples GP priors by uniformly drawing from pre-specified parameter ranges. Each sampled GP prior defines a distribution over functions, from which synthetic functions can be drawn. Data sampled from these functions are then used to supplement the limited offline dataset for surrogate training. However, this approach suffers from the following key limitations:

1. The GP priors used in ExPT are sampled from a pre-selected ranges rather than learned from offline data. As such, there is a risk that some of which might be too distant from the target oracle function. These can lead to synthetic functions whose gradient fields poorly align with the oracle's. This will in turn introduce spurious gradient structures that obscure the recurring gradient patterns across functions similar to the oracle that we aim to capture.

2. ExPT generates synthetic data only by evaluating each sampled posterior mean function at unlabeled inputs drawn from the offline dataset. Consequently, the resulting synthetic data might not correspond to any gradient-ascent trajectories that highlight the most informative regions of the gradient field. Furthermore, the Design-Bench testbed [Trabucco et al., 2022a] samples these inputs predominantly from low objective value regions of the oracle. As those inputs lie in the oracle's low objective value regime, they will likewise fall into the low objective value regions of the synthetic functions because the synthetic functions are constructed to resemble the oracle around the offline inputs. As a result, the synthetic data remain confined to low objective value design spaces. This limits the model's ability to capture the common gradient structures across synthetic functions that point toward high objective value designs.

To address these issues, we propose a new data generation method that can generate auxiliary functions which are more similar to the oracle function. Furthermore, we also develop sampling techniques that draw highly informative gradient-ascent trajectories from these auxiliary functions. The full data generation procedure is detailed in Algorithm 1.

Addressing Limitation 1. The key idea is to adapt the sampled GP prior in ExPT with the offline data via GP posterior computation to ensure that the resulting posterior has strong support over functions similar to the oracle. The corresponding GP posterior's mean functions then have closed-form expressions which can be used to run M-step gradient descent and ascent from the offline input points. This results in input trajectories that traverse between low objective value regions and high objective value regions, thus addressing the limitation. This implicitly assumes that the oracle is a smooth function, or that at least its high-objective-value regime can be approximated well by a smooth function. Under this assumption, GP posterior mean functions capture similar geometric structures that connect low-value and high-value regions of the objective landscape.

Addressing Limitation 2. Such trajectories can be used to learn the common increasing objective value gradient structure across the sampled functions via customized meta learning approach (see Section 3.2). Importantly, the GP is not used as a high-fidelity surrogate of the oracle. Instead, it serves only to generate a family of synthetic functions that behave similarly to the oracle within the offline data regime. Training the optimizer across this population of functions allows the meta-learner to capture optimization bias that generalizes to the target oracle despite the limited offline data. The central assumption is that if an optimizer performs well on average across a population of functions that are diverse but sufficiently similar to the oracle in the offline data regime, it will also perform well for the oracle itself. As these functions are similar to the oracle function, the common low objective value to high objective value gradient structure will likely align with the oracle's. To better balance between exploration and exploitation in the simulation runs towards high-value regions, we can also replace the posterior

Algorithm 1 SIM4OPT: Synthetic Data Generator

Input: Offline dataset $D_o = \{\boldsymbol{X}, \boldsymbol{z}\}$; evolution operator $\mathcal{T}$; number of evolution steps M; number of simulated functions n
Output: Mapping $\mathcal{S} = \{(i, D_s^{(i)})\}_{i=1}^n$, where each $D_s^{(i)} = \{(\boldsymbol{X}^{s,(i)}, \boldsymbol{z}^{s,(i)})\}$ is *ordered order by* $\boldsymbol{z}^{s,(i)}$
$\mathcal{S} \leftarrow \emptyset$
for $i = 1$ **to** n **do**
  $D_s^{(i)} \leftarrow \emptyset$ // synthetic dataset for function i
  Sample kernel parameters for function i
  $\phi_s^{(i)} = (\ell_s^{(i)}, (\sigma_s^2)^{(i)}) \sim U((\ell_0, \sigma_0^2) - \delta, (\ell_0, \sigma_0^2) + \delta)$
  Compute GP posterior mean $\bar{g}_{\phi_s^{(i)}}$ **via** Eq. (3)
  Set $\boldsymbol{X}_+^{0,(i)} \leftarrow \boldsymbol{X}$, $\boldsymbol{X}_-^{0,(i)} \leftarrow \boldsymbol{X}$
  for $m = 1$ **to** M **do**
    Explore low objective value regions:
    $\boldsymbol{X}_-^{m,(i)} \leftarrow \mathcal{T}\Big(-\bar{g}_{\phi_s^{(i)}}, \boldsymbol{X}_-^{m-1,(i)}\Big)$
    Explore high objective value regions:
    $\boldsymbol{X}_+^{m,(i)} \leftarrow \mathcal{T}\Big(\bar{g}_{\phi_s^{(i)}}, \boldsymbol{X}_+^{m-1,(i)}\Big)$
    Compute pseudo-labels
    $\boldsymbol{z}_-^{m,(i)} \leftarrow \bar{g}_{\phi_s^{(i)}}(\boldsymbol{X}_-^{m,(i)})$
    $\boldsymbol{z}_+^{m,(i)} \leftarrow \bar{g}_{\phi_s^{(i)}}(\boldsymbol{X}_+^{m,(i)})$
    Update simulated data:
    $D_s^{(i)} \leftarrow D_s^{(i)} \cup \{\boldsymbol{X}_-^{m,(i)}, \boldsymbol{z}_-^{m,(i)}\}$
    $D_s^{(i)} \leftarrow D_s^{(i)} \cup \{\boldsymbol{X}_+^{m,(i)}, \boldsymbol{z}_+^{m,(i)}\}$
  end for
  Monotone ordering (per function):
  $D_s^{(i)} \leftarrow \mathrm{SortByY}\Big(D_s^{(i)}\Big)$ *(ascending in* $\boldsymbol{z}^{s,(i)}$*)*
  Accumulate mapping: $\mathcal{S} \leftarrow \mathcal{S} \cup \{(i, D_s^{(i)})\}$
end for
Return $\mathcal{S}$

mean with its upper-confidence bound (UCB) which was previously designed as an acquisition function in the context of online black-box optimization.

