Hyperparameter Optimization via Interacting with Probabilistic Circuits

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

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

Abstract Despite the growing interest in designing truly interactive hyperparameter optimization (HPO) methods, to date, only a few allow to include feedback from experts. However, these methods add friction to the interactive process, rigidly requiring to fully specify the expert input as prior distribution ex ante and often imposing additional constraints on the optimization framework. This hinders the flexible incorporation of expertise and valuable knowledge of domain experts, which might provide partial feedback at any time during optimization. To overcome these limitations, we introduce a novel Bayesian optimization approach leveraging tractable probabilistic models named probabilistic circuits (PCs) as surrogate model. PCs encode a tractable joint distribution over the hybrid hyperparameter space and enable exact conditional inference and sampling, allowing users to provide valuable insights 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 have to be set with care to fully unleash the algorithm's potential. For example, in deep learning, to achieve high performance it is necessary to find optimal values for hyperparameters such as learning rate and batch size. A poor hyperparameter configuration, even for only one of them, could drastically drop the model performance (Bergstra and Bengio, 2012; Hutter et al., 2013; Probst et al., 2019). Manually finding good hyperparameters is a tedious and costly task. Thus, various approaches have been proposed to automatize this process which is referred to as *hyperparameter optimization* (HPO) (Bischl et al., 2023). HPO consists of three distinct components: a search space that defines possible configurations, a search strategy to navigate through the search space with the aim of efficiently finding a good candidate, and an evaluation function that indicates how good a candidate is. The evaluation function is commonly a black-box function, often with no closed form or information like gradients w.r.t. the hyperparameters that could guide the optimization. Moreover, the evaluation function is often computationally demanding. Therefore, a recurrent desideratum consists of minimizing the number of candidate evaluations.

To date, numerous search strategies have been devised with the goal of effectively traversing the search space. A simple strategy, often surprisingly competitive, is random search (RS) which operates by randomly sampling configurations (Bergstra and Bengio, 2012). However, in RS each hyperparameter configuration is chosen independently, disregarding relationships between hyperparameters and previous choices, leading to potentially wasteful computations. On the contrary, Bayesian optimization (BO) methods stand out for their ability to efficiently navigate the search space. They exploit knowledge about previously evaluated configurations to faster converge

[*]indicates equal contribution

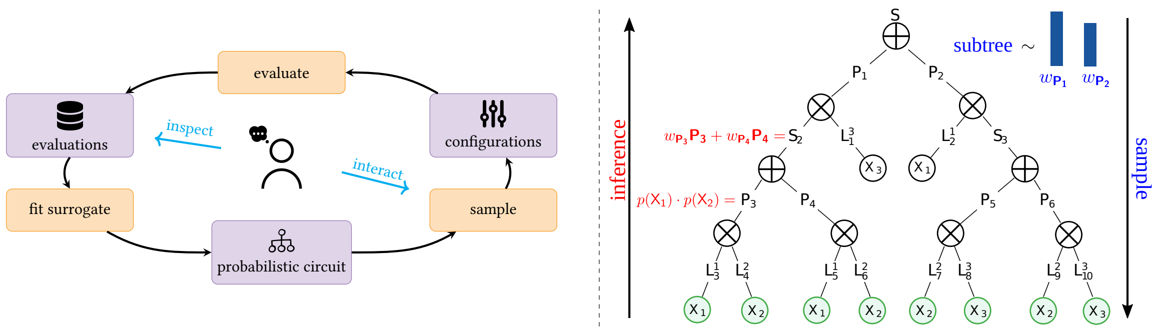


Figure 1: **Interactive Bayesian Hyperparameter Optimization**. We achieve interactive Bayesian HPO by controlling the sampling process of new configurations based on previous evaluations and optional expert knowledge (left). Probabilistic circuits (right) are employed as surrogate models which model a joint distribution over hyperparameters and evaluation score(s). They can answer to a wide range of probabilistic queries and perform sampling in a tractable fashion, allowing experts to take control of the generation of new candidates via efficient conditional sampling. Inference is performed by evaluating the circuit bottom-up while conditional sampling is achieved by following with a top-down pass.

on promising solutions (Hutter et al., 2011; Falkner et al., 2018). These methods leverage a surrogate model whose aim is to capture the underlying characteristics of the unknown objective function being optimized based on the information gained through previous evaluations. Thus, the surrogate model is used to identify the next most promising configuration to be evaluated balancing the exploitation of well-known regions of the search space with the exploration of undiscovered ones.

