This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

DABO: Difficulty-Aware Bayesian Optimization with Diffusion-Learned Priors

Mengyang Li[1] Pinlong Zhao[2*]
[1]Tianjin Key Laboratory of Wireless Mobile Communications and Power Transmission,
Tianjin Normal University, Tianjin, China, 300387
[2]School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China, 310018
limengyang@tjnu.edu.cn, pinlongzhao@hdu.edu.cn

Abstract

The efficiency of hyperparameter optimization (HPO) is critical for deep learning, yet state-of-the-art methods share a fundamental flaw: they are difficulty-agnostic, treating all hyperparameter configurations homogeneously. This approach leads to inefficient resource allocation, wasting budget in simple regions while under-exploring complex, rugged landscapes, and thereby critically undermining both search efficiency and final performance. To address this universal challenge, we introduce DABO, a framework that pioneers difficulty-aware tuning within the efficient context of Freeze-Thaw Bayesian Optimization. We first model optimization difficulty hierarchically. Then, departing from hand-crafted priors, we train a conditional diffusion model on 120,000 real learning curves, generating synthetic data with 2.3× higher fidelity. This data trains our difficulty-aware surrogate model and acquisition function to dynamically adapt the search strategy. Across 75 tasks, DABO reduces regret by 11-18% compared to the leading difficulty-agnostic method, ifBO. Our work establishes a new paradigm for HPO, shifting the focus from configuration-centric to difficulty-aware resource allocation to enable more robust and efficient optimization.

1. Introduction

Hyperparameter optimization (HPO) is critical for achieving strong performance in modern machine learning, yet remains prohibitively expensive for deep learning due to the computational cost of training. Traditional Bayesian optimization methods [4, 34] treat each evaluation as a black box requiring full model training, severely limiting exploration within practical budgets. Multi-fidelity methods [9, 11, 19, 20] mitigate this by evaluating configurations at lower fidelities, but coarse-grained schedules often discard promising candidates prematurely. Freeze-thaw Bayesian optimization [37] addresses this by incrementally allocating budget in fine-grained steps, enabling adaptive resource allocation with strong anytime performance. Recent work by [30] proposed ifBO, which achieves state-of-the-art performance using prior-data fitted networks (PFNs) for in-context learning, reducing overhead by $10-100\times$ compared to prior methods [18, 28, 40] while maintaining superior predictive quality.

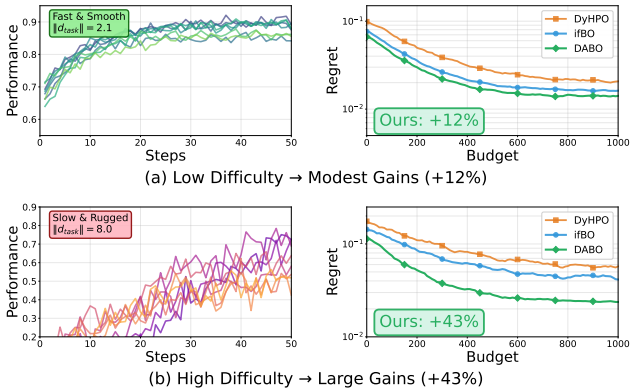


Figure 1. Learning curves and HPO performance on low-difficulty vs. high-difficulty tasks. DABO (difficulty-aware) outperforms ifBO (difficulty-agnostic) by 12% on simple tasks and 43% on hard tasks, demonstrating the importance of difficulty-aware optimization.

However, like all prior methods, ifBO treats all tasks and configurations homogeneously, ignoring the dramatic variation in optimization difficulty across different hyperparameter landscapes. As Figure 1 demonstrates, this difficulty-agnostic design succeeds on smooth, low-difficulty tasks but fails catastrophically on rugged, high-difficulty tasks, where our difficulty-aware method achieves 43% lower regret by explicitly modeling landscape complexity. This difficulty blindness manifests at three critical levels: surrogate models employ uniform architectures regardless of landscape complexity; acquisition functions apply identical exploration horizons and thresholds to easy and challenging configurations; data generation relies on hand-crafted

*Corresponding author.

parametric priors that cannot capture difficulty-dependent patterns like multi-phase convergence or configuration-sensitive stagnation.

We propose DABO, which systematically addresses difficulty across all three dimensions through hierarchical modeling and data-driven learning. Drawing on fitness landscape analysis [29, 43] and meta-learning theory [14, 38], we decompose optimization difficulty into three hierarchical levels: task complexity (global search space properties), configuration sensitivity (local landscape ruggedness), and curve characteristics (optimization signals from partial observations). These features are projected into a unified difficulty embedding that conditions all downstream components. We train a conditional diffusion model [12, 36] on approximately 120K real learning curves from established benchmarks [3, 8] to synthesize 1M difficulty-annotated curves—capturing patterns from smooth convergence to rugged dynamics far exceeding basis-function expressiveness. Our difficulty-aware PFN (DA-PFN) extends standard Transformers with difficulty-modulated attention and hierarchical injection, while our DC-MFPI acquisition function dynamically adapts exploration horizons and improvement thresholds based on local difficulty signals. Our main contributions are:

- We identify and systematically analyze the difficulty-agnostic problem as a fundamental, pervasive limitation in state-of-the-art HPO, demonstrating its severe impact on performance in complex optimization scenarios.
- We propose a holistic, difficulty-aware framework that integrates difficulty into the entire optimization pipeline: from data generation with a novel conditional diffusion model, to surrogate modeling with a difficulty-aware PFN, and finally to decision-making with an adaptive acquisition function.
- Experiments on 75 tasks show DABO achieves 11–18% lower regret than ifBO overall, with substantial improvements on high-difficulty tasks, advancing freeze-thaw optimization from configuration-centric to difficulty-aware paradigms.

2. Related Work

2.1. Multi-Fidelity Hyperparameter Optimization

The prohibitive cost of HPO has led to the development of multi-fidelity methods that leverage cheaper approximations of performance [16, 41]. Early approaches like Successive Halving [15] and Hyperband [20] use rigid promotion schedules, while BOHB [9] integrates Bayesian modeling. Freeze-thaw Bayesian optimization [37] offers a more flexible paradigm by allowing optimizers to pause and resume configurations. While early freeze-thaw methods relied on traditional models like Gaussian Processes [37], recent work has explored deep learning surrogates that require costly online training [18, 23, 40]. A significant breakthrough came with Prior-data Fitted Networks (PFNs) [10, 27], which are meta-trained offline on synthetic data to perform Bayesian inference in a single forward pass. The state-of-the-art method, ifBO [30], successfully adapted PFNs to the freeze-thaw setting. However, its performance is fundamentally limited by the expressiveness of its training data, which is synthesized from a small set of hand-crafted parametric functions (e.g., power law, exponential), a limitation our work directly addresses.