Indeed, we perform ablation experiments within OPTBIAS by varying the synthetic data generator (our proposed SIM4OPT and the data generator in ExPT) to demonstrate the effectiveness of SIM4OPT procedure.

3.2 Meta Learning using Simulated Data

To remove the redundant value-matching bias, the gradient matching approach in [Hoang et al., 2025] aims to learn a parameterized surrogate $g_\phi(\boldsymbol{x})$ that matches the oracle $g(\boldsymbol{x})$'s gradient field via $\phi^* = \arg\min_\phi \ell(\phi)$ with the following gradient matching loss:

$$\ell(\phi) \;=\; \mathbb{E}\left(\Delta z - \Delta\mathbf{x}^\top \int_0^1 \nabla g_\phi\Big(\mathbf{x} \;+\; t\Delta\mathbf{x}\Big)\mathrm{d}t\right)^2. \quad (4)$$

where $\Delta\mathbf{x} = \mathbf{x}' - \mathbf{x}$ and $\Delta z = g(\mathbf{x}') - g(\mathbf{x})$. Eq. (4) can be empirically estimated via sampling $(\mathbf{x}, \mathbf{x}')$ and their corresponding output values $(g(\mathbf{x}), g(\mathbf{x}'))$ from

Algorithm 2 OPTBIAS Algorithm

Input: Offline data D_o; simulated sets $\mathcal{S} = \{(i, D_s^{(i)})\}_{i=1}^n$ (Alg. 1); number of training epoch E; number of batches per epoch n, stepsize α, η, γ;
Output: Best designs x^*
Initialize meta-learner hyperparameters ϕ
Training meta-learner on simuated sets
for epoch 1 to E **do**
  Sample a batch of n functions $\{i, D_s^{(i)}\}_{i=1}^n \sim S$
  for i in $1 \dots n$ **do**
    Compute $\ell_i(\phi)$ via Eq. (6)
    Compute $\phi_i' = \phi - \alpha\nabla_\phi \ell_i(\phi)$
  end for
  Update $\phi \leftarrow \phi - \eta\nabla_\phi \sum_{i=1}^n \ell_i(\phi_i')$
end for
Fine-tuning on small offline data D_o
Compute $\ell(\phi)$ via Eq. (4)
Update $\phi \leftarrow \phi - \alpha\nabla_\phi \ell(\phi)$
Gradient search on surrogate $\nabla_x g_\phi(x)$
for t in $1 \dots M$ **do**
  $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \gamma\nabla_{\mathbf{x}} g_\phi(\mathbf{x}_{t-1})$
end for
return $\mathbf{x}^* = \mathbf{x}_M$

the offline dataset. However, this is inaccurate with limited data. To overcome this challenge, we will leverage the sampled gradient-ascent trajectories from our synthetic data generation procedure SIM4OPT,

$$\zeta^{(i)} \;=\; \Big(\Big(\mathbf{x}_1^{(i)}, z_1^{(i)}\Big), \Big(\mathbf{x}_2^{(i)}, z_2^{(i)}\Big), \dots, \Big(\mathbf{x}_\kappa^{(i)}, z_\kappa^{(i)}\Big)\Big), \quad (5)$$

with $\kappa = 2M + 1$ extracted from the synthetic functions obtained via Algorithm 1. In particular, let $\ell_i(\phi)$ denotes the gradient matching loss with respect to $\zeta^{(i)}$ of the i-th GP posterior mean function,

$$\ell_i(\phi) \;=\; \mathbb{E}_r\left(\Delta z_r - \Delta\mathbf{x}_r^\top \int_0^1 \nabla g_\phi\Big(\mathbf{x}_r + t\Delta\mathbf{x}_r\Big)\mathrm{d}t\right)^2$$
$$\text{with } \Delta\mathbf{x}_r \;=\; \mathbf{x}_{r+1}^{(i)} - \mathbf{x}_r^{(i)}\;, \Delta z_r \;=\; z_{r+1}^{(i)} - z_r^{(i)}\;. \quad (6)$$

The expectation in Eq. (6) is over uniform choice of $r \in [\kappa - 1]$. We can then learn a surrogate g_ϕ that matches with the common gradient-ascent structure across the sampled functions via minimizing the following meta-learning loss [Finn et al., 2017a, Antoniou et al., 2018],

$$\phi^* \quad = \quad \arg\min_\phi \mathbb{E}_i\Big[\ell_i\Big(\phi - \alpha\nabla_\phi \ell_i(\phi)\Big)\Big]\;, \quad (7)$$

where $\alpha > 0$ and $i \sim \mathbb{U}(1, n)$ assuming we have drawn n synthetic functions. Intuitively, this optimizes for a surrogate model whose gradient field can be fast-adapted towards any downstream target with limited data [Finn et al., 2017a]. We will then fine-tune $\phi = \phi^*$

with the small (real) offline data for the target optimization task via Eq. (4) to align with the oracle. This intuitively emphasizes learning the common gradient fields across different functions along such gradient-ascent trajectories. The solution of Eq. (6) can be intuitively viewed as a vantage point on the space of candidate gradient flows which can be fine-tuned quickly towards any downstream tasks. Furthermore, as the gradient loss $\ell_i(\phi)$ is constructed along the sampled gradient-ascent trajectories $\zeta^{(i)}$, it naturally emphasizes fitting the gradient field along such low objective value to high objective value trajectories.

