Hyperparameter Optimization via Interacting with Probabilistic Circuits

Jonas Seng*[1] **Fabrizio Ventola***[1] **Zhongjie Yu**[1] **Kristian Kersting**[1,2,3,4]

[1]Computer Science Dept., Technical University of Darmstadt, Germany
[2]hessian.ai
[3]Centre for Cognitive Science TU Darmstadt
[4]German Research Centre for AI (DFKI)

Abstract

Despite the growing interest in designing truly interactive hyperparameter optimization (HPO) methods, only a few allow human feedback to be included. However, these methods add friction to the interactive process, rigidly requiring users to define prior distributions ex ante and often imposing additional constraints on the optimization framework. This hinders flexible incorporation of expertise and valuable knowledge of domain experts, who might provide partial feedback at any time during optimization. To overcome these limitations, we introduce a novel Bayesian optimization approach leveraging probabilistic circuits (PCs) as a surrogate model. PCs encode a tractable joint distribution over the hybrid hyperparameter space and evaluation scores. They enable tractable and exact conditional inference and sampling, allowing users to provide beliefs interactively and generate configurations adhering to their feedback. We demonstrate the benefits of the resulting interactive HPO through an extensive empirical evaluation of diverse benchmarks, including the challenging setting of neural architecture search.

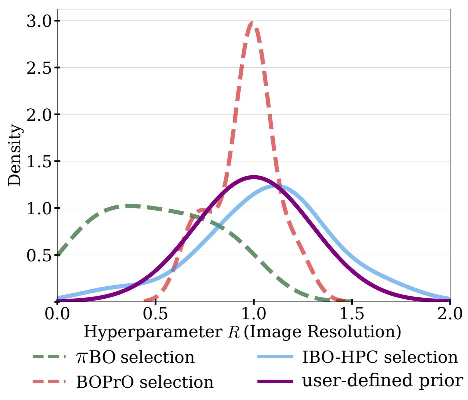
1 INTRODUCTION

Hyperparameters crucially influence the performance of machine learning (ML) algorithms and must be set carefully to fully unleash the algorithm's potential (Bergstra and Bengio, 2012; Hutter et al., 2013; Probst et al., 2019). Hyperparameter optimization (HPO) aims to automatize the tedious and costly manual tuning of hyperparameters (Bischl et al., 2023). Generally, HPO is framed as a black-box optimization with an expensive objective since its functional form is unknown and its evaluation requires the expensive training of ML models. In deep learning, there is also an interest in optimizing the hyperparameters defining the architecture of neural models, known as neural architecture search (NAS).

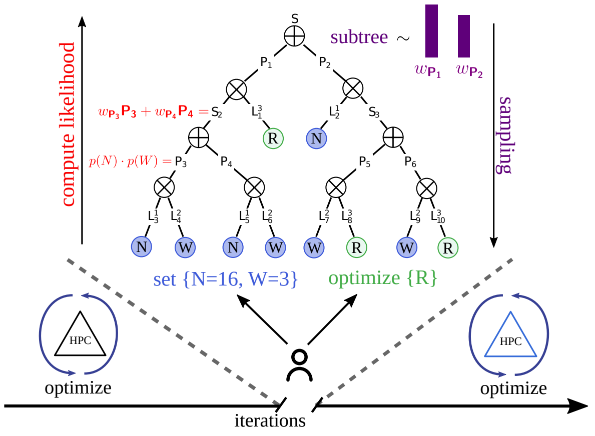
The goal in NAS and HPO is to traverse the search space efficiently to find good configurations, avoiding unpromising regions. Bayesian optimization (BO) methods have proven to be sample efficient and able to converge on good configurations quickly. In each iteration, basing on previously evaluated configurations, BO methods iteratively update a surrogate model to capture the characteristics of the unknown objective. The surrogate is then used by a selection policy to select the next configuration to evaluate, balancing the exploration of the search space with the exploitation of knowledge encoded by the surrogate (Wang et al., 2022; Garnett, 2023). Prominent policies delegate the handling of the exploration-exploitation trade-off to an acquisition function or employ sampling strategies (see App. A.1).

Despite the recent advances in HPO and NAS, hyperparameters are often still tuned manually (Bouthillier and Varoquaux, 2020), and cutting-edge neural architectures, e.g., transformers (Vaswani et al., 2017), are derived by hand. Given that many ML practitioners perform hyperparameter tuning purely based on their knowledge, experience, and intuition, integrating this valuable knowledge to guide HPO algorithms any time *during* optimization is of high value since it can substantially foster the search and mitigate its cost (for example, see Fig. 1(b), App. B & C). To accommodate user priors in BO, Souza et al. (2021) and Hvarfner et al. (2022) integrate weighting schemes that alter the behavior of the acquisition function based on the user-defined priors. While Souza et al. (2021) weight the surrogate's prediction with the prior, Hvarfner et al. (2022) directly weight the acquisition function. Although these approaches are valid and principled ways to guide an HPO task, their weighting schemes limit the set of compatible acquisition functions and selection policies. Furthermore, they require user knowledge to be available *ex ante*, i.e., before the optimization, hindering users to adjust their beliefs flexibly, at *any* iteration of the optimization. Moreover, these approaches might

*indicates equal contribution



(a)



(b)

Figure 1: **Interactive Bayesian Hyperparameter Optimization.** (a) Accurately reflecting user beliefs is crucial for interactive HPO to fully leverage human-provided knowledge. IBO-HPC (our method) precisely reflects the user prior provided over the hyperparameter R (image resolution), while πBO and BOPrO fail to do so. (b) In this example, users can guide IBO-HPC by providing knowledge about hyperparameters (here depth multiplier N and width multiplier W of a CNN) simply by conditioning the surrogate model (a probabilistic circuit) on their beliefs. This allows IBO-HPC to focus the optimization on resolution R, thus, improving convergence and the quality of the solutions.

not reflect user knowledge precisely, as defined in the priors, due to the nonlinear integration of the priors in the acquisition function. For example, in Fig. 1(a), the values selected for a hyperparameter (R) by BOPrO (Souza et al., 2021) and πBO (Hvarfner et al., 2022) during the first 20 iterations substantially deviate from the given user priors.

To overcome the above limitations we introduce INTERACTIVE BAYESIAN OPTIMIZATION VIA HYPERPARAMETER PROBABILISTIC CIRCUITS (IBO-HPC). Our novel BO method provides an elegant and flexible mechanism to incorporate user knowledge *any time* during optimization by directly *conditioning* the surrogate on user beliefs. This enables our selection policy to properly reflect the feedback provided in form of points or distributions over hyperparameter values. For our aim, as surrogate model, we exploit probabilistic circuits (PCs) (Choi et al., 2020) that, in contrast to common choices like Gaussian processes (GPs) (Rasmussen and Williams, 2006) and random forests (RFs) (Breiman, 2001), provide tractable flexible inference and (conditional) sampling. We refer to these PCs learned over hyperparameters and evaluations as *hyperparameter probabilistic circuits* (HPC). Furthermore, to be robust against misleading user input, we integrate a decay mechanism that decreases the influence of user knowledge over time. We show the benefits of our interactive approach through an extensive empirical evaluation including both HPO and NAS benchmarks.

2 RELATED WORK

BO methods leverage a surrogate model that aims to encode the behavior of the unknown objective function. Then, a selection policy queries the surrogate to choose only promising configurations to be evaluated next (Hutter et al., 2011; Snoek et al., 2012; Shahriari et al., 2016). Common choices for surrogate models are GPs (Rasmussen and Williams, 2006) or RFs (Breiman, 2001). Unfortunately, these models can answer only to a restricted number of queries limiting the integration of user knowledge. In NAS, to tame the high dimensionality, several alternatives to BO have been proposed (Zoph and Le, 2017; Pham et al., 2018; Liu et al., 2019; Den Ottelander et al., 2021). However, these methods do not allow to easily integrate human feedback.

While several approaches have been proposed to transfer knowledge between instances of HPO and NAS tasks (Yogatama and Mann, 2014; Wistuba et al., 2015; Perrone et al., 2018; Salinas et al., 2020; Horváth et al., 2021), incorporating user feedback has received little attention. Recent methods extend standard BO methods to allow users to incorporate feedback as prior beliefs Ramachandran et al. (2020); Souza et al. (2021); Hvarfner et al. (2022). However, they need an *ex ante* full specification of the priors and often impose additional constraints such as requiring invertible priors (Ramachandran et al., 2020) or a specific acquisition function (Souza et al., 2021). Furthermore, these methods struggle in properly reflecting the provided user knowledge. We believe that truly interactive HPO methods should not be limited to the ex ante full specification of user knowledge and should properly reflect the provided feedback.

3 INTERACTIVE HYPERPARAMETER OPTIMIZATION

Before introducing IBO-HPC, our flexible and truly interactive BO method, we introduce a formal definition of interactive selection policy. We start by recalling a formal definition of HPO (Kohavi and John, 1995).

Definition 1 (Hyperparameter optimization)**.** *Given hyperparameters $\boldsymbol{\mathcal{H}} = \{H_1, \ldots, H_n\}$ with associated domains $\mathbf{H}_1, \ldots, \mathbf{H}_n$, and a set of problem instances $\boldsymbol{\mathcal{X}}$, we define a search space $\boldsymbol{\Theta} = \mathbf{H}_1 \times \cdots \times \mathbf{H}_n$. For a given problem instance $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ and evaluation function $f : \boldsymbol{\Theta} \times \boldsymbol{\mathcal{X}} \to \mathbb{R}$, hyperparameter optimization aims to solve $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} f(\boldsymbol{\theta}; \mathbf{x})$.*

An interactive BO method should be capable of incorporating, at any time, the knowledge provided by users. Moreover, the selection policy should reflect the provided knowledge as specified by the user. Thus, we formalize the concept of an interactive selection policy (interactive policy in short) that adheres to these requirements.