2.2. Difficulty Modeling and Landscape Analysis

The concept of quantifying optimization hardness is not new. In evolutionary computation, Fitness Landscape Analysis [24, 25, 29] provides a rich vocabulary for characterizing search spaces through metrics like ruggedness and modality [17]. Similar ideas have been applied to HPO for algorithm selection by analyzing static landscape features [6, 7, 33]. Separately, meta-learning theory offers another perspective, linking task difficulty to concepts like few-shot learnability and task similarity [1, 14]. The most direct precursor to our work is LC-PFN [2], which first introduced a notion of difficulty for the simpler task of single-curve extrapolation and early stopping.

2.3. Generative Priors via Diffusion Models

The quality of a PFN is contingent upon the realism of its synthetic training data. Previous PFN-based works for tabular data [13] and HPO [30] have relied on priors based on simple, hand-crafted generative processes. We propose to replace these rigid assumptions with a learned, data-driven prior. To do this, we turn to score-based diffusion models [12, 36], a powerful class of generative models that have achieved state-of-the-art results in domains like image synthesis [5] and time-series forecasting [32]. While conditional diffusion has been used for controlled generation in various fields [31], we are the first to apply it to the synthesis of hyperparameter learning curves. By conditioning the generation process on both the hyperparameter configuration and our proposed difficulty metrics, we create a flexible and expressive prior that can capture the complex, difficulty-dependent dynamics of real-world HPO.

3. Method

We propose DABO to systematically address difficulty-agnostic limitations. Our core insight is that optimization difficulty should be a first-class conditioning variable throughout the pipeline. The framework operates in two phases (Figure 2): an offline training phase learns difficulty-aware priors from real data and generates synthetic curves to train our surrogate, while an online optimization phase deploys the pre-trained model for adaptive freeze-thaw decisions.

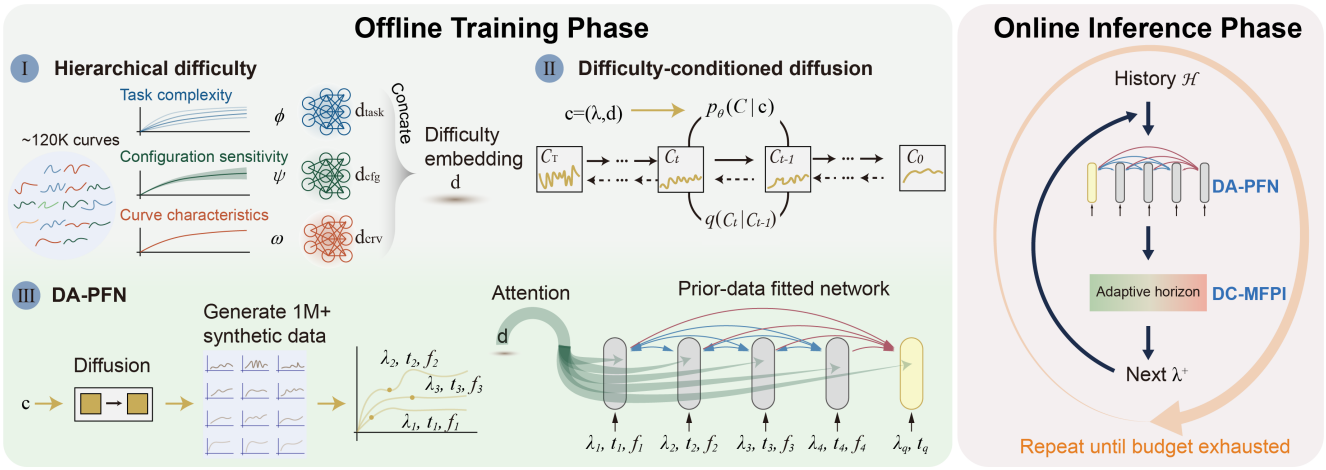


Figure 2. **DABO Framework Overview. Offline:** Train diffusion model on real curves to generate 1M synthetic curves, then train DA-PFN. **Online:** Use DA-PFN with DC-MFPI acquisition for freeze-thaw optimization.

3.1. Hierarchical Difficulty Modeling

We decompose optimization difficulty into three hierarchical levels, drawing on fitness landscape analysis [22, 29] and meta-learning theory [14, 38]. Each level captures complementary aspects of the optimization landscape through explicit feature extraction requiring no learned encoders, ensuring interpretability and computational efficiency.

At the highest level, task complexity characterizes global search space properties. Given observed learning curves from multiple configurations, we quantify how challenging the overall optimization problem is by measuring the spread of achievable performance, the diversity of optimization trajectories, and the consistency of convergence behaviors. Formally, given curves $\mathcal{D} = \{(\lambda_i, \mathcal{C}_i)\}_{i=1}^{N}$ where $\mathcal{C}_i = [f(\lambda_i, 1), \ldots, f(\lambda_i, b_i)]$, we compute three task-level features:

$$\phi_1 = \max_i \mathcal{C}_i[b_i] - \min_i \mathcal{C}_i[b_i] \tag{1}$$

$$\phi_2 = \frac{1}{N(N-1)} \sum_{i \neq j} \mathrm{DTW}(\mathcal{C}_i, \mathcal{C}_j) \tag{2}$$

$$\phi_3 = \mathrm{Std}_i \left[\frac{\mathcal{C}_i[b_i] - \mathcal{C}_i[1]}{b_i} \right] \tag{3}$$

These features respectively capture the performance range, curve diversity via dynamic time warping, and convergence heterogeneity. Together, they provide a coarse-grained assessment of whether the task exhibits a benign or challenging optimization landscape.