4 EXPERIMENTS AND RESULTS

This section presents the experiments and results to evaluate the performance of our proposed method OptBias in comparison to the baselines.

4.1 Benchmarks

We conduct empirical evaluations on six benchmark tasks spanning robotics, genomics, and molecular biology. Each task is formulated as a black-box optimization problem, where the objective function is fixed and unknown to the optimizer. The setting follows the offline experimental design protocol: algorithms are provided only with a static dataset of design-response pairs for training and must recommend a batch of final designs for evaluation under the true objective. The black-box function is used solely at test time to assess the quality of these recommended designs.

The benchmark suite includes two continuous and one discrete tasks from the `Design-Bench` framework [Trabucco et al., 2022b], as well as three additional discrete RNA inverse folding tasks based on ViennaRNA [Lorenz et al., 2011] and the Bootgen benchmark [Kim et al., 2023]. A brief overview of each task is provided below:

Ant Morphology (ANT) [Brockman et al., 2016]: A continuous task that involves optimizing the morphology of a quadruped robot to maximize forward locomotion speed. The input space is 60-dimensional and parameterizes the robot's body shape.

D'Kitty Morphology (DKITTY) [Ahn et al., 2020]: A similar morphology optimization task for a different robot (D'Kitty), with a 56-dimensional continuous design space. The objective is to generate designs that improve locomotion or directional control.

TF-Bind-8 (TF8) [Barrera et al., 2016]: This task focuses on optimizing DNA sequences of length 8 to maximize binding affinity to human transcription factors. The design space is discrete, consisting of strings over the nucleotide alphabet {A, C, G, T}.

RNA-Binding Tasks (RNA-A, RNA-B, RNA-C) [Lorenz et al., 2011]: These inverse folding tasks aim to design RNA sequences (length 14) that fold into predefined target secondary structures. The goal is to maximize the predicted binding affinity by minimizing the free energy of the resulting structure.

These benchmarks collectively cover a wide range of structural properties (continuous vs. discrete design space, high vs. low dimensionality, providing a rigorous benchmark for evaluation.

4.2 Baselines

Baselines. We compare against several recent baselines in offline optimization: (1) **GA** which fits a surrogate using offline data and uses gradient ascend to find optimal input designs that maximize the surrogate output [Trabucco et al., 2022b]; (2) **MINs** which learns an *inverse model* from scores to inputs (model inversion), so high-score conditions generate candidate designs directly, using only offline $(\mathbf{x}, z)$ pairs [Kumar and Levine, 2020]; (3) **COMs** which learns a conservative (lower-bounding) surrogate to curb over-optimism on OOD inputs [Trabucco et al., 2021b]; (4) **DEMO** which refines surrogate proposals with a diffusion prior to pull candidates toward the data distribution [Yuan et al., 2024]; (5) **LTR (learning-to-rank)** which trains listwise/pairwise ranking losses to order candidates by quality rather than regress objective values [Tan et al., 2024]; (6) **Match-Opt** which learns a differentiable surrogate whose *gradient field* is trained to match improvement signals in the offline data [Hoang et al., 2025]; (7) **Batch BO (q-EI)** which adopts batch Bayesian optimization using batch expected improvement to recommend multiple designs per round; (8) **REINFORCE** which learns a variance-reduced score-function gradient estimator using importance sampling on offline data [Williams, 1992]; and (9) **ExPT** which pre-trains a transformer on synthetic tasks and fine-tunes it with offline data of the target task [Nguyen et al., 2023].

4.3 Evaluation Methodology

To evaluate the effectiveness of OptBias in scenarios with limited offline data, we consider a challenging few-shot setting constructed from each benchmark task. Specifically, we form the few-shot dataset $\mathcal{D}_{\text{fs}}$ by selecting 1% of lowest value data points (aka poorest) from the offline dataset to simulate a realistic scenario with limited experimental data similar to the setup in ExPT [Nguyen et al., 2023]. Following the approach in [Trabucco et al., 2022a], each method generates 128 optimized design candidates, which are then evaluated by the oracle function. The candidates' performance

Table 1: Performance achieved by baselines under the limited data settings (using 1% of the offline dataset) across various benchmark tasks.

	Benchmarks						
Method	**Ant**	**D'Kitty**	**TFBind8**	**RNA1**	**RNA2**	**RNA3**	**Mean Rank**
D_{best} from 1% offline data	0.123	0.307	0.124	0.028	0.027	0.066	None
GA	0.734 ± 0.054	0.831 ± 0.034	0.782 ± 0.117	0.475 ± 0.134	0.497 ± 0.091	0.317 ± 0.072	5.17
MINs	0.767 ± 0.084	0.895 ± 0.011	0.762 ± 0.135	0.076 ± 0.062	0.019 ± 0.029	0.132 ± 0.059	7.67
COMs	0.908 ± 0.042	0.705 ± 0.012	0.438 ± 0.000	0.279 ± 0.059	0.278 ± 0.120	0.293 ± 0.044	6.08
DEMO	0.800 ± 0.150	0.845 ± 0.037	0.677 ± 0.168	0.166 ± 0.034	0.141 ± 0.045	0.295 ± 0.043	6.50
LTR	0.865 ± 0.118	0.91 ± 0.026	0.45 ± 0.027	0.19 ± 0.189	0.215 ± 0.262	0.417 ± 0.176	5.17
Match-Opt	0.859 ± 0.007	0.912 ± 0.013	0.438 ± 0.002	0.129 ± 0.000	0.119 ± 0.000	0.166 ± 0.000	6.92
Batch BO (q-EI)	0.482 ± 0.066	0.816 ± 0.030	0.928 ± 0.026	0.490 ± 0.114	0.523 ± 0.103	0.462 ± 0.097	4.00
REINFORCE	0.327 ± 0.062	0.588 ± 0.162	0.872 ± 0.057	0.071 ± 0.054	0.062 ± 0.054	0.119 ± 0.064	8.83
ExPT	0.9276 ± 0.009	0.955 ± 0.009	0.879 ± 0.080	0.249 ± 0.013	0.239 ± 0.009	0.341 ± 0.041	3.33
OptBias (ours)	0.960 ± 0.017	0.947 ± 0.012	0.945 ± 0.024	0.503 ± 0.025	0.540 ± 0.113	0.450 ± 0.134	**1.33**

Table 2: Performance comparison between our proposed method OptBias, which uses meta-learning for surrogate training with synthetic data, and its variant, which replaces meta-learning with a pre-training procedure.