Definition 2 (Interactive Policy)**.** *An interactive policy is a function $p : \boldsymbol{\mathcal{S}} \times \boldsymbol{\Theta} \times \boldsymbol{\mathcal{K}} \to \boldsymbol{\mathcal{P}}(\boldsymbol{\Theta})$ mapping from the set of surrogates $\boldsymbol{\mathcal{S}}$ and search space $\boldsymbol{\Theta}$ to the set of all distributions $\boldsymbol{\mathcal{P}}(\boldsymbol{\Theta})$ over the search space $\boldsymbol{\Theta}$ while able to incorporate user knowledge $\mathcal{K} \in \boldsymbol{\mathcal{K}}$.*

Def. 2 can be followed also by ignoring user knowledge. Thus, we introduce *feedback adhering interactive policies* that make sure that (i) user knowledge affects the policy's outcome and (ii) user knowledge is reflected in the interactive policy as specified.

Definition 3 (Feedback Adhering Interactive Policy)**.** *Given user knowledge $\mathcal{K} \in \boldsymbol{\mathcal{K}}$ and surrogate $s_t \in \boldsymbol{\mathcal{S}}$ at iteration t, an interactive policy p is effective if $p(\boldsymbol{\Theta}, s_t, \mathcal{K}) \neq p(\boldsymbol{\Theta}, s_t, \emptyset)$ where $\emptyset$ indicates that there is no user knowledge available for p. If further knowledge is provided as a distribution $q(\hat{\boldsymbol{\mathcal{H}}})$ over $\hat{\boldsymbol{\mathcal{H}}} \subset \boldsymbol{\mathcal{H}}$, we call p feedback adhering if it is effective and $\int_{\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}}} p(\boldsymbol{\Theta}, s_t, \mathcal{K}) = q(\hat{\boldsymbol{\mathcal{H}}})$ holds.*

Note that Def. 3 does not require user knowledge to have exclusively positive effects. Equipped with Def. 2 and 3 we now introduce IBO-HPC adhering to both definitions.

3.1 INTERACTIVE BAYESIAN OPTIMIZATION WITH HYPERPARAMETER PROBABILISTIC CIRCUITS

We now present IBO-HPC (see App. A.2 for pseudocode). Since our surrogate is a density estimator, we start off by sampling J hyperparameter configurations from a prior distribution $p(\boldsymbol{\mathcal{H}})$ and evaluate them via the objective function f. After evaluating each sampled configuration $\boldsymbol{\theta}$ we obtain a set $\boldsymbol{\mathcal{D}}$ of pairs $(\boldsymbol{\theta}, f_{\boldsymbol{\theta}}(\mathbf{x}))$ and fit a HPC s estimating the joint distribution $p(\boldsymbol{\mathcal{H}}, F)$, where $\boldsymbol{\mathcal{H}}$ is the set of hyperparameters and F is a random variable representing the evaluation score. Then, IBO-HPC runs a feedback adhering interactive policy to select a new configuration $\boldsymbol{\theta}$. We target the configurations that improve upon the current incumbent. Thus, a posterior distribution over the hyperparameter space is derived by conditioning on the best score observed so far $f^* = \max_f \boldsymbol{\mathcal{D}}$ and (optional) user knowledge $\mathcal{K}$. For now, $\mathcal{K}$ is assumed to be given in the form of conditions such as $\hat{\boldsymbol{\mathcal{H}}} = \hat{\mathbf{h}}$ where $\hat{\boldsymbol{\mathcal{H}}} \subset \boldsymbol{\mathcal{H}}$ is set to $\hat{\mathbf{h}}$. Then, a new configuration is sampled from the posterior:

$$\begin{aligned} &p(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}}, F = f^*) = s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}}, F = f^*), \\ &\boldsymbol{\theta} \sim p(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}}, F = f^*). \end{aligned} \tag{1}$$

Since users might be uncertain about hyperparameter values, defining a distribution $q(\hat{\boldsymbol{\mathcal{H}}})$ over $\hat{\boldsymbol{\mathcal{H}}}$ can be more reasonable than setting a fixed value. Interpreting $q(\hat{\boldsymbol{\mathcal{H}}})$ as a distribution over conditions $\hat{\boldsymbol{\mathcal{H}}} = \hat{\mathbf{h}}$ where $\hat{\mathbf{h}} \sim q(\hat{\boldsymbol{\mathcal{H}}})$ induces a different distribution than in Eq. 1:

$$p(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}}, F = f^*) \cdot q(\hat{\boldsymbol{\mathcal{H}}}) = s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}}, F = f^*) \cdot q(\hat{\boldsymbol{\mathcal{H}}}). \tag{2}$$

To allow IBO-HPC to recover from misleading user interactions, $\mathcal{K}$ is only used with probability ρ in each iteration where ρ is decreased with a decay factor γ over time. When user knowledge is provided at iteration T, the distribution over configurations after $T + t$ iterations becomes:

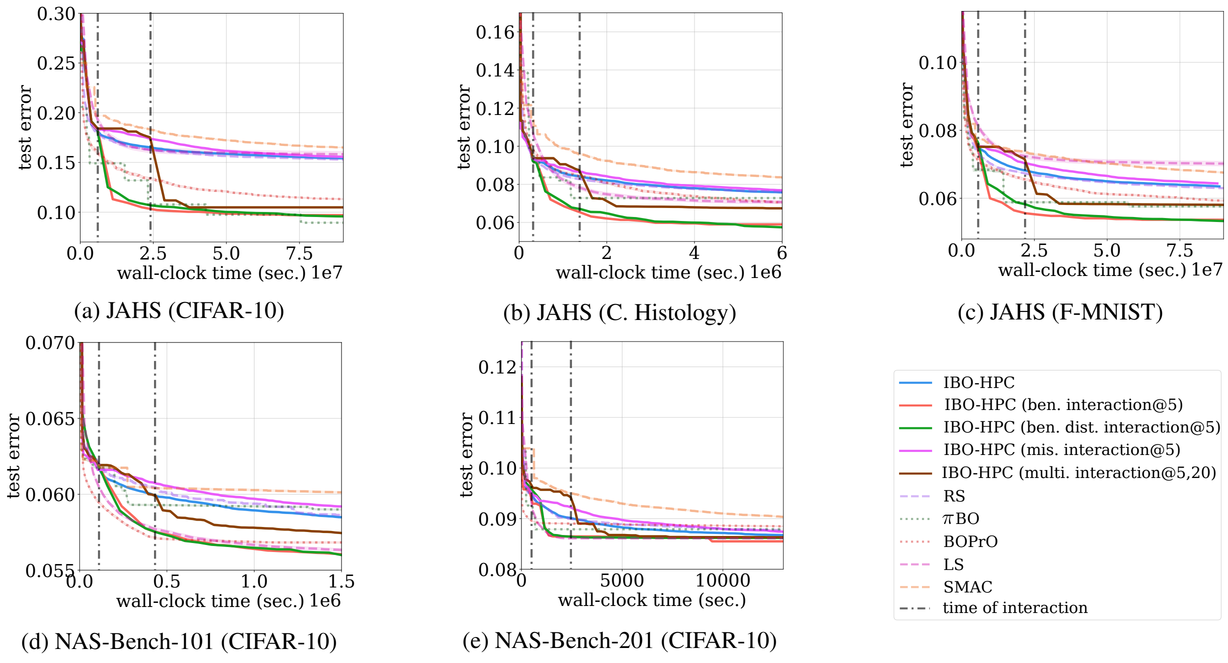
$$\gamma^t \rho \cdot s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}}, F = f^*) \cdot q(\hat{\boldsymbol{\mathcal{H}}}) + (1 - \gamma^t \rho) \cdot s(\boldsymbol{\mathcal{H}} | F = f^*). \tag{3}$$

Representing the distributions in Eq. 2 and 3 as an HPC (see App. D) is not trivial since the prior q is defined over an arbitrary subset and no further assumptions about q are made. Thus we approximate Eq. 2 by sampling N times from $q(\hat{\boldsymbol{\mathcal{H}}})$ and use Eq. 1 to obtain N conditional distributions following the user prior $q(\hat{\boldsymbol{\mathcal{H}}})$. To select a promising configuration, for each conditional, IBO-HPC samples B configurations and chooses the one maximizing the likelihood $s(\boldsymbol{\mathcal{H}} | F = f^*)$. This results in N configurations from which one is sampled uniformly for the next evaluation. To foster exploration, by leveraging uncertainty encoded in the (conditional) distribution of the surrogate, we only retrain the surrogate every L iterations. An iteration is concluded by updating the set of evaluations $\boldsymbol{\mathcal{D}}$. The algorithm runs until convergence or when another condition for termination is met.

We now show that IBO-HPC's policy to select the next configuration is a feedback adhering interactive policy according to Def. 3. Since modeling dependencies among hyperparameters is not trivial for users, we assume that users provide as beliefs a fully (naively) factorized distribution.

Proposition 1 (IBO-HPC Policy is feedback adhering interactive)**.** *Given a search space $\boldsymbol{\Theta}$ over hyperparameters $\boldsymbol{\mathcal{H}}$, an HPC $s \in \boldsymbol{\mathcal{S}}$, user knowledge $\mathcal{K} \in \boldsymbol{\mathcal{K}}$ in form of a prior q over $\hat{\boldsymbol{\mathcal{H}}} \subset \boldsymbol{\mathcal{H}}$ s.t. $\int_{\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}}} s(\boldsymbol{\mathcal{H}} | F = f^*) \neq q(\hat{\boldsymbol{\mathcal{H}}})$, the selection policy of IBO-HPC is feedback adhering interactive.*

The proof of Prop. 1 is given in App. E. We now follow with the empirical evaluation of IBO-HPC.



(a) JAHS (CIFAR-10) (b) JAHS (C. Histology) (c) JAHS (F-MNIST)

(d) NAS-Bench-101 (CIFAR-10) (e) NAS-Bench-201 (CIFAR-10)

Figure 2: **IBO-HPC is truly interactive and effective on HPO and NAS tasks.** For 5 tasks across three challenging benchmarks, IBO-HPC is competitive with state-of-the-art methods when no user knowledge is provided and outperforms all competitors when the same beneficial (ben.) user feedback is provided to IBO-HPC and to the competitors. Furthermore, IBO-HPC recovers from misleading (mis.) user beliefs and allows users to adjust their beliefs at any time (multi.).