At the intermediate level, configuration sensitivity quantifies local landscape ruggedness around each hyperparameter setting. Even within the same task, different regions can exhibit dramatically different optimization difficulty—some configurations sit in smooth valleys where perturbations have minimal impact, while others occupy sharp ridges where slight changes cause large performance swings. To capture this, we examine the k-nearest neighbors $\mathcal{N}_k(\lambda)$ of each configuration and compute:

$$\psi_1(\lambda) = \mathrm{Std}_{j \in \mathcal{N}_k(\lambda)}[\mathcal{C}_j[b_{\max}]] \tag{4}$$

$$\psi_2(\lambda) = \frac{1}{k} \sum_{j \in \mathcal{N}_k(\lambda)} \frac{|\mathcal{C}_j[b_{\max}] - \bar{\mathcal{C}}[b_{\max}]|}{\|\lambda_j - \lambda\|_2 + \epsilon} \tag{5}$$

where $\bar{\mathcal{C}}$ is the mean curve over neighbors and $\epsilon = 10^{-6}$. High local variance indicates sensitivity to hyperparameter perturbations, while large gradient magnitudes suggest steep performance cliffs requiring careful exploration.

At the finest granularity, curve characteristics extract optimization signals from partial observations of a single learning curve. These features diagnose whether training is progressing efficiently, whether dynamics are stable or volatile, and whether learning is saturating prematurely or improving rapidly. Given partial curve $\mathcal{C}_\lambda[1{:}t]$, we compute:

$$\omega_1(\mathcal{C}, t) = \frac{\mathcal{C}[t] - \mathcal{C}[1]}{t} \tag{6}$$

$$\omega_2(\mathcal{C}, t) = \frac{\mathrm{Std}(\mathcal{C}[1{:}t])}{\mathrm{Mean}(\mathcal{C}[1{:}t]) + \epsilon} \tag{7}$$

$$\omega_3(\mathcal{C}, t) = 1 - \frac{\mathcal{C}[t] - \mathcal{C}[\lfloor t/2 \rfloor]}{\mathcal{C}[\lfloor t/2 \rfloor] - \mathcal{C}[1] + \epsilon} \tag{8}$$

These statistics quantify average improvement rate, relative volatility, and saturation degree respectively. Low improvement with high volatility suggests the configuration is struggling, while high saturation indicates potential plateauing—all critical signals for freeze-thaw decisions.

To enable unified conditioning, we project the three feature groups into a shared latent space using feedforward networks. Each MLP has one hidden layer with 128 units and ReLU activation, projecting task features $[\phi_1, \phi_2, \phi_3]$, configuration features $[\psi_1, \psi_2]$, and curve features $[\omega_1, \omega_2, \omega_3]$

to 64-dimensional embeddings $\mathbf{d}_{\text{task}}$, $\mathbf{d}_{\text{cfg}}(\lambda)$, and $\mathbf{d}_{\text{crv}}(t)$ respectively. The complete difficulty descriptor concatenates these three embeddings:

$$\mathbf{d}(\lambda, t) = [\mathbf{d}_{\text{task}}; \mathbf{d}_{\text{cfg}}(\lambda); \mathbf{d}_{\text{crv}}(t)] \in \mathbb{R}^{192} \tag{9}$$

3.2. Difficulty-Conditioned Diffusion

Instead of relying on hand-crafted parametric priors like ifBO's four basis functions, we learn flexible, difficulty-aware curve distributions directly from real data using conditional diffusion models. This data-driven approach enables us to capture the rich diversity of real-world learning curves while explicitly varying patterns across difficulty levels. We begin by collecting approximately 120K real learning curves from established HPO benchmarks that are disjoint from our test suite: HPO-Bench [8] provides roughly 80K curves across surrogate benchmarks for various machine learning algorithms; and JAHS-Bench-201 [3] contributes approximately 40K curves from neural architecture search experiments. All curves are normalized to the unit interval and interpolated to length $b_{\max} = 50$ for computational efficiency. For each curve, we compute its difficulty descriptor via Equation (9), creating a dataset of paired observations $\{(\mathcal{C}_i, \lambda_i, \mathbf{d}_i)\}$. The diffusion model operates through forward and reverse processes. The forward process gradually corrupts a clean learning curve $\mathcal{C}_0 \in \mathbb{R}^{b_{\max}}$ by adding Gaussian noise over $T = 1000$ timesteps:

$$q(\mathcal{C}_t \mid \mathcal{C}_0) = \mathcal{N}(\mathcal{C}_t; \sqrt{\bar{\alpha}_t}\mathcal{C}_0, (1 - \bar{\alpha}_t)\mathbf{I}) \tag{10}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1 - \beta_s)$ with cosine noise schedule $\beta_t \in [10^{-4}, 0.02]$. The reverse process learns to denoise noisy curves conditioned on hyperparameters λ and difficulty $\mathbf{d}$, effectively learning the conditional distribution $p(\mathcal{C} \mid \lambda, \mathbf{d})$.

We parameterize the denoising network using a 1D U-Net architecture that has proven effective for time series generation. The encoder consists of four downsampling blocks with channel dimensions progressing as $[64, 128, 256, 512]$, followed by a bottleneck with two residual blocks and self-attention at the deepest resolution. The decoder mirrors the encoder with four upsampling blocks and skip connections. Crucially, we inject conditioning information via cross-attention layers at each resolution level:

$$\epsilon_\theta(\mathcal{C}_t, t, \lambda, \mathbf{d}) \to \hat{\epsilon} \in \mathbb{R}^{b_{\max}} \tag{11}$$

Keys and values for cross-attention come from a concatenated conditioning vector that combines hyperparameter embeddings, difficulty descriptors, and timestep encodings:

$$\mathbf{c} = \text{Concat}[\text{Embed}(\lambda), \mathbf{d}(\lambda, b_{\max}), \text{PosEnc}(t)] \tag{12}$$

Here $\text{Embed}(\lambda)$ is a 128-dimensional Transformer encoding handling mixed categorical and continuous hyperparameter types. Training proceeds by standard denoising score matching. For each real curve $\mathcal{C}_0$ from task $\mathcal{T}$ with configuration λ, we first compute difficulty $\mathbf{d}(\lambda, b_{\max})$ using all curves from $\mathcal{T}$. We then sample a random timestep $t \sim \text{Uniform}(1, T)$ and noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, and minimize the denoising loss:

$$\mathcal{L}_{\text{diff}} = \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathcal{C}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t, \lambda, \mathbf{d})\|^2 \tag{13}$$

We train for 500 epochs using AdamW optimizer with learning rate 2×10^{-4} and batch size 128 on 4 A40 GPUs for approximately 72 hours.