	Benchmarks					
Method	**Ant**	**D'Kitty**	**TFBind8**	**RNA1**	**RNA2**	**RNA3**
OptBias (pre-training)	0.884 ± 0.043	0.932 ± 0.007	0.438 ± 0.000	0.129 ± 0.000	0.119 ± 0.000	0.166 ± 0.000
OptBias (meta-learning)	0.960 ± 0.017	0.947 ± 0.012	0.945 ± 0.024	0.503 ± 0.025	0.540 ± 0.113	0.450 ± 0.134

are ranked and reported at 100th percentile. All results are averaged over four independent runs. Specifically, each optimization method starts with 128 initial candidates The 128 final candidate recommendations are then determined via updating the initial candidates following the corresponding optimization policy. In low-data regime, we score every unlabeled design from the offline data with our surrogate model, select the top 256 by predicted quality, and then randomly choose 128 of these as initial candidates.

4.4 Results and Discussion

This section evaluates **OptBias** across a diverse set of offline optimization tasks, including locomotion control (`ant`, `dkitty`), sequence design from the RNA inverse folding benchmark (`rna1`, `rna2`, `rna3`), and protein–DNA binding affinity prediction (`tfb-8`). Our experiments aim to answer the following questions:

Q1. *How does* OptBias *compare to existing SOTA baselines in settings with limited data?*

Q2. *What is the impact of using meta-learning to train a surrogate on synthetic data?*

Q3. *What is the impact of generating data with* Sim4Opt *on optimization performance?*

Q4. *How sensitive is* OptBias *to the design choices of the synthetic task generation procedure in* Sim4Opt*?*

Q5. *How well do synthetic designs generated by* Sim4Opt *capture high-value regions of the objective landscape?*

We report the results and analyze the observations regarding each of the above questions below.

Addressing Q1. Under limited-data settings (i.e., using 1% of the original offline dataset), OptBias attains the best performance on 4/6 benchmark tasks. OptBias also achieves second-best performance on the remaining tasks: *D'Kitty* (0.947 vs. ExPT's 0.955) and *RNA3* (0.450 vs. BO-qEI's 0.462). The performance gap in these tasks is however marginal. Overall, these results demonstrate that incorporating synthetic data helps improve performance significantly over SOTA offline optimization methods in limited data settings. The significant improvement of OptBias over MatchOpt also demonstrates that incorporating synthetic data is essential to improve gradient matching (to capture optimization bias).

Addressing Q2. To ablate the impact of using meta-learning in surrogate training, we compare the optimization performance achieved by OptBias and a variant that replaces meta-learning [Finn et al., 2017b] with a pre-training procedure that runs MatchOpt for gradient matching on data sampled from all synthetic tasks. The key difference here is that meta-learning optimizes for a surrogate that can be fast-adapted to match the gradient field of any synthetic functions with limited data whereas the pre-training procedure attempts to match all gradient fields simultaneously. Our hypothesis is that meta-learning is more effective than pre-training when using a low-

Table 3: Performance comparison between our proposed method OptBias and its variant which replaces Sim4Opt with ExPT's data generation procedure.

	Benchmarks					
Method	**Ant**	**D'Kitty**	**TFBind8**	**RNA1**	**RNA2**	**RNA3**
OptBias w/ Random	0.701 ± 0.157	0.857 ± 0.058	0.882 ± 0.0548	0.493 ± 0.0458	0.532 ± 0.0549	0.518 ± 0.077
OptBias (ours)	0.960 ± 0.017	0.947 ± 0.012	0.945 ± 0.024	0.503 ± 0.025	0.540 ± 0.113	0.450 ± 0.134

complexity model, which is often preferable in offline optimization since the available data is often task-specific and moderate in size, thus cannot support pre-training a high-complexity model. Intuitively, we can think of meta-learning as a mechanism that recognizes the most common parts across the gradient fields of different synthetic functions and filters out irrelevant parts, which makes fine-tuning easier on downstream target tasks. Conversely, pre-training (regardless of model size) tends to overfit in settings with moderate and less diverse data. This insight is supported by our empirical results in Table 2 which shows that OptBias consistently outperforms its variant that replaces meta-learning with pre-training.

Addressing Q3. To ablate the impact of generating data with Sim4Opt, we also compare OptBias with a variant which replaces Sim4Opt with the ExPT's data generation produre for pre-training [Nguyen et al., 2023]. The key difference (as previously discussed in Section 3.1) is that unlike Sim4Opt, ExPT generates synthetic data only within the offline input regime rather than along potential gradient-ascent trajectories that pass through the offline input regime. Our hypothesis here is that as offline optimization is essentially a search from low objective value to high objective value regimes with the input space, observing explicit examples of trajectories with increasing objective function values that extend beyond the offline input regime is more informative than observing randomly within it. This will in turn help improve fine-tuning in limited data setting (i.e., 1% data), resulting in better optimization performance overall as reported in Table 3. It can be observed that OptBias (which uses Sim4Opt for data generation) is significantly better than its variant (that uses ExPT's data generation procedure) on most tasks and on average.