4 EXPERIMENTAL EVALUATION

We evaluate the performance of IBO-HPC on different HPO and NAS benchmarks under several conditions: fully automatic (i.e., no user feedback), interactive with beneficial user feedback, and with multiple user feedback (misleading first, then beneficial). We provided user knowledge either as a distribution or by setting the value for a subset of hyperparameters. The surrogates are based on PCs for hybrid domains (Molina et al., 2018). In each scenario, we compare IBO-HPC against five diverse competitors: random search (RS) (Bergstra et al., 2011), local search (LS) (White et al., 2020), SMAC (Hutter et al., 2011) as standard HPO methods and BOPrO (Souza et al., 2021) with πBO (Hvarfner et al., 2022) which allow ex ante incorporation of user knowledge. We evaluate IBO-HPC on five real-world image classification tasks from three popular benchmarks: CIFAR-10 from NAS-Bench-101 (Ying et al., 2019), NAS-Bench-201 (Dong and Yang, 2020), and JAHS (Bansal et al., 2022), plus Fashion-MNIST and Colorectal Histology from JAHS. In our evaluation, we optimize w.r.t the validation accuracy and run all algorithms with 500 seeds for $2k$ iterations. We report the mean test error over the wall-clock time with the corresponding standard error. See App. F for details.

Fig. 2 shows that the performance of IBO-HPC without user interaction (blue) is competitive or superior to SMAC on most tasks. While in NAS, due to the discrete nature of the search space, LS performs slightly better than IBO-HPC and thus confirms to be a strong baseline for NAS (Den Ottelander et al., 2021). IBO-HPC outperforms SMAC and LS in the more complex and realistic case of joint architecture and hyperparameter search spaces (JAHS). When IBO-HPC is provided with beneficial user feedback after 5 iterations (green, orange), besides outperforming all standard HPO methods (as expected), it also outperforms πBO and BOPrO on 4/5 tasks, although both competitors obtain user knowledge ex ante. This confirms that IBO-HPC reflects and leverages user beliefs more effectively. Finally, Fig. 2 shows also that IBO-HPC successfully recovers when misleading user beliefs are provided (pink) and that IBO-HPC allows users to adjust their beliefs at any time (brown). Additional experiments and ablation studies are in App. F.6.

5 CONCLUSION

We introduced a novel interactive BO method named IBO-HPC that leverages the flexible inference of probabilistic circuits to easily incorporate user feedback at any time during optimization. IBO-HPC is competitive with state-of-the-art methods when no user knowledge is provided and it outperforms competitors when available. Moreover, IBO-HPC reliably recovers from misleading user beliefs and converges remarkably faster when provided with valuable user knowledge, thus, saving resources. Currently, to leverage new evaluations, IBO-HPC needs to retrain the surrogate from scratch. Thus, we envision the exploration of continual learning techniques to avoid the frequent retraining, increasing efficiency, and the reuse of knowledge over different tasks.

Acknowledgements

This work was supported by the National High-Performance Computing Project for Computational Engineering Sciences (NHR4CES) and the Federal Ministry of Education and Research (BMBF) Competence Center for AI and Labour ("KompAKI", FKZ 02L19C150). Furthermore, this work benefited from the cluster project "The Third Wave of AI".

REFERENCES

Bansal, A., Stoll, D., Janowski, M., Zela, A., and Hutter, F. (2022). Jahs-bench-201: A foundation for research on joint architecture and hyperparameter search. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NIPS)*.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2).

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A., Deng, D., and Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2).

Bouthillier, X. and Varoquaux, G. (2020). Survey of machine-learning experimental methods at NeurIPS2019 and ICLR2020. Research report, Inria Saclay Ile de France.

Breiman, L. (2001). Random forests. 45(1):5–32.

Choi, Y., Vergari, A., and Van den Broeck, G. (2020). Probabilistic circuits: A unifying framework for tractable probabilistic models. Technical report, UCLA.

Den Ottelander, T., Dushatskiy, A., Virgolin, M., and Bosman, P. A. N. (2021). Local search is a remarkably strong baseline for neural architecture search. In *Evolutionary Multi-Criterion Optimization: 11th International Conference*.

Dong, X. and Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*.

Garnett, R. (2023). *Bayesian Optimization*. Cambridge University Press.

Horváth, S., Klein, A., Richtarik, P., and Archambeau, C. (2021). Hyperparameter transfer learning with adaptive complexity. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1378–1386.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference*.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2013). Identifying key algorithm parameters and instance features using forward selection. In *Learning and Intelligent Optimization: 7th International Conference*.

Hvarfner, C., Stoll, D., Souza, A., Lindauer, M., Hutter, F., and Nardi, L. (2022). πbo: Augmenting acquisition functions with user beliefs for bayesian optimization. In *International Conference on Learning Representations (ICLR)*.

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

Kohavi, R. and John, G. H. (1995). Automatic parameter selection by minimizing estimated error. In *International Conference on Machine Learning (ICML)*.

Liu, H., Simonyan, K., and Yang, Y. (2019). Darts: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*.

Molina, A., Vergari, A., Mauro, N. D., Natarajan, S., Esposito, F., and Kersting, K. (2018). Mixed sum-product networks: A deep architecture for hybrid domains. In *AAAI Conference on Artificial Intelligence*.

Peharz, R., Tschiatschek, S., Pernkopf, F., and Domingos, P. M. (2015). On theoretical properties of sum-product networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Perrone, V., Jenatton, R., Seeger, M. W., and Archambeau, C. (2018). Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018). Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning (ICML)*.

Probst, P., Boulesteix, A.-L., and Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965.

Ramachandran, A., Gupta, S., Rana, S., Li, C., and Venkatesh, S. (2020). Incorporating expert prior in bayesian optimisation via space warping. *Knowledge-Based Systems*, 195:105663.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Salinas, D., Shen, H., and Perrone, V. (2020). A quantile-based approach for hyperparameter transfer learning. In *International Conference on Machine Learning (ICML)*, pages 8438–8448.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*.

Souza, A. L. F., Nardi, L., Oliveira, L. B., Olukotun, K., Lindauer, M., and Hutter, F. (2021). Bayesian optimization with a prior for the optimum. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*.

Wang, X., Jin, Y., Schmitt, S., and Olhofer, M. (2022). Recent advances in bayesian optimization.

White, C., Nolen, S., and Savani, Y. (2020). Exploring the loss landscape in neural architecture search. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015). Sequential model-free hyperparameter tuning. In *2015 IEEE International Conference on Data Mining*, pages 1033–1038.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., and Hutter, F. (2019). Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning (ICML)*.

Yogatama, D. and Mann, G. S. (2014). Efficient transfer learning method for automatic hyperparameter tuning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Hyperparameter Optimization via Interacting with Probabilistic Circuits (Supplementary Material)

Jonas Seng*[1] **Fabrizio Ventola***[1] **Zhongjie Yu**[1] **Kristian Kersting**[1,2,3,4]

[1]Computer Science Dept., Technical University of Darmstadt, Germany
[2]hessian.ai
[3]Centre for Cognitive Science TU Darmstadt
[4]German Research Centre for AI (DFKI)

A METHOD

In this section of the supplementary material we provide an overview on Bayesian optimization in A.1 and the pseudocode of our method in A.2.

A.1 BAYESIAN OPTIMIZATION

Bayesian optimization (BO) aims to optimize a black-box objective function $f : \boldsymbol{\Theta} \rightarrow \mathbb{R}$ which is costly to evaluate, i.e., to find the input $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} f(\boldsymbol{\theta})$ (Shahriari et al., 2016). BO typically tackles such problem in sequential steps, leveraging two key ingredients: a probabilistic surrogate model and a selection policy determining the next $\boldsymbol{\theta}'$ to be evaluated. Given a set $\boldsymbol{\mathcal{D}}_n$ of observations that correspond to the configurations with associated evaluations $(\boldsymbol{\theta}_j, f(\boldsymbol{\theta}_j))_{j=1...n}$, the surrogate $s \in \boldsymbol{\mathcal{S}}$ aims to induce $p(f|\boldsymbol{\mathcal{D}}_n)$. The selection policy uses s to select the next $\boldsymbol{\theta}' \in \boldsymbol{\Theta}$ s.t. it achieves a good exploration-exploitation trade-off. Prominent selection policies optimize an acquisition function $a : \boldsymbol{\Theta} \times \mathcal{S} \rightarrow \mathbb{R}$ (e.g. expected improvement (Jones et al., 1998)) that assigns a utility score of each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ under a surrogate $s \in \boldsymbol{\mathcal{S}}$, or. apply Thompson sampling (Wang et al., 2022). The obtained tuple $(\boldsymbol{\theta}', f(\boldsymbol{\theta}'))$ is added to $\boldsymbol{\mathcal{D}}_n$ and used to update the surrogate model for the next iteration. This process is repeated until a given budget is exhausted or convergence.

Accepted for the 7th Workshop on Tractable Probabilistic Modeling at UAI (TPM 2024).