Albeit the recent advancements in HPO and neural architecture search (NAS) could facilitate the design and optimization of neural models for non-experts, the majority of cutting-edge neural architectures, e.g., transformers (Vaswani et al., 2017), are derived and optimized manually. Thus, the ability to integrate expert knowledge and interactive expert-guided optimization is of high value and can substantially foster the search and mitigate its cost. While being efficient in search space navigation, current BO methods lack the ability to easily integrate expert knowledge for an arbitrary subset of hyperparameters at arbitrary points in time during optimization. This might be helpful in several circumstances. For instance, based on initial evaluations, an expert might want to fix certain dimensions of the search space during optimization to observe whether such search bias is beneficial and helps in finding a better-performing configuration. After a few optimization steps, the expert might decide to change their search bias, or even remove it, and assess whether performances can be further improved. Although the recent efforts made to allow users infusing prior knowledge into an HPO algorithm *before* optimization and related to the *entire search space* (Hvarfner et al., 2022; Giovanelli et al., 2023), there is still a lack of methods enabling expert-guided interactive exploration of *arbitrary subspaces* any time *during* optimization. Unfortunately, this limitation with the additional shared requirement of formulating feedback in form of a probability distribution hinder the expert's interaction with an ongoing search process.

In this paper, we introduce Interactive Bayesian Optimization via Hyperparameter Probabilistic Circuits (IBO-HPC)[1] as a novel BO method that enables the agile integration of expert knowledge at any time during optimization. This is achieved by employing as a surrogate model a fairly recent tractable probabilistic model called probabilistic circuits (PCs) (Poon and Domingos, 2011) encoding a joint distribution over the hyperparameter space and evaluation space. In the following, we will refer to these models as *hyperparameter probabilistic circuits* (HPC). By leveraging HPCs, our method empowers the search strategy to navigate towards promising regions of the

[1]We make our code available at `https://github.com/ml-research/ibo-hpc` and provide all logs and data at `https://1drv.ms/u/s!Aty3JfFPZnuutVz0eNifHh6Uhvjz?e=hX2chn`.

search space while allowing the incorporation of a variable amount of expert knowledge at any time. As depicted in Figure 1, such incorporation is performed by exploiting the abilities of HPCs to answer a wide range of probabilistic queries and thus by conditioning the inference of promising configurations with expert knowledge resulting in faster convergence to the best solutions.

To summarize, we make the following contributions: **(1)** We introduce a formal definition of interactivity in HPO and a novel HPO method named IBO-HPC which enables the incorporation of a variable amount of expert knowledge at any time leveraging HPCs. **(2)** We formally show that IBO-HPC conforms to our notion of interactivity. **(3)** We provide an extensive empirical evaluation of IBO-HPC showing that it is competitive with state-of-the-art HPO and NAS methods without expert interaction and outperforms them when leveraging expert knowledge.

2 Related Work

The main goal of our paper is to leverage the inference capabilities of probabilistic circuits to exploit expert feedback in HPO and NAS at any time. While several works have introduced methods to provide explanations to make HPO more trustworthy (Hutter et al., 2014; Moosbauer et al., 2021; Watanabe et al., 2023; Segel et al., 2023), involving feedback during search has received little to no attention and tackled from distinct directions. For instance, in the context of multi-objective problems, Giovanelli et al. (2023) presented a framework where quality indicators of the Pareto front are learned taking into account user preference. From another direction, and more similar to ours, different methods based on BO allow users to incorporate feedback as prior beliefs (Hvarfner et al., 2022), but they are limited to an *ex ante* full specification of the priors in the form of a probability distribution, often with additional constraints such as requiring invertible priors (Ramachandran et al., 2020) or a specific acquisition function (Souza et al., 2021). However, we believe that truly interactive methods should be able to include expert feedback at any time and in a more flexible and simple way, allowing users to narrow down the search space by either fixing desired hyperparameters to specific values or provide a belief in form of a probability distribution. Additionally, we believe that users should interact with the surrogate directly instead of shaping the acquisition function through a distribution such as in Hvarfner et al. (2022). This makes specifying beliefs easier as users do not have to factor in the behavior of the acquisition function and, with that, the exploration-exploitation trade-off. For completeness, before introducing our method, we provide a brief overview of both HPO and NAS.