Addressing Q4. To evaluate the robustness of Sim4Opt to design choices such as GP kernel selection, trajectory length, and the use of UCB vs. posterior mean, we conducted an ablation study across several commonly used configurations. The results summarized in Table 4 show the final optimization performance from the OptBias approach over four seeds for each setting. Across ANT and DKitty, the variation in optimization performance across configurations is small relative to the overall gains provided by Sim4Opt. These results demonstrate that OptBias is stable fora wide range of reasonable hyperparameter choices for the synthetic task generation procedure.

Addressing Q5. To evaluate the effectiveness of synthetic designs generated by multiple GPs, we first collect high pseudo-value candidates produced by these models. We then evaluate these candidates using the oracle function and analyze the resulting distribution of ground-truth objective values. As shown in Fig. 3, the synthetic designs exhibit a broad and informative distribution that sufficiently covers high-oracle-value regions. This coverage provides richer supervision signals, which are critical for addressing the out-of-distribution (OOD) challenge when performing gradient matching. In contrast, standard offline datasets are typically biased toward low-value regimes and lack sufficient high-quality samples. These observations strongly motivate our meta-training procedure on synthetic data, which enhances the model's ability to generalize beyond the support of the original dataset and effectively tackles the OOD problem.

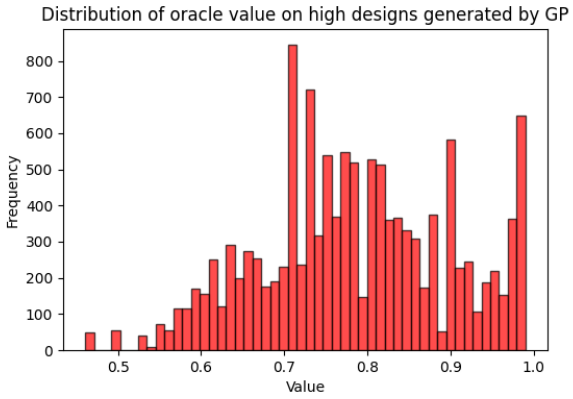

Figure 3: Empirical distribution of oracle-evaluated y-values (objective function values) over synthetic designs generated by multiple Gaussian Processes.

5 RELATED WORK

Online Black-box Optimization. Many classical approaches to experimental design for black-box optimization problems operate in an online manner: sequentially select a sequence of one or more experiments based on the outcomes of past experiments for optimization. Bayesian Optimization (BO) is the most

Table 4: Ablation study on GP design choices used in the Sim4Opt synthetic task generation procedure. Results show final optimization performance of OptBias averaged over four seeds.

Setting	ANT (mean ± std)	DKitty (mean ± std)
Matern Kernel	0.957 ± 0.004	0.898 ± 0.022
UCB Acquisition	0.937 ± 0.023	0.948 ± 0.010
Trajectory Length = 70	0.942 ± 0.016	0.947 ± 0.009
Lengthscale = 1.5	0.950 ± 0.008	0.871 ± 0.031
Lengthscale = 2.0	0.916 ± 0.063	0.956 ± 0.007
OptBias (main results)	0.960 ± 0.010	0.947 ± 0.012

prominent framework [Garnett, 2023, Snoek et al., 2012b, Swersky et al., 2013] for sample-efficient black-box function optimization. BO relies on two key elements. First, a surrogate model such as Gaussian Processes [Srinivas et al., 2009, Deshwal and Doppa, 2021, Deshwal et al., 2023], Neural Processes [Garnelo et al., 2018a,b, Gordon et al., 2019, Kim et al., 2019, Nguyen and Grover, 2022, Singh et al., 2019], and Bayesian Neural Networks [Goan and Fookes, 2020] to make predictions with quantified uncertainty for unknown inputs. Second, a goal-driven acquisition function to to select one or more experiments. Some popular examples include expected improvement, upper confidence bound, and those based on information theory [Garnett, 2023, Ament et al., 2023, Hernández-Lobato et al., 2014, Wang and Jegelka, 2017, Belakaria et al., 2019, 2020].

However, in real-world scenarios where the overhead and cost of setting up experiments are prohibitively expensive (e.g., wet lab experiments that require expensive materials and equipment), black-box optimization in the online setting is impractical. The closest BO approach to offline optimization is one-shot batch BO, e.g., q-EI [Wang et al., 2020]. Prior work in offline optimization and the experiments from this paper have demonstrated improvements over this BO baseline.

Offline Black-box Optimization. This is an emerging and more practical paradigm which assumes access to an existing database of previously collected input-output pairs (aka offline dataset) and solves this problem in an offline manner. Existing methods can be broadly classified into two categories. First, *forward approaches* construct a surrogate model that approximates the true objective, and then optimize this proxy to identify promising candidates. The central challenge lies in ensuring reliable extrapolation for out-of-distribution (OOD) inputs. The most common strategy is to penalize large surrogate predictions on OOD inputs. There are several methods that implement this strategy using principles from adversarial robustness [Trabucco et al., 2021b, Yao et al., 2024], uncertainty quantification [Trabucco et al., 2022b], and semi-supervised learning [Yuan et al., 2023, Chen et al., 2023]. Recent work has shown that learning relative rankings of inputs is a better objective to create effective surrogate models. To complement surrogate modeling, there are methods to create policies to guide gradient search towards high-performing inputs [Chemingui et al., 2024]. Second, *inverse approaches* primarily learn a conditional deep generative model that maps to potential designs, bypassing explicit objective modeling. Methods in this category differ in the choice of generative model: model inversion networks in MINs [Kumar and Levine, 2020], auto-regressive model in BONET [Krishnamoorthy et al., 2022], and diffusion model in DDOM, ROOT [Krishnamoorthy et al., 2023, Dao et al., 2025b].