A.2 IBO-HPC PSEUDOCODE

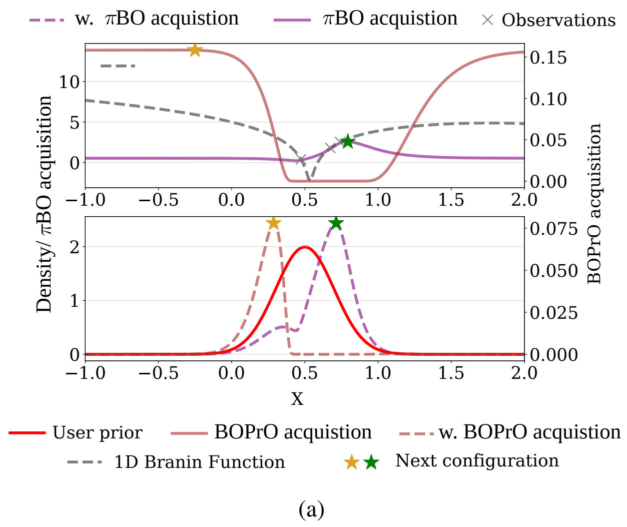
Algorithm 1: Interactive BO with HPCs (IBO-HPC). Our interactive BO method allows for flexible incorporation of user knowledge at any iteration via conditional sampling enabling true interaction with users.

```
Data: Search space $\boldsymbol{\Theta}$ over $\boldsymbol{\mathcal{H}} = \{H_1, \dots, H_n\}$, problem instance $\mathbf{x} \in \mathcal{X}$, prior distribution $p(\boldsymbol{\Theta})$, objective
      $f : \boldsymbol{\Theta} \times \boldsymbol{\mathcal{X}} \to \mathbb{R}$, user prior $q(\hat{\boldsymbol{\mathcal{H}}})$ is optional and can be provided at any time, decay $\gamma$
1  $\boldsymbol{\mathcal{D}} = \emptyset$;
2  for $i \in \{1, \dots, J\}$ do
3      $\boldsymbol{\theta} \sim p(\boldsymbol{\Theta})$;
4      $\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{D}} \cup \{(\boldsymbol{\theta}, f(\boldsymbol{\theta}, \mathbf{x}))\}$;
5  while not converged do
6      every $L$-th iteration, fit HPC $s$ on $\boldsymbol{\mathcal{D}}$;
7      $f^* = \max_f \boldsymbol{\mathcal{D}}$;
8      $b \sim \text{Ber}(\rho)$;
9      if prior $q(\hat{\boldsymbol{\mathcal{H}}})$ given and $b = 1$ then
10         sample $N$ conditions $\mathbf{h} \sim q(\hat{\boldsymbol{\mathcal{H}}})$;
11         $\mathbf{C} = \emptyset$;
12         for condition $\mathbf{h}_i$ in $\mathbf{h}$ do
13             sample $B$ configurations $\boldsymbol{\theta}'_{1,\dots,B} \sim s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}}, f^*)$;
14             $\boldsymbol{\theta}_i^* = \arg\max_{\boldsymbol{\theta}' \in \boldsymbol{\theta}'_{1,\dots,B}} s(\boldsymbol{\theta}' | f^*)$;
15             $\mathbf{C} = \mathbf{C} \cup \boldsymbol{\theta}_i^*$;
16         $\boldsymbol{\theta}^* \sim \mathcal{U}(\mathbf{C})$;
17     else
18         $\boldsymbol{\theta}^* \sim s(\boldsymbol{\mathcal{H}} | f^*)$;
19     $\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{D}} \cup (\boldsymbol{\theta}', f(\boldsymbol{\theta}', \mathbf{x}))$;
20     $\rho = \gamma \cdot \rho$;
21     present evaluations $\boldsymbol{\mathcal{D}}$;
```

B MOTIVATION & REAL-WORLD EXAMPLE

Reflecting user knowledge accurately is crucial for interactive HPO methods to fully benefit from human knowledge and improve trustworthiness. Existing weighting scheme based methods like πBO and BOPrO fail to reflect user priors accurately in their selection policy as it can be seen in Fig. 3(a). Here, we show a 1d-example of a Branin function with an optimum around $x = 0.5$. The user prior (in red) is placed at $x = 0.3$. Both πBO and BOPrO fail to select the next configuration from the high-density region of the prior. Thus, the user prior is not incorporated in the selection process as a user would expect. Our method, IBO-HPC, solves this issue which we now demonstrate based on a real-world example.

To this end, we ran πBO, BOPrO – both leveraging a weighting scheme to incorporate user priors –, and IBO-HPC on the CIFAR-10 task of the JAHS benchmark (Bansal et al., 2022). We specified a prior distribution (Fig. 3(b), purple) over the hyperparameter RESOLUTION (R) that controls the down-/up-sampling rate of an image fed into a neural network. The rest of the hyperparameters for this specific task (i.e. the network architecture and all other hyperparameters; see App. D for details) were optimized by πBO, BOPrO and IBO-HPC without any user knowledge. All methods received the same user prior (πBO and BOPrO from the beginning of the optimization; IBO-HPC after 5 iterations). From the iteration the user prior was provided on, we then considered the values chosen for RESOLUTION by πBO, BOPrO and IBO-HPC for the next 20 iterations and estimated a density of selected values for R (see Fig. 3(b)). We obtained that neither the choices for R by πBO (green dashed line), nor the choices of BOPrO (red dashed line) reflect the user prior as specified. While πBO's choices of RESOLUTION are biased towards smaller values, BOPrO does not reflect the user's uncertainty well in its choices of RESOLUTION. In contrast, IBO-HPC (blue solid line) precisely reflects the user prior as specified (up to random variations due to sampling).



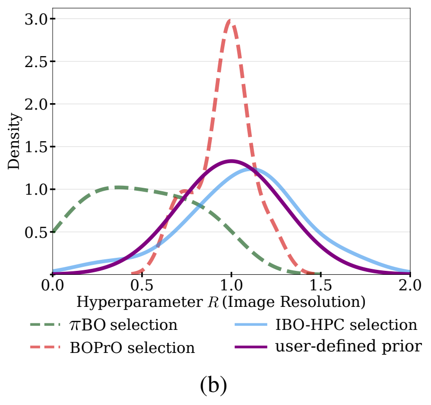


Figure 3: **IBO-HPC reflects user priors as specified.** In contrast to other weighting scheme based methods like πBO and BOPrO, IBO-HPC reflects the user prior as specified in its selection policy.

C WORKING EXAMPLE

In the following, we consider a more detailed example of our proposed method from a user perspective. We assume that we optimize only 3 hyperparameters here, width multiplier W, depth multiplier N, and resolution R of a CNN (see the JAHS benchmark (Bansal et al., 2022)).

The optimization starts where each of the hyperparameters gets optimized by our method. At some point, the user interacts with the optimization process and sets W and N to a fixed value (blue in Fig. 4). From then on, the model only optimizes the remaining hyperparameter R (green in Fig. 4), using conditional sampling from the resulting conditional distribution that the HPC represents after the interaction.

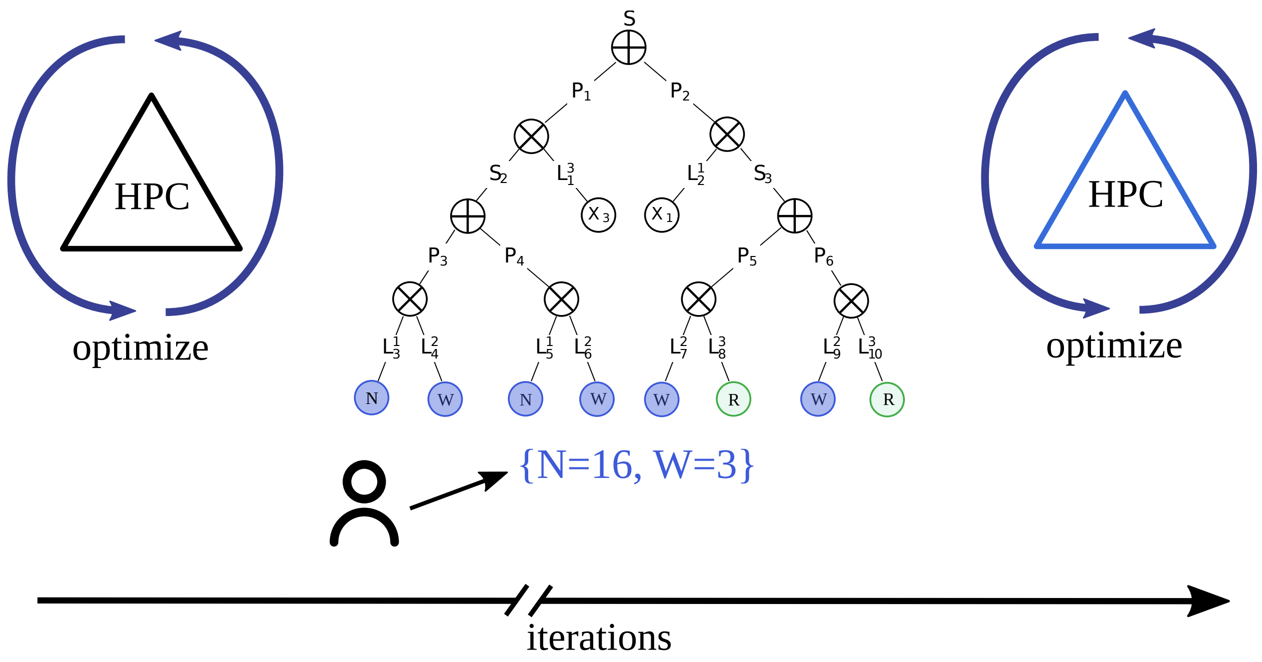


Figure 4: **Example of IBO-HPC.** A user specifies certain aspects of the hyperparameter search space during optimization. Afterward, IBO-HPC takes user knowledge into account when sampling new configuration candidates.

D OUR SURROGATE MODELS: PROBABILISTIC CIRCUITS

Since probabilistic circuits (PCs) are a key component of our method, we provide more details on these models in the following. Let us first start with a rigorous definition of PCs.

Definition 4. *A probabilistic cricuit (PC) is a computational graph encoding a distribution over a set of random variables* $\mathbf{X}$*. It is defined as a tuple* $(\mathcal{G}, \phi)$ *where* $\mathcal{G} = (V, E)$ *is a rooted, directed acyclic graph and* $\phi : V \rightarrow 2^{\mathbf{X}}$ *is the scope function assigning a subset of random variables to each node in* $\mathcal{G}$*. For each internal node* N *of* $\mathcal{G}$*, the scope is defined as the union of scopes of its children, i.e.* $\phi(\mathsf{N}) = \cup_{\mathsf{N}' \in \mathrm{ch}(\mathsf{N})}$*. Each leaf node* L *computes a distribution/density over its scope* $\phi(\mathsf{L})$*. All internal nodes of* $\mathcal{G}$ *are either a sum node* S *or a product node* P *where each sum node computes a convex combination of its children, i.e.,* $\mathsf{S} = \sum_{\mathsf{N} \in \mathrm{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{N}} \mathsf{N}$*, and each product computes a product of its children, i.e.,* $\mathsf{P} = \prod_{\mathsf{N} \in \mathrm{ch}(\mathsf{P})} \mathsf{N}$.

With this definition at hand, we describe the tractable key operations of PCs relevant to our method in more detail.