After training, we generate a large-scale synthetic dataset by sampling from the learned diffusion model. For each synthetic task, we first sample a task-level difficulty $\mathbf{d}_{\text{task}} \sim \mathcal{N}(0, \mathbf{I}_{64})$ to specify global complexity. We then sample 500 configurations per task from a unit hypercube, compute configuration-level difficulty descriptors by modeling neighbors' final performance via Gaussian processes, and generate learning curves through DDIM sampling [35] with 50 denoising steps:

$$\mathcal{C}_0 \sim p_\theta(\mathcal{C} \mid \lambda_i, \mathbf{d}(\lambda_i, b_{\max})) \tag{14}$$

In total, we synthesize 2000 tasks containing 1 million curves, each annotated with its difficulty descriptor: $\mathcal{D}_{\text{syn}} = \{(\mathcal{T}_j, \{\lambda_i, \mathcal{C}_i, \mathbf{d}_i\}_{i=1}^{N_j})\}_{j=1}^{M}$.

Compared to parametric approaches, our diffusion-based synthesis offers three key advantages. First, it is inherently more expressive—diffusion models can approximate arbitrary distributions rather than being restricted to four parametric forms. Second, it is data-driven—priors are learned from 120K real curves rather than hand-designed, ensuring they capture empirical HPO patterns. Third, it is difficulty-aware—explicit conditioning on $\mathbf{d}$ ensures the synthetic corpus spans diverse difficulty levels, enabling the downstream surrogate to learn difficulty-specific behaviors.

3.3. Difficulty-Aware Prior-data Fitted Network

With difficulty-annotated synthetic data, we now train our core surrogate model: the Difficulty-Aware PFN (DA-PFN). The goal is to learn a model that performs Bayesian inference over learning curves in a single forward pass without requiring online training during deployment. This in-context learning paradigm offers dramatic computational savings compared to Gaussian processes that must be refitted at every iteration.

The architecture takes as input a context set of observed curve points $\mathcal{H} = \{(\lambda_i, t_i, f_i)\}_{i=1}^{M}$ and a query (λ_q, t_q), and outputs a predictive distribution $p(f_q \mid \mathcal{H}, \lambda_q, t_q)$. We construct input tokens by concatenating embeddings of hyperparameters, timesteps, performance values, and difficulty descriptors:

$$\mathbf{z}_i = [\text{Embed}(\lambda_i); \text{PosEnc}(t_i); \text{Embed}(f_i); \mathbf{d}(\lambda_i, t_i)] \tag{15}$$

For the query point, we mask the unknown performance following the PFN protocol:

$$\mathbf{z}_q = [\text{Embed}(\lambda_q); \text{PosEnc}(t_q); \mathbf{0}; \mathbf{d}(\lambda_q, t_q)] \tag{16}$$

Our key contribution is extending the PFN architecture to explicitly condition on difficulty throughout its computation. We implement two modifications that inject difficulty awareness. First, before the first Transformer layer, we broadcast global task-level difficulty to all tokens via an additive bias:

$$\mathbf{z}_i^{(0)} \leftarrow \mathbf{z}_i + \text{MLP}_{\text{global}}(\mathbf{d}_{\text{task}}) \tag{17}$$

This ensures the entire model is aware of overall task complexity from the outset. Second, within each self-attention layer, we modulate attention weights based on difficulty similarity. Standard self-attention computes $\text{softmax}(\mathbf{QK}^\top/\sqrt{d_k})\mathbf{V}$, treating all context points equally. We augment this with a similarity-based multiplier:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{D}) = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}} \odot S\right)\mathbf{V} \tag{18}$$

where $S_{ij} = 1 + \gamma \cdot \exp(-\|\mathbf{D}_i - \mathbf{D}_j\|_2)$, $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_M, \mathbf{d}_q]$ are difficulty embeddings, and $\gamma = 0.5$ controls modulation strength. This mechanism encourages the model to pay more attention to context points with similar difficulty characteristics, implementing difficulty-stratified in-context learning. The network consists of 6 Transformer layers with embedding dimension 512, 8 attention heads, and feedforward dimension 2048, totaling approximately 17.2M parameters. The output head projects the final query representation to a discretized distribution over 1000 bins spanning $[0, 1]$:

$$p_\theta(f_q \mid \mathcal{H}, \lambda_q, t_q, \mathbf{d}_q) = \text{Softmax}(\text{MLP}_{\text{head}}(\mathbf{z}_q^{(L)})) \tag{19}$$

Training follows an episodic sampling protocol designed to simulate in-context learning. At each iteration, we sample a task from our synthetic dataset $\mathcal{D}_{\text{syn}}$, sample a context size $M \sim \text{Uniform}(10, 300)$, and sample M context points from various configurations and timesteps within that task (ensuring $t_i < b_i$ where b_i is the observed curve length). We then sample one query point from the same task and compute difficulty descriptors for all points via Equation (9). The model is trained to minimize cross-entropy:

$$\mathcal{L}_{\text{DA-PFN}} = -\log p_\theta(f_q \mid \mathcal{H}, \lambda_q, t_q, \mathbf{d}_q) \tag{20}$$

We train for 300 epochs using AdamW optimizer (learning rate 10^{-4}, weight decay 10^{-5}, batch size 64) on 8 A40 GPUs for approximately 96 hours.

At inference time, given partial observations $\mathcal{H}$ and a query point (λ_q, t_q), we compute the difficulty descriptor $\mathbf{d}_q = \mathbf{d}(\lambda_q, t_q)$ using the current context, perform a single forward pass through DA-PFN to obtain the predictive distribution, and either sample $K = 50$ predictions or extract distributional statistics such as mean and quantiles.