Hyperparameter Optimization (HPO). Finding a good configuration for a given algorithm often requires a considerable amount of human and computational resources. The main aim of HPO is to ease this process by reframing such search as an optimization task. Besides a systematic search where expert input is essential to narrow the space of candidates making the search feasible, the simplest approach is random search (Bergstra and Bengio, 2012). However, in its simplicity, random search disregards any relevant information about the relationships between configurations and their goodness and thus might require several expensive evaluations before encountering a good solution. On the contrary, Bayesian optimization methods, in particular, sequential model-based ones, are more sophisticated counterparts that aim to learn such relationships based on what has been previously observed. These methods employ a surrogate model to approximate the objective function and use it to evaluate only promising configurations (Hutter et al., 2011; Snoek et al., 2012). When it comes to high-dimensional search spaces, however, BO becomes challenging. Eriksson et al. (2019) tackle high-dimensional BO problems by defining trust-based regions in which the objective can be approximated by simple models (e.g. linear models) instead of a global surrogate.

Neural networks are preponderant in machine learning. Still, to perform well, they need an accurate hyperparameter tuning and long training procedures. Therefore, methods tailored to neural models try to save computation by avoiding the complete training and base their judgment on fidelities. For instance, Domhan et al. (2015) try to save computation by predicting the loss curves based on the performance achieved after a few training iterations. A related method proposed by Li et al. (2018) integrates an early-stopping strategy that allocates resources only to the most promising candidates selected with a bandit-based approach. To further avoid unnecessary evaluations Falkner et al. (2018) combine a bandit approach with BO. A different direction is taken by evolutionary methods that enhance stochastic methods employing a heuristic function (Jaderberg et al., 2017).

Neural Architecture Search (NAS). Most neural models achieving state-of-the-art performance are crafted by hand, requiring significant efforts and expertise. Thus, there is a great demand of automatic methods to find valuable neural architectures. The first modern approach is based on reinforcement learning (Zoph and Le, 2017) where agents build an architecture and get its evaluation on the task at hand as a reward. Prominent alternatives are based on one-shot methods that use stochastic gradient descent to select a subgraph from a superarchitecture (Pham et al., 2018; Liu et al., 2019). Although being efficient, the major drawback of this approach is the limited search space composed only by a relatively small set of valid subgraphs of the superarchitecture. Surprisingly, recent works have shown that local search (LS) is a strong baseline for NAS, often outperforming more sophisticated methods (White et al., 2020; Den Ottelander et al., 2021). Nevertheless, LS performance strongly depends on the horizons of the neighborhood and on the predefined operators to create it. For an exhaustive review of NAS methods, we point to the work by White et al. (2023).

3 Interactive Hyperparameter Optimization

Our primary goal is to enhance HPO by providing a flexible way to interact with the optimization process. To this aim, we provide a method that enables practitioners to interact by providing a variable amount of knowledge at any desired point during optimization. In this section, we briefly revise Bayesian optimization as a general framework for HPO before introducing our definition of interactivity. Next, we introduce HPCs and present the concrete instantiation of our interactive optimization method. For completeness, we start with a general formal definition of the HPO problem (Kohavi and John, 1995; Hutter et al., 2019).

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

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \Theta} f(\boldsymbol{\theta}; \mathbf{x}) \tag{1}$$

3.1 Interactivity in Bayesian HPO

Bayesian HPO. The aim of BO is to optimize a black-box objective function $f : \Theta \to \mathbb{R}$ which is costly to evaluate, i.e., to find the input $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \Theta} f(\boldsymbol{\theta})$ that minimizes f (Shahriari et al., 2016). BO typically tackles such problem in sequential steps, leveraging two key ingredients: a probabilistic surrogate model and an acquisition function. Given a set $\mathcal{D}_n$ of observations, that in our case correspond to the configurations with their evaluations $(\boldsymbol{\theta}_j, f(\boldsymbol{\theta}_j))_{j=1\dots n}$, the surrogate aims to induce $p(f|\mathcal{D}_n)$ while the acquisition function $a : \Theta \to \mathbb{R}$, such as expected improvement (Jones et al., 1998), estimates the utility of an evaluation at an arbitrary point $\boldsymbol{\theta}$. The configuration $\boldsymbol{\theta}'$ which is evaluated next is selected by maximizing a. The obtained tuple $(\boldsymbol{\theta}', f(\boldsymbol{\theta}'))$ is added to $\mathcal{D}_n$ and used to update the surrogate model for the next iteration. This process is repeated until there is resource budget available or convergence.