However, effectiveness of all these methods critically depends on large amounts of offline data and *small data* is the norm in many real-world applications which is the focus of this paper. A very recent work referred to as ExPT [Nguyen et al., 2023] addresses the small data challenge using pre-training on synthetic data generated from a GP with random hyper-parameters but does not capture the optimization bias important for strong performance. Our OptBias framework achieves this using meta learning with data from an improved synthetic data generation procedure.

6 CONCLUSION

This paper studied the under-studied problem of black-box optimization from small offline datasets which is common in many scientific applications. It addressed the challenge of small data setting using a novel synthetic data generation procedure and meta learning to capture optimization bias within a meta surrogate which is then fine-tuned on the small real data from the target task. Our experiments and ablations demonstrate the effectiveness of both synthetic data generator and meta learning approach, resulting in improvements over state-of-the-art baselines. Exploring alternative meta-learning approaches within OptBias is an interesting direction for future work.

References

Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. Robel: Robotics benchmarks for learning with low-cost robots. In *Conference on robot learning*, pages 1300–1313. PMLR, 2020.

Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, 2023.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

Luis A Barrera, Anastasia Vedenko, Jesse V Kurland, Julia M Rogers, Stephen S Gisselbrecht, Elizabeth J Rossin, Jaie Woodard, Luca Mariani, Kian Hong Kock, Sachi Inukai, et al. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351(6280):1450–1454, 2016.

Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Syrine Belakaria, Aryan Deshwal, Nitthilan Kannappan Jayakodi, and Janardhan Rao Doppa. Uncertainty-aware search framework for multi-objective bayesian optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10044–10052, 2020.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559 (7715):547–555, 2018.

Yassine Chemingui, Aryan Deshwal, Trong Nghia Hoang, and Janardhan Rao Doppa. Offline model-based optimization via policy-guided gradient search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11230–11239, 2024. doi: 10.1609/aaai.v38i10.29001.

Can Chen, Yingxueff Zhang, Jie Fu, Xue Steve Liu, and Mark Coates. Bidirectional learning for offline infinite-width model-based optimization. *Advances in Neural Information Processing Systems*, 35:29454–29467, 2022.

Can (Sam) Chen, Christopher Beckham, Zixuan Liu, Xue Liu, and Christopher Pal. Parallel-mentoring for offline model-based optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL `https://arxiv.org/abs/2309.11592`.

Can Sam Chen, Christopher Beckham, Zixuan Liu, Xue Liu, and Christopher Pal. Robust guided diffusion for offline black-box optimization. *arXiv preprint arXiv:2410.00983*, 2024.

Manh C Dao, Phi Le Nguyen, Thao N Truong, and Trong N Hoang. Incorporating surrogate gradient norm to improve offline optimization techniques. *Advances in Neural Information Processing Systems*, 37:8014–8046, 2024.

Manh Cuong Dao, Phi Le Nguyen, Thao Nguyen Truong, and Trong Nghia Hoang. Boosting offline optimizers with surrogate sensitivity. *arXiv preprint arXiv:2503.04181*, 2025a.

Manh Cuong Dao, The Hung Tran, Phi Le Nguyen, Thao Nguyen Truong, and Trong Nghia Hoang. Root: Rethinking offline optimization as distributional translation via probabilistic bridge, 2025b. URL `https://arxiv.org/abs/2509.16300`.

Shankar Dara et al. Machine learning in drug discovery: A review. *Archives of Computational Methods in Engineering*, 2021.

Aryan Deshwal and Jana Doppa. Combining latent space and structured kernels for bayesian optimization over combinatorial spaces. *Advances in neural information processing systems*, 34:8185–8200, 2021.

Aryan Deshwal, Cory M Simon, and Janardhan Rao Doppa. Bayesian optimization of nanoporous materials. *Molecular Systems Design & Engineering*, 6 (12):1066–1086, 2021.

Aryan Deshwal, Syrine Belakaria, Janardhan Rao Doppa, and Dae Hyun Kim. Bayesian optimization over permutation spaces. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 6515–6523, 2022.

Aryan Deshwal, Sebastian Ament, Maximilian Balandat, Eytan Bakshy, Janardhan Rao Doppa, and David Eriksson. Bayesian optimization over high-dimensional combinatorial spaces via dictionary-based embeddings. In *International Conference on Artificial Intelligence and Statistics*, pages 7021–7039. PMLR, 2023.

Azza Fadhel, Yassine Chemingui, Minh Hoang, Aryan Deshwal, Trong Nghia Hoang, and Jana Doppa. Nanoporous materials discovery via search bias-guided surrogate modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 38422–38431, 2026a.

Azza Fadhel, Nathaniel W Zuckschwerdt, Aryan Deshwal, Susmita Bose, Amit Bandyopadhyay, and Jana

Doppa. Discovery of feasible 3d printing configurations for metal alloys via ai-driven adaptive experimental design. *arXiv preprint arXiv:2601.17587*, 2026b.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, 2017a.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1126–1135. PMLR, 2017b.

Nickolas Gantzler, Aryan Deshwal, Janardhan Rao Doppa, and Cory M Simon. Multi-fidelity bayesian optimization of covalent organic frameworks for xenon/krypton separations. *Digital Discovery*, 2(6): 1937–1956, 2023.

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*, pages 1704–1713. PMLR, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

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

Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. In *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018*, pages 45–87. Springer, 2020.

Jonathan Gordon, Wessel P Bruinsma, Andrew YK Foong, James Requeima, Yann Dubois, and Richard E Turner. Convolutional conditional neural processes. *arXiv preprint arXiv:1910.13556*, 2019.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

José M Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in neural information processing systems*, 27, 2014.