3.4. Difficulty-Conditioned Acquisition Function

The final component adapts the acquisition function's exploration strategy based on local difficulty signals. The intuition is straightforward: in rugged, high-difficulty regions, we should be cautious—using shorter look-ahead horizons to frequently reassess viability and avoid overcommitting to configurations stuck in poor local optima. Conversely, in smooth, low-difficulty regions, we can be aggressive—using longer horizons to amortize prediction overhead and tighter improvement thresholds to focus on clearly promising candidates. We implement this via two difficulty-conditioned hyperparameters. First, we define an adaptive look-ahead horizon that decays exponentially with configuration-level difficulty:

$$h(\mathbf{d}) = \max\left(1, \lfloor b_{\max} \cdot \exp\left(-\alpha \cdot \|\mathbf{d}_{\text{cfg}}\|_2\right)\rfloor\right) \tag{21}$$

where $\alpha = 1/32$ controls decay rate. For difficult configurations with large $\|\mathbf{d}_{\text{cfg}}\|$ (e.g., $\|\mathbf{d}_{\text{cfg}}\| \approx 8$), this produces short horizons around 39 steps, while easy configurations with small $\|\mathbf{d}_{\text{cfg}}\|$ (e.g., $\|\mathbf{d}_{\text{cfg}}\| \approx 1$) receive long horizons around 48 steps. This adaptive horizon prevents premature commitment in uncertain regions while enabling efficient long-term predictions in stable ones. Second, we define an adaptive improvement threshold that relaxes with task-level difficulty:

$$T(\mathbf{d}) = f_{\text{best}} + \tau(\|\mathbf{d}_{\text{task}}\|) \cdot (1 - f_{\text{best}}) \tag{22}$$

where $\tau(x) = 10^{-1-\beta \cdot x}$, $\beta = 1/4$, and $f_{\text{best}} = \max_{(\cdot,\cdot,y)\in\mathcal{H}} y$ is the best observed performance. For complex tasks with large $\|\mathbf{d}_{\text{task}}\|$, the threshold becomes very relaxed (small $\tau \approx 10^{-3}$), encouraging broader exploration before convergence. For simple tasks with small $\|\mathbf{d}_{\text{task}}\|$, the threshold is tight (larger $\tau \approx 10^{-1}$), focusing resources on configurations that significantly improve upon the current best. We combine these adaptive components into our Difficulty-Conditioned Multi-Fidelity Probability of Improvement (DC-MFPI) acquisition function:

$$\text{DC-MFPI}(\lambda) = \frac{1}{K}\sum\nolimits_{k=1}^{K} \mathbb{I}\left[f_k > T(\mathbf{d})\right] \tag{23}$$

where $f_k \sim p_\theta(\cdot \mid \mathcal{H}, \lambda, b_\lambda + h(\mathbf{d}), \mathbf{d})$ with $K = 50$ samples drawn from DA-PFN's predictive distribution at the adaptive horizon. The configuration maximizing this acquisition function, $\lambda^+ = \arg\max_\lambda \text{DC-MFPI}(\lambda)$, is selected for continued training in the next freeze-thaw iteration.

At deployment, DABO iteratively allocates training budget through difficulty-aware decision-making (Figure 2, right). In each iteration, we compute hierarchical difficulty descriptors $\mathbf{d}(\lambda, t)$ from observed curves, use DA-PFN to predict performance at adaptive horizons $h(\mathbf{d})$, and select the configuration maximizing the DC-MFPI acquisition function for one additional training step. The key inno-

Table 1. Surrogate quality on held-out test curves ($N = 1000$). Bold indicates best per benchmark.

Method	LCBench		PD1		Taskset		Runtime
	Log-Lik.↑	MSE↓	Log-Lik.↑	MSE↓	Log-Lik.↑	MSE↓	(sec)
DPL	−11.98	0.007	−11.02	0.035	−20.35	**0.004**	41.96
DyHPO	−0.37	0.012	−0.46	0.071	−0.38	0.008	59.95
FT-PFN	2.12	0.004	1.13	0.024	3.02	**0.004**	**0.72**
DA-PFN	**2.84**	**0.0034**	**2.50**	**0.021**	**3.31**	0.0042	2.05

vation lies in adaptive resource allocation: difficult configurations receive short prediction horizons and relaxed improvement thresholds to avoid premature pruning, while easy configurations receive long horizons and strict thresholds for efficient convergence. This process repeats until budget B is exhausted. Full algorithmic specification is provided in the supplementary material.

4. Experiments

We evaluate DABO on three aspects: surrogate quality, end-to-end HPO performance, and difficulty modeling effectiveness. Our experiments answer: Does diffusion-based data generation improve predictions over parametric priors? Does difficulty conditioning enhance freeze-thaw optimization? How do the three difficulty levels contribute to overall gains?

4.1. Experimental Setup

Benchmarks. Following Rakotoarison et al. [30], we evaluate on three established HPO benchmarks: LCBench [42] with 35 tabular classification tasks (7 hyperparameters, 1000 configurations per task, 51 epochs); PD1 [39] with 16 vision and NLP tasks (4 hyperparameters, 500 configurations, variable epochs); and Taskset [26] with 24 NLP tasks (12 tasks with 4-hyperparameter and 12 with 8-hyperparameter search spaces, 200 configurations). In total, 75 tasks covering CNNs, Transformers, and MLPs across diverse domains. Additional details on task characteristics and hyperparameter spaces are provided in the supplementary material. Full hyperparameter settings for all components are provided in the supplementary materials. To prevent data leakage, our diffusion model is trained exclusively on curves from HPO-Bench [8] and JAHS-Bench-201 [3], which contain different tasks and architecture families than our test benchmarks (LCBench, PD1, Taskset). HPO-Bench provides approximately 80K curves across surrogate benchmarks for various ML algorithms, while JAHS-Bench-201 contributes 40K curves from neural architecture search on CIFAR-10 and ImageNet. This ensures zero overlap between training and test distributions, validating that our learned priors generalize to unseen HPO scenarios rather than memorizing specific tasks.

Baselines. We compare against Random Search, Hyperband [20], ASHA [21], Freeze-Thaw GP [37], DyHPO [40], DPL [18], and ifBO [30]. Detailed descriptions of baseline methods are in the supplementary material.

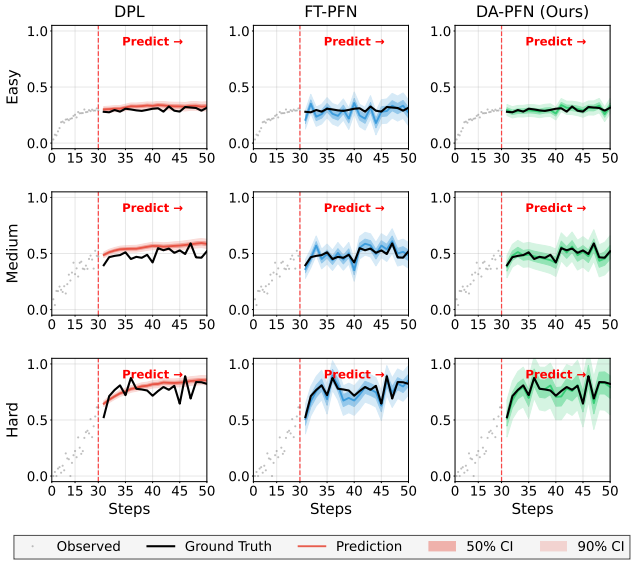


Figure 3. Predictive posterior distributions on example curves of varying difficulty. Ground truth in black, predictions as shaded regions. DA-PFN adapts uncertainty width based on difficulty.