Inference. Inference in PCs is a bottom-up procedure. To compute the probability of given evidence $\mathbf{X} = \mathbf{x}$, the densities of the leaf nodes are evaluated first. This yields a density value for each leaf. The leaf densities are then propagated bottom-up by computing all product/sum nodes. Eventually, the root node holds the probability/density of $\mathbf{x}$. Note that typically, multiple leaf nodes correspond to the same random variable. Thus, if the children of a sum node have the same scope, we can interpret sum nodes as mixture models. Conversely, if the children of a product node have *non*-overlapping scopes, a product node can be interpreted as a product distribution of two (independent) random variables. We call these two properties smoothness and decomposability. More formally, *smoothness* means that for each sum node $\mathsf{S} \in V$ it holds that $\phi(\mathsf{N}) = \phi(\mathsf{N}')$ for $\mathsf{N}, \mathsf{N}' \in \mathrm{ch}(\mathsf{S})$. *Decomposability* means that for each product node $\mathsf{P} \in V$ it holds that $\phi(\mathsf{N}) \cap \phi(\mathsf{N}') = \emptyset$ for $\mathsf{N}, \mathsf{N}' \in \mathrm{ch}(\mathsf{P})$, $\mathsf{N} \neq \mathsf{N}'$. Hence, PCs can be interpreted as hierarchical mixture models.

Marginalization. Decomposability implies that marginalization is tractable in PCs and can be done in linear time of the circuit size. This is because integrals that can be rewritten by nesting single-dimensional integrals can be computed only in terms of leaf integrals, which are assumed to be tractable as they follow certain distributions (e.g., Gaussian). Computing such nested integrals only in terms of leaf integrals is possible because single-dimensional integrals commute with the sum operation and affect only a single child of product nodes. For more details on the computational implications of decomposability, refer to (Peharz et al., 2015).

Practically, there are two ways to marginalize certain variables from the scope of a PC. One approach is structure-preserving, and marginalization is achieved by setting all leaves corresponding to the set of random variables that are supposed to be marginalized to 1. The second approach constructs a new PC representing the marginal distribution, i.e. the structure of the PC is changed. The second approach is beneficial if samples should be drawn from the marginalized PC because the sampling procedure remains the same, i.e. the PC is adopted to obtain the marginal distribution, not vice versa.

Conditioning. Computing a conditional distribution $p(\mathbf{X}_1|\mathbf{X}_2) = \frac{p(\mathbf{X})}{\int_{\mathbf{X}_2} p(\mathbf{X})}$ where $\mathbf{X}_1 \cup \mathbf{X}_2 = \mathbf{X}$ and $\mathbf{X}_1 \cap \mathbf{X}_2 = \emptyset$ is achieved by combining marginalization (denominator) and inference (numerator). Since inference is tractable for PCs in general and marginalization is tractable for decomposable PCs, conditioning is also tractable.

Sampling. Sampling in PCs is a top-down procedure and recursively samples a sub-tree, starting at the root. Each sum node S holds a parameter vector $\mathbf{w}$ s.t. $\sum_{i=0}^{|\mathrm{ch}(\mathsf{S})|} \mathbf{w}_i = 1$. Based on the distribution induced by $\mathbf{w}$, one of the children of S is sampled as a sub-tree. By decomposability, the scope of the children of a product node is non-overlapping. Thus, sampling from a product node corresponds to sampling from all its child nodes. If a leaf node is reached, a sample is obtained from the distribution at that leaf.

Given the hybrid nature of hyperparameter search spaces, in this work, we focus on a type of PCs tailored for hybrid domains named mixed sum-product networks (MSPNs) (Molina et al., 2018). An MSPN is a decomposable and smooth PC with piecewise polynomial leaves. These properties allow MSPNs to represent valid distributions (Peharz et al., 2015) over hybrid domains (i.e., discrete and continuous variables).

E PROOFS

In this section we provide the proof of Proposition 1 of the main paper.

E.1 IBO-HPC'S POLICY IS FEEDBACK ADHERING INTERACTIVE

Proposition 1 (IBO-HPC Policy is feedback adhering interactive). Given a search space $\mathbf{\Theta}$ over hyperparameters $\boldsymbol{\mathcal{H}}$, an HPC $s \in \boldsymbol{\mathcal{S}}$, user knowledge $\mathcal{K} \in \boldsymbol{\mathcal{K}}$ in form of a prior q over $\hat{\boldsymbol{\mathcal{H}}} \subset \boldsymbol{\mathcal{H}}$ s.t. $\int_{\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}}} s(\boldsymbol{\mathcal{H}} | F = f^*) \neq q(\hat{\boldsymbol{\mathcal{H}}})$, the selection policy of IBO-HPC is feedback adhering interactive.

Proof. We have to show that the policy of IBO-HPC is feedback adhering, i.e. it conforms with Def. 3: The distribution over the configuration space used to obtain new configurations is different if user knowledge is provided from the distribution used if no user knowledge is provided (policy is effective) and the provided user knowledge is represented during configuration selection as specified (feedback adhering).

We first show that the selection policy of IBO-HPC is effective.

IBO-HPC selection policy is effective. Since the decay mechanism allowing IBO-HPC to recover from misleading knowledge can be treated as a constant in each iteration, it is enough if $s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}} = \hat{\mathbf{h}}, F = f^*) \cdot q(\hat{\boldsymbol{\mathcal{H}}} = \hat{\mathbf{h}}) \neq s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}} = \emptyset, F = f^*) \cdot q(\hat{\boldsymbol{\mathcal{H}}} = \emptyset)$ holds for any surrogate s representing a joint distribution over search space $\boldsymbol{\mathcal{H}}$ and prior q over $\hat{\boldsymbol{\mathcal{H}}} \subset \boldsymbol{\mathcal{H}}$ to make the policy effective. Note that we assume that $\mathcal{K}$ is given in form of a prior $q(\hat{\boldsymbol{\mathcal{H}}})$ over $\hat{\boldsymbol{\mathcal{H}}}$ as before. Since $\emptyset \notin \hat{\boldsymbol{\mathcal{H}}}$ is assumed, our policy ignores any prior if no user knowledge is provided. Thus, in this case, the policy samples from the distribution

$$s(\boldsymbol{\mathcal{H}} | F = f^*) = s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}}, F = f^*) \cdot \int_{\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}}} s(\boldsymbol{\mathcal{H}} | F = f^*). \tag{4}$$

Since $s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}}, F = f^*)$ is the same, regardless of whether user knowledge is given or not, user knowledge will lead to a different distribution if $\int_{\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}}} s(\boldsymbol{\mathcal{H}} | F = f^*) \neq q(\hat{\boldsymbol{\mathcal{H}}})$ holds. Since Prop. 1 demands that this is the case, our policy is effective according to Def. 2.

We can now proceed and show feedback adherence of the IBO-HPC selection policy.

IBO-HPC selection policy is feedback adhering. The proof that our policy is feedback adhering directly follows by design: If a user prior $q(\hat{\boldsymbol{\mathcal{H}}})$ is given, Eq. 3 is approximated by sampling N conditions $\mathbf{h}'_{1,\dots,N} \sim q(\hat{\boldsymbol{\mathcal{H}}})$ and computing N conditionals $s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}} = h'_1, F = f^*), \dots, s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}} = h'_N, F = f^*)$. We can approximate $q(\hat{\boldsymbol{\mathcal{H}}})$ arbitrarily close with $N \to \infty$. To select the next configuration, we sample B configurations from each of the N conditionals and select the configuration maximizing $s(\boldsymbol{\mathcal{H}} | F = f^*)$ for each conditional. This leaves us with N candidates. Note that at this point, the hyperparameters $\hat{\boldsymbol{\mathcal{H}}}$ still follow $q(\hat{\boldsymbol{\mathcal{H}}})$ with $N \to \infty$ as the conditions of $s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}} = h'_1, F = f^*), \dots, s(\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}} | \hat{\boldsymbol{\mathcal{H}}} = h'_N, F = f^*)$ remain fixed and only hyperparameters of the set $\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}}$ can vary/are sampled. Thus, maximizing the likelihood $s(\boldsymbol{\mathcal{H}} | F = f^*)$ is only done w.r.t. hyperparameters in $\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}}$. This implies that sampling hyperparameters $\boldsymbol{\mathcal{H}} \setminus \hat{\boldsymbol{\mathcal{H}}}$ can be biased while sampling from $q(\hat{\boldsymbol{\mathcal{H}}})$ is unaffected because the conditions $\mathbf{h}'_{1,\dots,N}$ are sampled first in i.i.d. fashion. Our policy selects the configuration evaluated next by uniformly sampling from the remaining N candidates. Since uniformly sampling L times from a set of N samples from a distribution q results in approximating q arbitrarily close for $N \to \infty$ and $L \to \infty$, we conclude that user priors are exactly reflected as specified in our selection policy. This concludes our proof that the selection policy of IBO-HPC is effective and feedback adhering. □

F EXPERIMENTAL DETAILS

Here we present additional details of our empirical evaluation. We provide our code at `https://github.com/ml-research/ibo-hpc` and provide all logs and data at `https://1drv.ms/u/s!Aty3JfFPZnuutVz0eNifHh6Uhvjz?e=hX2chn`.

F.1 DEFINING USER INTERACTIONS

For the experiments, beneficial and misleading user interactions have been defined as user priors for each benchmark. To define priors, we randomly sampled $10k$ configurations and kept the best/worst performing ones, denoted as $\mathbf{h}^+$ and $\mathbf{h}^-$, respectively. To demonstrate that user priors over a few hyperparameters are enough to improve the performance of IBO-HPC considerably, we defined beneficial interactions by selecting a small subset of hyperparameters $\hat{\boldsymbol{\mathcal{H}}} \subset \boldsymbol{\mathcal{H}}$. Then, we defined a prior over each $H \in \hat{\mathcal{H}}$ s.t. the probability of sampling the value of H given in $\mathbf{h}^+$, denoted by $\mathbf{h}^+[H]$, is 1000 times higher than sampling a different value than $\mathbf{h}^+[H]$. For misleading interaction, $\hat{\mathcal{H}}$ was chosen to be large to demonstrate that IBO-HPC recovers even if a large amount of misleading information is provided. We then defined priors over $\hat{\mathcal{H}}$ as for beneficial interactions; however, this time the probability to sample $\mathbf{h}^-[H]$ is 1000 times higher than for other values for each $H \in \hat{\mathcal{H}}$. The priors were chosen to be rather strong since, as emphasized in Sec. 1, the stronger the prior, the better πBO and BOPrO reflect user knowledge in their selection policy. To aim to a fair comparison, we opted for such strong priors. Furthermore, we aimed to show that IBO-HPC reliably recovers from receiving large amounts of strongly misleading knowledge. Sometimes, it is easier for users to specify a concrete value for certain hyperparameters instead of defining a distribution. Thus, we also conducted experiments with priors defined as points.