Minh Hoang, Azza Fadhel, Aryan Deshwal, Janardhan Rao Doppa, and Trong Nghia Hoang. Learning surrogates for offline black-box optimization via gradient matching. *arXiv preprint arXiv:2503.01883*, 2025.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019.

Minsu Kim, Federico Berto, Sungsoo Ahn, and Jinkyoo Park. Bootstrapped training of score-conditioned generator for offline design of biological sequences. *Advances in Neural Information Processing Systems*, 36:67643–67661, 2023.

Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Generative pretraining for black-box optimization. *arXiv preprint arXiv:2206.10786*, 2022.

Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. *arXiv preprint arXiv:2306.07180*, 2023.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *Advances in neural information processing systems*, 33:5126–5137, 2020.

Ronny Lorenz, Stephan H Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for molecular biology*, 6(1):26, 2011.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. *arXiv preprint arXiv:2207.04179*, 2022.

Tung Nguyen, Sudhanshu Agrawal, and Aditya Grover. Expt: Synthetic pretraining for few-shot experimental design. *Advances in Neural Information Processing Systems*, 36:45856–45869, 2023.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Gautam Singh, Jaesik Yoon, Youngsung Son, and Sungjin Ahn. Sequential neural processes. *Advances in Neural Information Processing Systems*, 32, 2019.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proc. NIPS*, pages 2960–2968, 2012a.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012b.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *Proc. ICML*, pages 2171–2180. PMLR, 2015.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

Rong-Xi Tan, Ke Xue, Shen-Huan Lyu, Haopu Shang, Yao Wang, Yaoyuan Wang, Sheng Fu, and Chao Qian. Offline model-based optimization by learning to rank. *arXiv preprint arXiv:2410.11502*, 2024.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *Proc. ICML*, pages 10358–10368. PMLR, 2021a.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pages 10358–10368. PMLR, 2021b.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization, 2022a. URL `https://arxiv.org/abs/2202.08450`.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pages 21658–21676. PMLR, 2022b.

Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. *Nature reviews Drug discovery*, 18(6):463–477, 2019.

Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel bayesian global optimization of expensive functions. *Operations Research*, 68(6):1850–1865, 2020.

Yining Wang, Simon Du, Sivaraman Balakrishnan, and Aarti Singh. Stochastic zeroth-order optimization in high dimensions. In *Proc. AISTATS*, pages 1356–1365. PMLR, 2018.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International conference on machine learning*, pages 3627–3635. PMLR, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Michael S. Yao, Yimeng Zeng, Hamsa Bastani, Jacob Gardner, James C. Gee, and Osbert Bastani. Generative adversarial model-based optimization via source critic regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL `https://arxiv.org/abs/2402.06532`.

Ye Yuan, Can Chen, Zixuan Liu, Willie Neiswanger, and Xue Liu. Importance-aware co-teaching for offline model-based optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL `https://arxiv.org/abs/2309.11600`.

Ye Yuan, Youyuan Zhang, Can Chen, Haolun Wu, Zixuan Li, Jianmo Li, James J Clark, and Xue Liu. Design editing for offline model-based optimization. *arXiv preprint arXiv:2405.13964*, 2024.

Taeyoung Yun, Sujin Yun, Jaewoo Lee, and Jinkyoo Park. Guided trajectory generation with diffusion models for offline model-based optimization. *Advances in Neural Information Processing Systems*, 37:83847–83876, 2024.

Checklist

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

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]

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

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

APPENDIX

A. ADDITIONAL PERFORMANCE COMPARISON WITH OTHER BASELINES

In addition to the 9 baseline methods in the main text, we further compare our method with 6 recent offline optimization techniques: (1) **DDOM** which learns a conditional diffusion generator $p_\theta(x \mid y)$ that enables guided sampling to generate high-score design candidates [Krishnamoorthy et al., 2023]; (2) **Parallel-mentoring (Tri-mentoring)** which learns three proxies to mentor each other via voting-based pairwise supervision and soft labels to strengthen the ensemble and curb OOD errors [Chen et al., 2023]; (3) **BDI** enforces a forward and backward mapping between static datasets and high-scoring designs using neural tangent kernel theory [Chen et al., 2022]; (4) **RGD** combines explicit proxy guidance with proxy-free diffusion modeling and iteratively refines the proxy via diffusion feedback [Chen et al., 2024], (5) **GTG** generates design trajectories by guiding diffusion models along high-value paths for offline optimization [Yun et al., 2024]; and (6) **PGS** reinterprets offline optimization as a reinforcement learning task, optimizing for an effective policy using sampled trajectories from offline data [Chemingui et al., 2024]. Other methods, such as IGNITE and BOSS [Dao et al., 2024, 2025a], are boosting frameworks that can be incorporated into arbitrary surrogate models. Therefore, we do not include these boosted variants as direct comparison baselines.

Table 5: Performance achieved by baselines under the limited data settings (using 1% of the offline dataset) across various benchmark tasks.

	Benchmarks			
Method	**Ant**	**D'Kitty**	**TFBind8**	**Mean Rank**
D_{best} from 1% offline data	0.123	0.307	0.124	None
DDOM	0.324 ± 0.024	0.746 ± 0.012	0.771 ± 0.121	5.33
Tri-mentoring	0.831 ± 0.015	0.928 ± 0.008	0.645 ± 0.030	4.0
BDI	0.298 ± 0.000	0.728 ± 0.000	0.447 ± 0.000	5.67
RGD	0.870 ± 0.022	0.888 ± 0.015	0.890 ± 0.057	3.17
GTG	0.825 ± 0.050	0.913 ± 0.020	0.890 ± 0.054	3.17
PGS	0.572 ± 0.021	0.882 ± 0.013	0.536 ± 0.051	4.83
OptBias (ours)	0.960 ± 0.017	0.947 ± 0.012	0.945 ± 0.024	1.00