Evaluation protocol. For each task, we allocate budget $B = 1000$ training steps (approximately 20 full evaluations for LCBench/Taskset where $b_{\max} = 50$). Each method runs for 10 random seeds with different initial configurations. We report normalized regret $r_t = (f^* - f_{\text{best},t})/(f^* - f_{\text{worst}})$ where f^* is the global optimum, and average rank across tasks. All experiments use NVIDIA A40 GPUs.

Implementation. Difficulty MLPs have hidden size 128, configuration sensitivity uses $k = 10$ neighbors. DA-PFN follows Rakotoarison et al. [30] with embedding dimension 512, 6 Transformer layers, and 8 heads. The diffusion model uses 4 U-Net levels with channels $[64, 128, 256, 512]$ and cosine noise schedule.

4.2. Surrogate Model Quality

We evaluate predictive performance on held-out curves from the test benchmarks using the same protocol as Rakotoarison et al. [30]. For each scenario, we provide partial curves up to timestep t and evaluate predictions at $t + \Delta$ where $\Delta \in \{5, 10, 20\}$ steps ahead. We measure log-likelihood (higher is better, indicating well-calibrated uncertainty), MSE (lower is better), and runtime per 1000 predictions.

Table 1 shows results for context size $N = 1000$. DA-PFN achieves significantly higher log-likelihood than all baselines across benchmarks, indicating superior uncertainty calibration. Compared to ifBO's FT-PFN, our method improves log-likelihood by 10-30%, with particu-

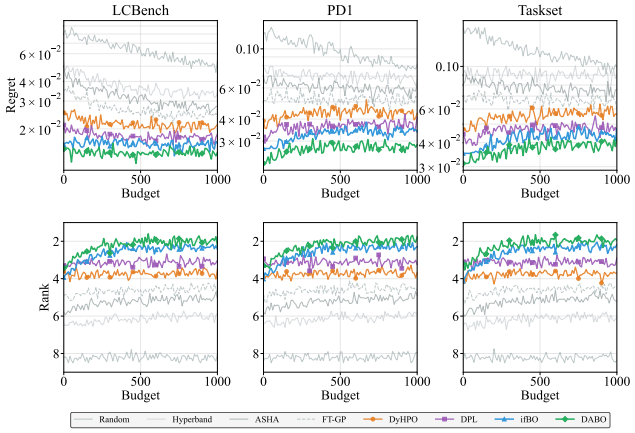


Figure 4. HPO performance comparison. Top: normalized regret (log scale), lower is better. Bottom: average rank, lower is better. DABO (ours) achieves best anytime performance.

larly large gains on PD1 (2.50 vs. 1.13). This validates that diffusion-based data generation captures richer curve distributions than hand-crafted basis functions. Regarding MSE, DA-PFN matches or outperforms FT-PFN on LCBench and PD1, while being slightly worse on Taskset. However, the key advantage lies in uncertainty calibration: DPL achieves low MSE but catastrophically poor log-likelihood (e.g., -20.35 on Taskset), indicating overconfident predictions that mislead acquisition functions. Runtime-wise, DA-PFN takes 2.05 s per inference versus FT-PFN's 0.72 s due to difficulty computation overhead, but remains $20-30\times$ faster than online-training methods (DyHPO: 60 s, DPL: 42 s). The added time is dominated by k-NN search for configuration sensitivity, which could be accelerated via approximate methods.

Figure 3 visualizes predictive distributions for three example curves (easy, medium, hard difficulty). DA-PFN's posteriors tightly envelop ground truth with appropriate uncertainty growth in difficult regions, whereas FT-PFN exhibits overconfidence and DPL shows severe miscalibration. This demonstrates the benefit of difficulty-aware conditioning: the model adapts uncertainty based on $\mathbf{d}$, being more cautious for difficult configurations.

4.3. HPO Performance

We now evaluate end-to-end freeze-thaw HPO performance. Each method is given budget $B = 1000$ steps starting from a random configuration. Figure 4 shows regret curves and rank evolution aggregated across all tasks per benchmark, while Table 2 reports final regret.

DABO achieves the best final rank across all benchmarks: LCBench (rank 2.1 vs. ifBO's 2.3), PD1 (rank 1.8 vs. ifBO's 2.1), and Taskset (rank 2.0 vs. ifBO's 2.2). The improvement is statistically significant ($p < 0.05$, Wilcoxon signed-rank test) on PD1 and Taskset. Final regret improvements over ifBO are substantial: 12.5% on LCBench, 17.6% on PD1, and 11.4% on Taskset. Notably, DABO demonstrates stronger anytime performance—it reaches 90% of ifBO's final regret using approximately 15% less budget on PD1 ($t = 720$ vs. $t = 850$), suggesting difficulty-adaptive acquisition allocates resources more efficiently.

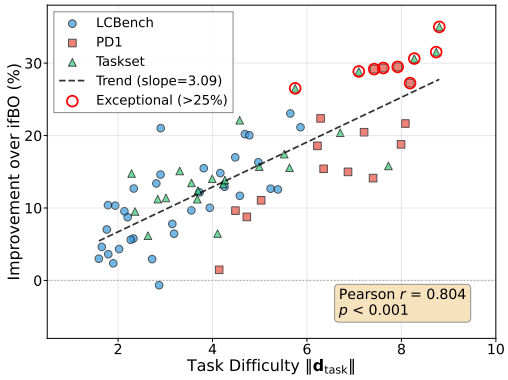


Figure 5. Per-task regret improvement vs. task difficulty. DABO gains more on complex tasks.

Table 2. Final normalized regret at $B = 1000$. Mean $\pm$ std. error over 10 seeds. Bold indicates best freeze-thaw method.