F.2 SEARCH SPACE EXTENSION OF JAHS

To make the HPO on JAHS more challenging, we decided to extend the search space slightly as JAHS – as a surrogate benchmark – allows us to query hyperparameter values which were not actually evaluated in the benchmark. We defined three search spaces for JAHS which are presented in the following Table 1.

	S1	S2	S3
Activation	[Mish, ReLU, Hardswish]	[Mish, ReLU, Hardswish]	[Mish, ReLU, Hardswish]
Learning Rate	[1e-3, 1e0]	[1e-3, 1e0]	[1e-3, 1e0]
Weight Decay	[1e-5, 1e-2]	[1e-5, 1e-2]	[1e-5, 1e-2]
Trivial Argument	[True, False]	[True, False]	[True, False]
Op1	0-6	0-6	0-6
Op2	0-6	0-6	0-6
Op3	0-6	0-6	0-6
Op4	0-6	0-6	0-6
Op5	0-6	0-6	0-6
Op6	0-6	0-6	0-6
N	1-15	1-11	1-5
W	1-31	1-23	1-16
Epoch	1-200	1-200	1-200
Resolution	0-1	0-1	0-1

Table 1: **JAHS Search Space.** We define three versions of the JAHS search space, ranging from simpler to harder spaces.

F.3 INTERACTIONS

Here we provide the interactions used for our experiments.

JAHS. The following JSON code shows the interactions performed in our JAHS experiments. The first interaction is a misleading interaction, followed by a beneficial interaction and a no interaction (for recovery).

```
[
    {
        "type": "bad",
        "intervention": {"Activation": 1, "LearningRate": 0.8201676371308472, "N": 15,
        "Op1": 3, "Op2": 4, "Op3": 1, "Op4": 2, "Resolution": 0.5096959403985494,
        "TrivialAugment": 0, "W": 14,
         "WeightDecay": 0.002697686639935806, "epoch": 10},
        "iteration": 5
    },
    {
        "type": "good",
        "intervention": {"N": 3, "W": 16, "Resolution": 1},
        "iteration": 15
    },
    {
        "type": "good",
        "intervention": null,
        "iteration": 20
    },
    {
        "type": "good",
        "kind": "dist",
        "intervention": {"N": {"dist": "cat", "parameters":
        [1, 1, 1, 1e4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]},
        "W": {"dist": "cat", "parameters":
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1e4]},
        "Resolution": {"dist": "uniform", "parameters": [0.98, 1.02]}},
        "iteration": 5
    }
]
```

NAS-Bench-101. The following JSON code shows the interactions performed in our experiments on NAS-Bench-101. The first interaction is a misleading interaction, followed by a beneficial interaction and a no interaction (for recovery).

```
[
    {
        "type": "bad",
        "kind": "point",
        "intervention": [0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1],
        "iteration": 5
    },
    {
        "type": "good",
        "kind": "point",
        "intervention": [1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1],
        "iteration": 12
    },
    {
        "type": "good",
        "kind": "point",
        "intervention": null,
        "iteration": 20
    },
    {
        "type": "good",
        "kind": "dist",
        "intervention": {
            "e_0_1": {"dist": "cat", "parameters": [1, 1e4]},
            "e_0_2": {"dist": "cat", "parameters": [1e4, 1]},
            "e_0_3": {"dist": "cat", "parameters": [1, 1e4]},
            "e_0_4": {"dist": "cat", "parameters": [1e4, 1]},
            "e_0_5": {"dist": "cat", "parameters": [1, 1e4]},
            "e_0_6": {"dist": "cat", "parameters": [1, 1e4]},
            "e_1_2": {"dist": "cat", "parameters": [1, 1e4]},
            "e_1_3": {"dist": "cat", "parameters": [1e4, 1]},
            "e_1_4": {"dist": "cat", "parameters": [1e4, 1]},
            "e_1_5": {"dist": "cat", "parameters": [1e4, 1]},
            "e_1_6": {"dist": "cat", "parameters": [1e4, 1]},
            "e_2_3": {"dist": "cat", "parameters": [1e4, 1]},
            "e_2_4": {"dist": "cat", "parameters": [1, 1e4]},
            "e_2_5": {"dist": "cat", "parameters": [1e4, 1]},
            "e_2_6": {"dist": "cat", "parameters": [1e4, 1]},
            "e_3_4": {"dist": "cat", "parameters": [1e4, 1]},
            "e_3_5": {"dist": "cat", "parameters": [1, 1e4]},
            "e_3_6": {"dist": "cat", "parameters": [1e4, 1]},
            "e_4_5": {"dist": "cat", "parameters": [1, 1e4]},
            "e_4_6": {"dist": "cat", "parameters": [1e4, 1]},
            "e_5_6": {"dist": "cat", "parameters": [1, 1e4]}
        },
        "iteration": 5
    }
]
```

NAS-Bench-201. The following JSON code shows the interactions performed in our experiments on NAS-Bench-201. The first interaction is a misleading interaction, followed by a beneficial interaction and a no interaction (for recovery).

```
[
    {
        "type": "good",
        "kind": "point",
        "intervention": {"Op_0": 2, "Op_1": 2, "Op_2": 0},
        "iteration": 5
    },
    {
        "type": "bad",
        "kind": "point",
        "intervention": {"Op_0": 1, "Op_1": 2, "Op_2": 1},
        "iteration": 5
    },
    {

        "type": "good",
        "kind": "point",
        "intervention": null,
        "iteration": 20
    },
    {
        "type": "good",
        "kind": "dist",
        "intervention": {"Op_0": {"dist": "cat", "parameters": [1, 1, 1e4, 1, 1]},
                         "Op_1": {"dist": "cat", "parameters": [1, 1, 1e4, 1, 1]},
                         "Op_2": {"dist": "cat", "parameters": [1e4, 1, 1, 1, 1]}},
        "iteration": 5
    }
]
```

F.4 HYPERPARAMETERS OF IBO-HPC

IBO-HPC comes with a few hyperparameters itself, which have to be set. For our experiments, we set the number of iterations the surrogate is not retained to $L = 20$ and the decay value to $\gamma = 0.9$. For a fair comparison, we let all methods optimize for 2000 iterations. The learning algorithm of our surrogate models, i.e. PCs for hybrid domains (Molina et al., 2018), also has some hyperparameters to be set. Its structure learning algorithm splitting procedure employs the RDC independence test, thus, we set the threshold to detect independencies between random variables to 0.3 (Molina et al., 2018). Furthermore, we set the minimum number of instances per leaf μ to 40. Refer to Molina et al. (2018) for further details on the learning algorithm.

F.5 HARDWARE

We ran all our experiments on DGX-A100 machines and used 10 CPUs for each run, thus, parallelizing some sub-routines (e.g. learning of PCs). We did not use any GPUs as we queried the selected benchmarks to provide the performance of configurations. The JAHS benchmark requires a relatively large RAM ($> 16GB$) to run smoothly as it loads large ensemble models.

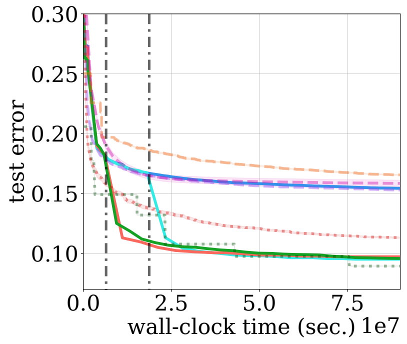
F.6 ADDITIONAL RESULTS & ABLATION STUDIES

In this section, we provide further results and ablations. Fig. 5 demonstrates that IBO-HPC's convergence rate remarkably increases when beneficial user beliefs (either as points or distributions) are provided at various points in time. In general, early beneficial interactions lead to faster convergence, i.e., the earlier a beneficial interaction is done, the faster IBO-HPC converges. However, also later beneficial interactions lead to a significant speed-up, as shown in Fig. 11. Besides the speed-up, the quality of the solution found is considerably better when beneficial beliefs are provided.

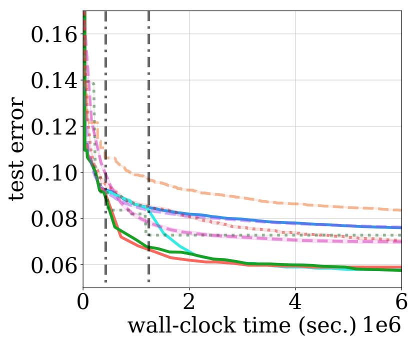
The capabilities of IBO-HPC to recover reliably are shown in Fig. 6. Here, we provided a strongly misleading user beliefs (as points) at iteration 5. It can be seen that, without further user interaction, IBO-HPC manages to recover and achieves almost the same performance as if no interaction was provided. Moreover, the results show that IBO-HPC successfully incorporates multiple user interactions (here at iteration 5 (misleading) and iteration 20 (beneficial)). It can be seen that after the misleading belief is provided, IBO-HPC starts to recover, and when provided with a beneficial belief, its convergence speed remarkably increases. Furthermore, the quality of the solutions found has improved considerably.

Fig. 7 shows the CDF of test accuracy across the baselines and IBO-HPC. It can be seen that IBO-HPC spends more computational resources in good-performing configurations than other methods while achieving state-of-the-art or better results. In other words, IBO-HPC avoids exploration in unpromising regions of the search space. This is because IBO-HPC samples configurations from a conditional distribution where the condition is the best evaluation score obtained so far. Thus, exploration is purely data-driven and focuses on regions that perform similarly to the incumbent solution at a particular iteration.