B. HYPERPARAMETERS

We present our hyperparameter setting in this section. In our synthetic data generator (Alg. 1), we generate $n = 128$ simulated functions, and use the evolution operator $\mathcal{T}$ as gradient ascent with 100 gradient steps. The learning rate of this gradient process is 0.05 for continuous tasks and 0.005 for discrete tasks. The evolution operator $\mathcal{T}$ could be set as any other optimization method, such as a genetic program or local search. We currently use gradient search due to its simplicity. After the generation phase, we trained our meta-learner for E epochs, with $E = 50$ for continuous tasks and $E = 100$ for discrete tasks, using a learning rate of $\eta = 0.001$. After training, we fine-tuned the model for 20 epochs and then leveraged the fine-tuned model for gradient search with a step size of $\gamma = 0.001$ for continuous tasks, $\gamma = 0.1$ for discrete tasks, and 300 steps to obtain our optimal designs. We provide a comprehensive report of the hyperparameters used in our implementation for both continuous and discrete tasks.

C. ABLATION STUDIES

We vary the number of GP-sampled functions used during meta-training, $K \in 8, 16, 32, 64, 128$. Each function defines a different optimization task, so the meta-learner must focus on patterns shared across tasks rather than on task-specific details. Sets are nested (smaller K is a subset of larger K). *Mean Rank* averages over tasks with values for all methods. Performance stabilizes for $K \geq 64$, with only small gains at $K = 128$ (see Table 6 in Appendix), showing that a modest number of synthetic functions is enough.

Overall, increasing the number of meta-training functions (K) consistently improves performance across tasks, indicating more diverse training functions enhance generalization.

Continuous (e.g., AntMorphology)

Setting	Value
Num context / target	16 / 64
Batch size	128
Epochs (E)	50
Inner-loop learning rate	0.1
Meta learning rate	0.001
Norm layer	BatchNorm
Meta-training function generator (GP)	
Num functions (n)	128
Kernel	RBF
Init lengthscale / variance	1.0 / 1.0
Noise	0.01
Top-k fraction (poorest data)	0.01
Inner gradient steps	100
Learning rate (GP sim)	0.05
Finetuning / gradient search	
Finetuning epochs	20
Gradient search iterations	300
Stepsize (γ)	0.001

Discrete (e.g., TFBind-8)

Setting	Value
Num context / target	16 / 64
Batch size	128
Epochs (E)	100
Inner-loop learning rate	0.1
Meta learning rate	0.001
Norm layer	BatchNorm
Meta-training function generator (GP)	
Num functions (n)	128
Kernel	RBF
Init lengthscale / variance	6.25 / 6.25
Noise	0.01
Top-k fraction (poorest data)	0.01
Inner gradient steps	100
Learning rate (GP sim)	0.005
Finetuning / gradient search	
Finetuning epochs	25
Gradient search iterations	300
Stepsize (γ)	0.1

Table 6: Hyperparameters for continuous (left) and discrete (right) tasks.

Method (K)	Ant	D'Kitty	TFBind8	RNA1	RNA2	RNA3	Mean Rank
OptBias (128)	**0.960 ± 0.017** (1)	**0.947 ± 0.012** (1)	**0.945 ± 0.024** (1)	0.503 ± 0.025 (2)	**0.540 ± 0.113** (1)	0.450 ± 0.134 (4)	**1.67**
OptBias (64)	0.912 ± 0.082 (3)	0.938 ± 0.029 (2)	0.916 ± 0.087 (2)	**0.541 ± 0.149** (1)	0.531 ± 0.115 (2)	**0.550 ± 0.035** (1)	1.83
OptBias (16)	0.926 ± 0.060 (2)	0.906 ± 0.050 (3)	0.855 ± 0.080 (4)	0.437 ± 0.137 (3)	0.453 ± 0.136 (3)	0.539 ± 0.044 (2)	2.83
OptBias (8)	0.823 ± 0.130 (4)	0.890 ± 0.081 (4)	0.871 ± 0.060 (3)	0.368 ± 0.214 (4)	0.377 ± 0.183 (4)	0.471 ± 0.088 (3)	3.67

Table 7: Performance of **OptBias** under different K across Design-Bench tasks. Higher is better. Bold marks the best value (and its rank) per task, and the best mean rank.

D. MODEL ARCHITECTURE

We use a lightweight meta-learning regression network: a small feed-forward stack **with hidden layers [512, 128, 32]** and normalization + LeakyReLU activations, followed by a linear head to a single scalar; it supports externally supplied (inner-loop) parameters and step-specific normalization statistics for fast adaptation across tasks.

E. SCALABILITY OF GAUSSIAN PROCESSES WITHIN SIM4OPT

Gaussian Processes are known to scale cubically with the number of training points. However, the focus of this work is on data-scarce offline optimization settings where the available dataset is typically small. In such regimes the computational cost remains manageable even in moderately high-dimensional spaces. For example, the datasets used in our experiments already have relatively high input dimensions: the ANT task uses 60-dimensional inputs ($d = 60$) and the DKitty task uses 56-dimensional inputs ($d = 56$). The training complexity of GPs is $O(n^3 + dn^2)$, where n is the dataset size and d is the input dimension, which scales linearly with respect to d. For larger datasets, scalable GP approximations such as sparse Gaussian Processes can be used to further reduce computational cost. These methods approximate the full covariance structure using a set of inducing points, allowing the training complexity to scale approximately linearly with the number of data points while maintaining strong predictive performance [Hensman et al., 2013].

F. GPU COMPUTE FOR EXPERIMENTS

All experiments were run on a single **NVIDIA A40 (46 GB)** with **CUDA 12.9** and driver **575.51.03**.

G. CODE AVAILABILITY

The official implementation of this work is publicly available at: `https://github.com/azzafadhel/OptBias.git`