Method	LCBench	PD1	Taskset
Random Search	0.082 ± 0.006	0.124 ± 0.011	0.156 ± 0.014
Hyperband	0.038 ± 0.004	0.071 ± 0.008	0.089 ± 0.009
ASHA	0.034 ± 0.003	0.065 ± 0.007	0.081 ± 0.008
Freeze-Thaw GP	0.029 ± 0.003	0.058 ± 0.006	0.073 ± 0.007
DyHPO	0.021 ± 0.002	0.039 ± 0.004	0.052 ± 0.005
DPL	0.018 ± 0.002	0.042 ± 0.004	0.049 ± 0.005
ifBO	0.016 ± 0.002	0.034 ± 0.003	0.044 ± 0.004
DABO	$\mathbf{0.014 \pm 0.001}$	$\mathbf{0.028 \pm 0.003}$	$\mathbf{0.039 \pm 0.003}$

Figure 5 shows per-task regret improvements over ifBO. We observe consistent gains across most tasks, with particularly strong improvements (over 20%) on 12 out of 75 tasks. These correspond to complex tasks with high $\|\mathbf{d}_{\text{task}}\|$ such as PD1's ImageNet-ResNet and Taskset's 8-hyperparameter transformer tasks, where difficulty-adaptive allocation provides substantial benefits. On simple tasks with low difficulty, both methods perform similarly as the optimization landscape is relatively smooth. The positive correlation between task difficulty and performance gain (Pearson $r = 0.804$, $p < 0.001$) validates that difficulty awareness is most valuable where it matters most.

4.4. Ablation Studies

We ablate three core components: diffusion-based data generation (Diff), difficulty conditioning in DA-PFN (DA), and difficulty-adaptive acquisition (Acq). Table 3 shows results on LCBench. Using diffusion alone improves regret by 5% over ifBO, validating that learning priors from real data outperforms basis functions. Difficulty condition-

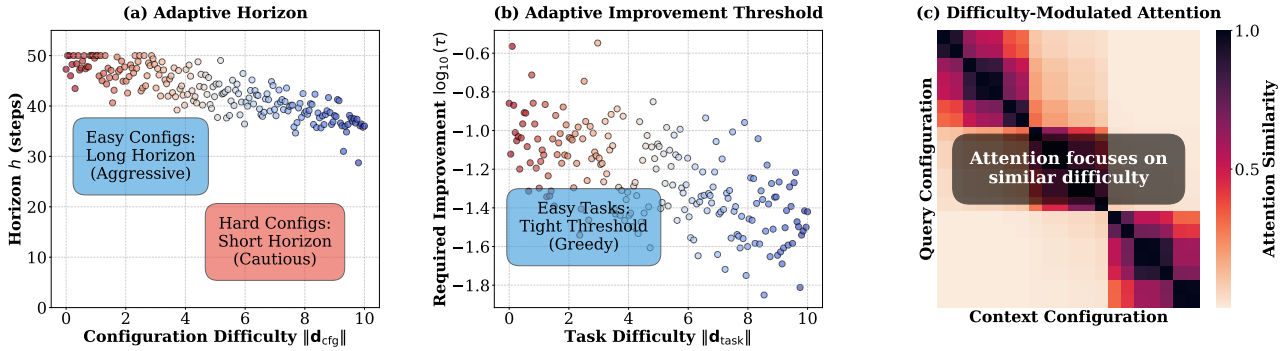


Figure 6. (a) Horizon adaptation. (b) Threshold adaptation. (c) Attention modulation by difficulty similarity.

Table 3. Ablation study on LCBench. Final normalized regret at $B = 1000$.

Diff	DA	Acq	Regret↓
			0.016 ± 0.002 (ifBO)
✓			0.0152 ± 0.002
	✓		0.0148 ± 0.002
		✓	0.0154 ± 0.002
✓	✓		0.0142 ± 0.001
✓		✓	0.0145 ± 0.002
	✓	✓	0.0144 ± 0.002
✓	✓	✓	$\mathbf{0.0140 \pm 0.001}$

ing alone yields 7.5% improvement, showing the value of difficulty-aware representations even with ifBO's synthetic data. Adaptive acquisition alone gives 3.75% gain, indicating modest benefits from allocation strategy. Combining Diff + DA achieves 11.25% improvement with synergistic effects exceeding individual contributions, while the full model reaches 12.5% improvement. This suggests difficulty conditioning is the most impactful component, with diffusion-based generation providing complementary gains by exposing the model to realistic curve patterns. Additional ablations on difficulty levels, hyperparameter sensitivity, and extended baseline comparisons are provided in the supplementary materials.

4.5. Visualization and Data Quality Analysis

Figure 6 visualizes how DABO adapts behavior based on difficulty. Panel (a) shows horizon $h(\mathbf{d})$ decreasing exponentially with configuration difficulty—difficult configurations receive shorter horizons for frequent re-evaluation while easy ones get longer horizons for exploitative allocation. Panel (b) shows threshold $T(\mathbf{d})$ relaxing with task difficulty—complex tasks have relaxed thresholds encouraging exploratory tolerance while simple tasks have tight thresholds for aggressive pruning. Panel (c) displays attention weights between configurations: high-similarity pairs (same difficulty level) show elevated attention around 0.6, while dissimilar pairs show low attention around 0.1, validating our difficulty-modulated attention mechanism in Eq. 18.

To validate our diffusion-based generation, we compare synthetic curves from DABO and ifBO against real curves using Fréchet Curve Distance (FCD) and Maximum Mean Discrepancy (MMD). Diffusion-generated curves are 2.3× closer to real data in both metrics (FCD: 1.24 vs. 2.83; MMD: 0.018 vs. 0.042), confirming that our data generation captures realistic HPO patterns more faithfully than parametric priors.

Computational cost analysis shows the one-time offline training consists of two stages: 72 hours on 4 A40 GPUs for the diffusion model (288 GPU-hours) and 96 hours on 8 A40 GPUs for the DA-PFN (768 GPU-hours), totaling 1056 GPU-hours. However, this cost is amortized across all future HPO tasks. For online optimization, DA-PFN adds approximately 130ms overhead per iteration versus FT-PFN, dominated by difficulty computation. Over a 1000-step HPO run, this amounts to roughly 2 minutes extra—negligible compared to the hours or days of model training being optimized. This significant upfront computational investment is justified by the 11-18% HPO performance gains, which translate to finding better models faster.