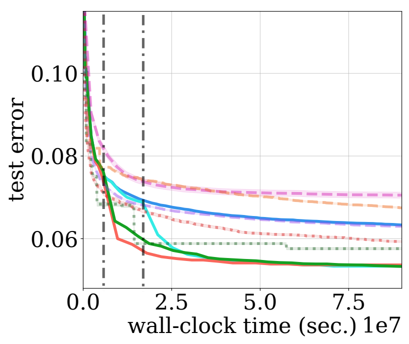
Fig. 8 shows the influence of the decay parameter γ in cases where harmful or misleading user knowledge was provided to IBO-HPC at an early iteration (10 in this case). It can be seen that for higher γ, IBO-HPC requires more time to recover than for smaller γ. This aligns with our expectations since a larger γ corresponds to a high likelihood of the user knowledge being used for many iterations. In contrast, if γ is small, likely, the user knowledge is only considered for a certain number of iterations with high likelihood. Thus, for smaller γ IBO-HPC can recover faster.



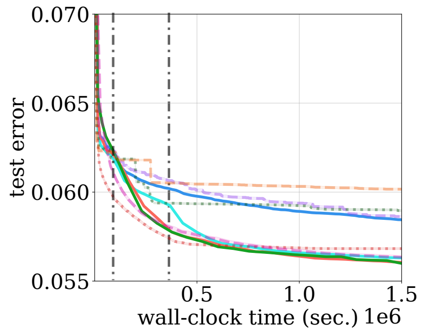
(a) JAHS (CIFAR-10)



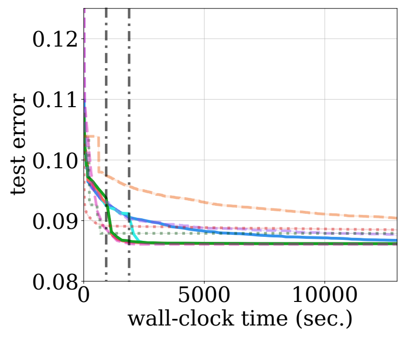
(b) JAHS (C. Histology)



(c) JAHS (F-MNIST)



(d) NAS-Bench-101 (CIFAR-10)



(e) NAS-Bench-201 (CIFAR-10)

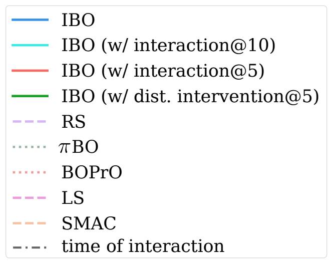


Figure 5: **IBO-HPC outperforms state of the art.** For 5 tasks across three challenging benchmarks, IBO-HPC is competitive with state-of-the-art methods when no user knowledge is provided. When beneficial user beliefs (vertical dotted line) are provided, either as distributions (green) or point values (orange, cyan), it outperforms all competitors w.r.t. convergence and solution quality on most tasks. Early interactions (green/orange=5th iteration, cyan=10th iteration) speed convergence up.

Fig. 9 shows the effect of conditioning on the $\{0.25, 0.5, 0.75\}$-quantile of the obtained evaluation scores instead of the maximum evaluation score. As expected, the higher the quantile, the better the performance of IBO-HPC as we aim to maximize the objective function. Thus, conditioning on higher values guides the optimization algorithm to configurations that yield better evaluation scores.

Lastly, Fig. 10 depicts the effect of changing L, i.e., the number of samples drawn from the surrogate before the surrogate is updated. We found that the sample size has no effect on the overall performance of IBO-HPC. However, for some tasks (JAHS CIFAR-10 and CO), depending on the choice of L, we detect a relevant variation of convergence speed in early iterations. Choosing $L = 20$ seems to lead to fast and stable convergence.

In Fig. 8–10, we followed the same experimental protocol as for all other experiments except that each algorithm has been run on each task with different random seeds 100 times instead of 500 times.

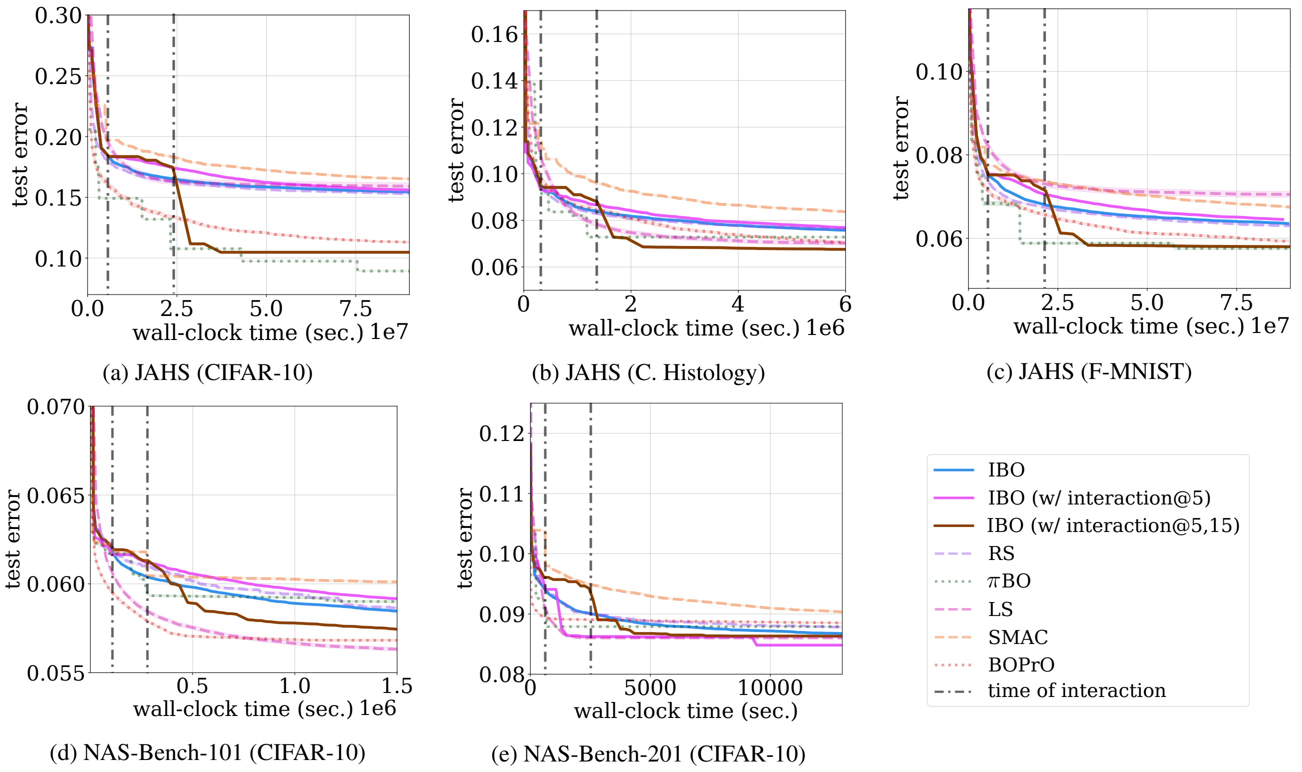
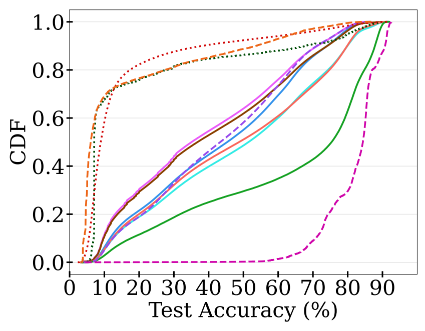
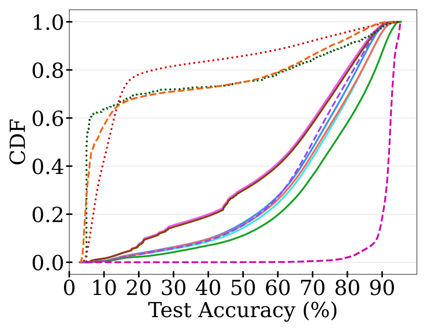


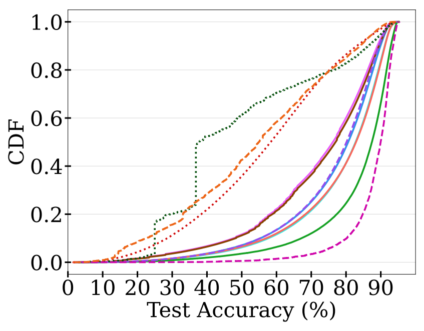
Figure 6: **IBO-HPC recovers from misleading interactions.** IBO-HPC automatically recovers (solid pink) from misleading feedback provided as points at the 5th iteration of the search (1st grey dash-dotted vertical line). After providing beneficial beliefs at iteration 20 (2nd grey dash-dotted vertical line), IBO-HPC (solid brown) catches up with or outperforms πBO and BOPrO.



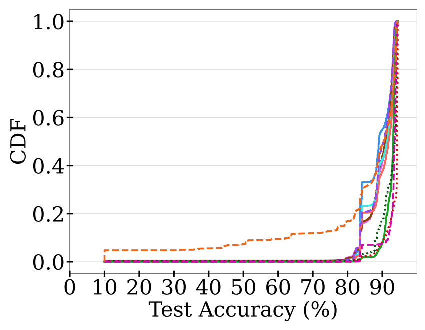
(a) JAHS (CIFAR-10)



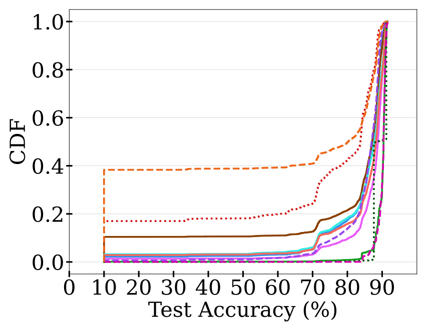
(b) JAHS (C. Histology)



(c) JAHS (F-MNIST)



(d) NAS-Bench-101 (CIFAR-10)



(e) NAS-Bench-201 (CIFAR-10)

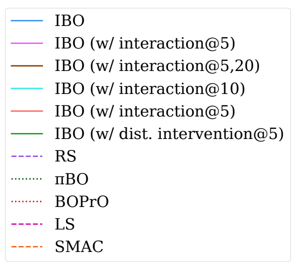
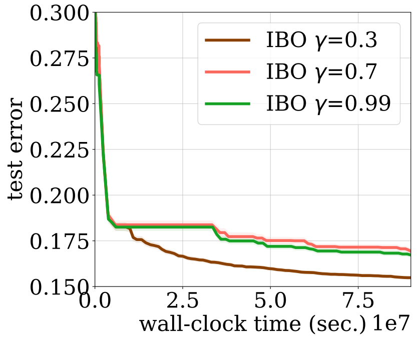
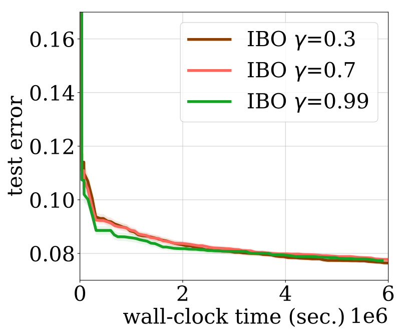


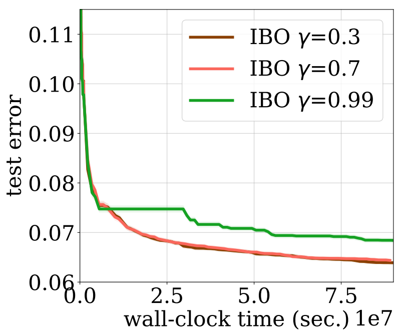
Figure 7: **CDF of test accuracy.** IBO-HPC samples represent better configurations than most other BO baselines on most tasks. Thus, IBO-HPC invests more computational resources in good configurations than other methods. We conjecture that this is because IBO-HPC selects configurations s.t. they are likely to perform similarly to the incumbent solution in each iteration. Interestingly, RS also samples several good configurations on the JAHS benchmark.



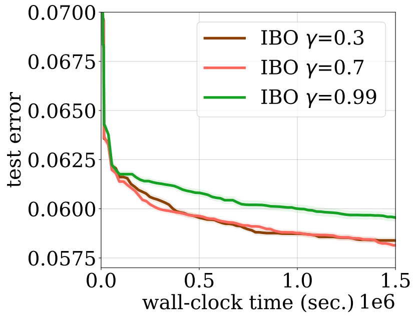
(a) JAHS (CIFAR-10)



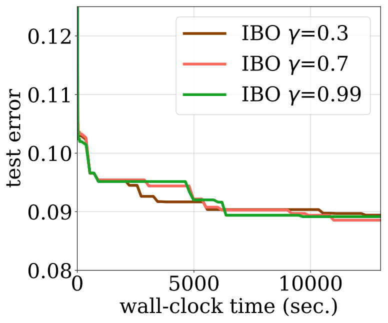
(b) JAHS (C. Histology)



(c) JAHS (F-MNIST)

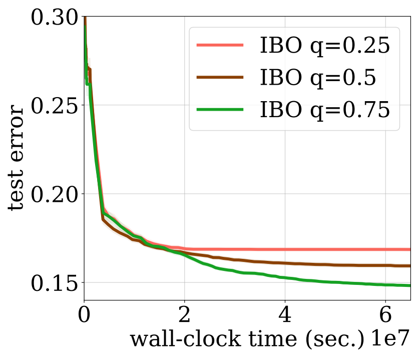


(d) NAS-Bench-101 (CIFAR-10)

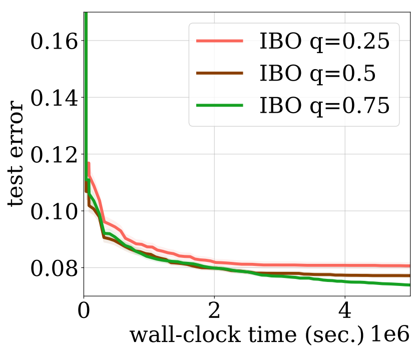


(e) NAS-Bench-201 (CIFAR-10)

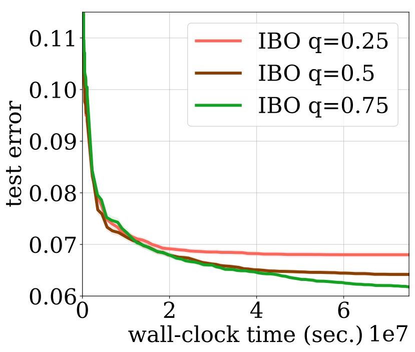
Figure 8: **Ablation: Effect of γ on recovery of IBO-HPC.** As expected, we found that IBO-HPC recovers faster for smaller values of γ. This is due to the smaller γ values leading to a higher decay of the probability of conditioning on the provided user knowledge. Thus, with faster decay, IBO-HPC recovers faster from harmful or misleading user knowledge (provided at iteration 10).



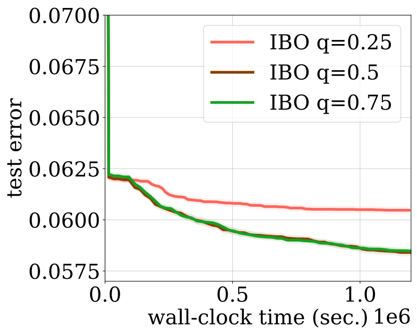
(a) JAHS (CIFAR-10)



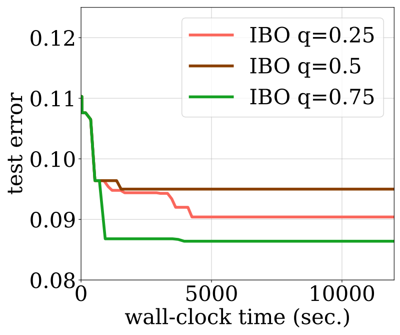
(b) JAHS (C. Histology)



(c) JAHS (F-MNIST)

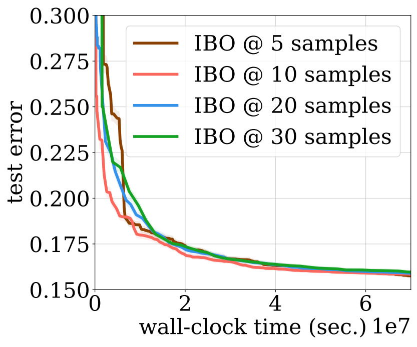


(d) NAS-Bench-101 (CIFAR-10)

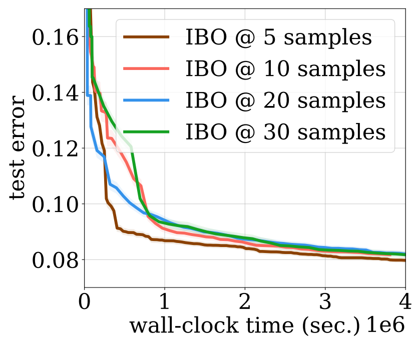


(e) NAS-Bench-201 (CIFAR-10)

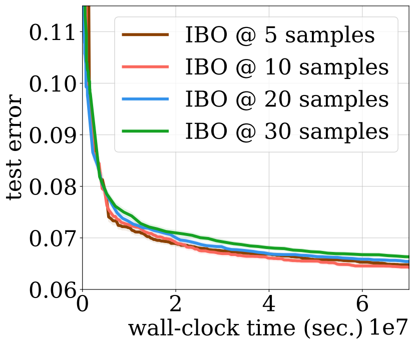
Figure 9: **Conditioning on suboptimal evaluation scores slows down IBO-HPC.** Conditioning on the evaluation score of good configurations is crucial for the performance of IBO-HPC. To analyze the effect of conditioning on evaluation scores of suboptimal configurations, we conditioned on the $\{0.25, 0.5, 0.75\}$-quantile of all evaluation scores obtained until iteration t. As expected, for higher quantiles (i.e., better evaluation scores), IBO-HPC finds better configurations.



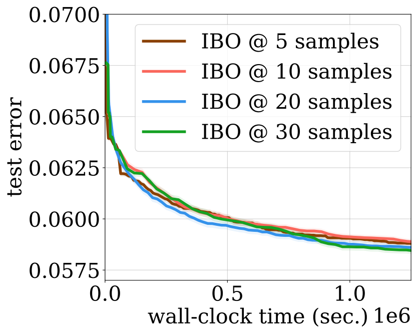
(a) JAHS (CIFAR-10)



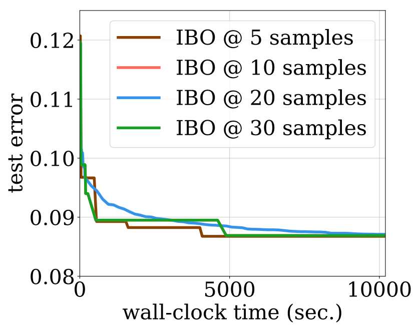
(b) JAHS (C. Histology)



(c) JAHS (F-MNIST)



(d) NAS-Bench-101 (CIFAR-10)



(e) NAS-Bench-201 (CIFAR-10)

Figure 10: ***L* has no considerable effect on IBO-HPC's performance.** We found that updating the surrogate model every $L = \{5, 10, 20, 30\}$ iterations does not lead to considerable differences in the performance and convergence speed of IBO-HPC. We observed a considerable variation in convergence speed only in early iterations with JAHS, on CIFAR-10 and CO tasks. However, these variations vanish in the course of the optimization.

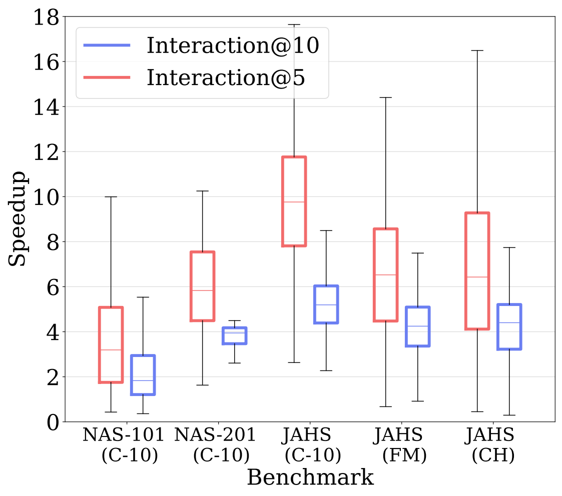


Figure 11: **User beliefs speed up IBO-HPC.** Beneficial interactions lead to significant median convergence speed-ups, from 2 to 10×.