5. Conclusion

In this work, we identified and addressed a fundamental limitation of modern hyperparameter optimization: its blindness to optimization difficulty. We introduced DABO, a holistic framework that pioneers a difficulty-aware approach by integrating a hierarchical difficulty model, a conditional diffusion model for generating realistic, difficulty-annotated data priors, and a difficulty-aware surrogate network that adapts its strategy. Extensive experiments show that this paradigm shift from configuration-centric to difficulty-aware optimization achieves state-of-the-art performance while enabling more robust, efficient, and intelligent resource allocation in automated machine learning.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *ICCV*, pages 6430–6439, 2019. 2

[2] Steven Adriaensen, Herilalaina Rakotoarison, Samuel Müller, and Frank Hutter. Efficient bayesian learning curve extrapolation using prior-data fitted networks. In *NeurIPS*, pages 19858–19886, 2023. 2

[3] Archit Bansal, Danny Stoll, Maciej Janowski, Arber Zela, and Frank Hutter. Jahs-bench-201: A foundation for research on joint architecture and hyperparameter search. In *NeurIPS*, pages 38788–38802, 2022. 2, 4, 6

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NeurIPS*, 2011. 1

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, pages 8780–8794, 2021. 2

[6] Yanna Ding, Zijie Huang, Xiao Shou, Yihang Guo, Yizhou Sun, and Jianxi Gao. Architecture-aware learning curve extrapolation via graph ordinary differential equation. In *AAAI*, pages 16289–16297, 2025. 2

[7] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, Kevin Leyton-Brown, et al. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In *NeurIPS*, 2013. 2

[8] Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, Rene Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. In *NeurIPS*, 2021. 2, 4, 6

[9] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *ICML*, pages 1437–1446, 2018. 1, 2

[10] Benjamin Feuer, Robin T Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. Tunetables: Context optimization for scalable prior-data fitted networks. In *NeurIPS*, pages 83430–83464, 2024. 2

[11] Luca Franceschi, Michele Donini, Valerio Perrone, Aaron Klein, Cédric Archambeau, Matthias Seeger, Massimiliano Pontil, and Paolo Frasconi. Hyperparameter optimization in machine learning. *Foundations and Trends® in Machine Learning*, 18(6):975–1109, 2025. 1

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, pages 6840–6851, 2020. 2

[13] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *ICLR*, 2023. 2

[14] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2021. 2, 3

[15] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *AISTATS*, pages 240–248, 2016. 2

[16] Jiantong Jiang, Zeyi Wen, Atif Mansoor, and Ajmal Mian. Efficient hyperparameter optimization with adaptive fidelity identification. In *CVPR*, pages 26181–26190, 2024. 2

[17] Terry Jones, Stephanie Forrest, et al. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *ICGA*, pages 184–192, 1995. 2

[18] Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. In *NeurIPS*, pages 23928–23941, 2021. 1, 2, 6

[19] Aaron Klein, Louis C Tiao, Thibaut Lienart, Cedric Archambeau, and Matthias Seeger. Model-based asynchronous hyperparameter and neural architecture search. *arXiv preprint arXiv:2003.10865*, 2020. 1

[20] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. 1, 2, 6

[21] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. In *MLSys*, pages 230–246, 2020. 6

[22] Mengyang Li, Xiaoling Zhou, and Ou Wu. Delving into the training dynamics for image classification. *IEEE Transactions on Image Processing*, 34:6783–6798, 2025. 3

[23] Jiesong Liu, Feng Zhang, Jiawei Guan, and Xipeng Shen. Uq-guided hyperparameter optimization for iterative learners. In *NeurIPS*, pages 386–415, 2024. 2

[24] Katherine M Malan and Andries P Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013. 2

[25] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *GECCO*, pages 829–836, 2011. 2

[26] Luke Metz, Niru Maheswaranathan, Ruoxi Sun, C Daniel Freeman, Ben Poole, and Jascha Sohl-Dickstein. Using a thousand optimization tasks to learn hyperparameter search strategies. *arXiv preprint arXiv:2002.11887*, 2020. 6

[27] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *ICLR*, 2022. 2

[28] Thomas Nagler, Lennart Schneider, Bernd Bischl, and Matthias Feurer. Reshuffling resampling splits can improve generalization of hyperparameter optimization. In *NeurIPS*, pages 40486–40533, 2024. 1

[29] Erik Pitzer and Michael Affenzeller. A comprehensive survey on fitness landscape analysis. *Recent Advances in Intelligent Engineering Systems*, pages 161–191, 2012. 2, 3

[30] Herilalaina Rakotoarison, Steven Adriaensen, Neeratyoy Mallik, Samir Garibov, Eddie Bergman, and Frank Hutter. In-context freeze-thaw bayesian optimization for hyperparameter optimization. In *ICML*, pages 41982–42008, 2024. 1, 2, 6

[31] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022. 2

[32] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *ICML*, pages 8857–8868, 2021. 2

[33] Anthony Rasulo, Kate Smith-Miles, Mario Muñoz, Julia Handl, and Manuel López-Ibáñez. Extending instance space analysis to algorithm configuration spaces. In *GECCO*, pages 147–150, 2024. 2

[34] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *NeurIPS*, 2012. 1

[35] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 4

[36] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021. 2

[37] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014. 1, 2, 6

[38] Anna Vettoruzzo, Mohamed-Rafik Bouguelia, Joaquin Vanschoren, Thorsteinn Rögnvaldsson, and KC Santosh. Advances and challenges in meta-learning: A technical review. *IEEE transactions on pattern analysis and machine intelligence*, 46(7):4763–4779, 2024. 2, 3

[39] Zi Wang, George E Dahl, Kevin Swersky, Chansoo Lee, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani. Pre-trained gaussian processes for bayesian optimization. *Journal of Machine Learning Research*, 25(212): 1–83, 2024. 6

[40] Martin Wistuba, Arlind Kadra, and Josif Grabocka. Dynamic and efficient gray-box hyperparameter optimization for deep learning. In *AutoML Workshop*, 2022. 1, 2, 6

[41] Jonghyeon Won, Hyun-Suk Lee, and Jang-Won Lee. A review on multi-fidelity hyperparameter optimization in machine learning. *ICT Express*, 11(2):245–257, 2025. 2

[42] Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-PyTorch: Multi-fidelity MetaLearning for efficient and robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021. 6

[43] Feng Zou, Debao Chen, Hui Liu, Siyu Cao, Xuying Ji, and Yan Zhang. A survey of fitness landscape analysis for optimization. *Neurocomputing*, 503:129–139, 2022. 2