An interactive Bayesian optimization method should be capable of incorporating, at any time, the knowledge provided by the experts to update the surrogate model and the search space Θ accordingly. Thus, as commonly done in the literature, defining a prior distribution over f is not sufficient and a broader set of interactions must be allowed (e.g., setting hyperparameters to an exact value). Before introducing our method, we now provide a formalization of this idea of interactivity in the following definition.

Definition 2 (Interactivity). *Given a Bayesian optimization algorithm A using a surrogate model $s : \Theta \rightarrow \mathbb{R}$ approximating f for a given problem instance $\mathbf{x} \in \mathcal{X}$, and given $\mathcal{K} \in \boldsymbol{\mathcal{K}}$ with $\boldsymbol{\mathcal{K}}$ being the set of external knowledge (e.g., constraints or priors), we define interactivity by an operator transform$(s, \Theta, \mathcal{K}) : \mathcal{S} \times \Phi \times \boldsymbol{\mathcal{K}} \rightarrow \mathcal{S} \times \Phi$ which transforms a surrogate s from the space of all surrogates $\mathcal{S}$ and a search space $\Theta \in \Phi$ from the set of search spaces Φ s.t. it incorporates the provided $\mathcal{K}$.*

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

```
Data: Search space $\Theta$ over $\mathcal{H} = \{H_1, \dots, H_n\}$, problem instance $\mathbf{x} \in \mathcal{X}$, prior distribution $p(\Theta)$, evaluation function $f : \mathcal{X} \times \Theta \rightarrow \mathbb{R}$, $J$ initial samples, user knowledge $\mathcal{K}$ is optional and can be provided at any time
1  $\mathcal{D} = \emptyset$;
2  for $i \in \{1, \dots, J\}$ do
3      $\boldsymbol{\theta} \sim p(\Theta)$;
4      $\mathcal{D} = \mathcal{D} \cup \{(\boldsymbol{\theta}, f(\boldsymbol{\theta}, \mathbf{x}))\}$;
5  end
6  while not converged do
7      estimate $p(\mathcal{H}, F)$ w/ HPC $s$ from $\mathcal{D}$;
8      $f^* = \max_f \mathcal{D}$;
9      if user knowledge $\mathcal{K} \neq \emptyset$ then
10         ensure all variables addressed in $\mathcal{K}$ exist in $\mathcal{H}$;
11         $\boldsymbol{\theta}' \sim p(\mathcal{H} \setminus \mathcal{K} | \mathcal{K}, F = f^*)$;
12     else
13         $\boldsymbol{\theta}' \sim p(\mathcal{H} | F = f^*)$;
14     end
15     $\mathcal{D} = \mathcal{D} \cup (\boldsymbol{\theta}', f(\boldsymbol{\theta}', \mathbf{x}))$;
16     present evaluations $\mathcal{D}$;
17 end
```

In Def. 2, the set of knowledge, denoted as $\boldsymbol{\mathcal{K}}$, is intentionally left unspecified as it heavily relies on the choice of algorithm A and surrogate s employed for the optimization process. As an example, in the context of HPO, expert knowledge could be a set of constant values for a subset of hyperparameters, effectively reducing the search space. Similarly, the properties of the transform function are not explicitly defined due to their dependence on the characteristics of the problem and surrogate model at hand.

3.2 Interactive Bayesian Optimization with Hyperparameter Probabilistic Circuits (IBO-HPC)

In this section, we introduce an interactive Bayesian optimization method that fulfills Def. 2. It employs hyperparameter probabilistic circuits as surrogate model levereging their flexible inference and sampling. Before delving into our method, we first provide preliminaries w.r.t. these models.

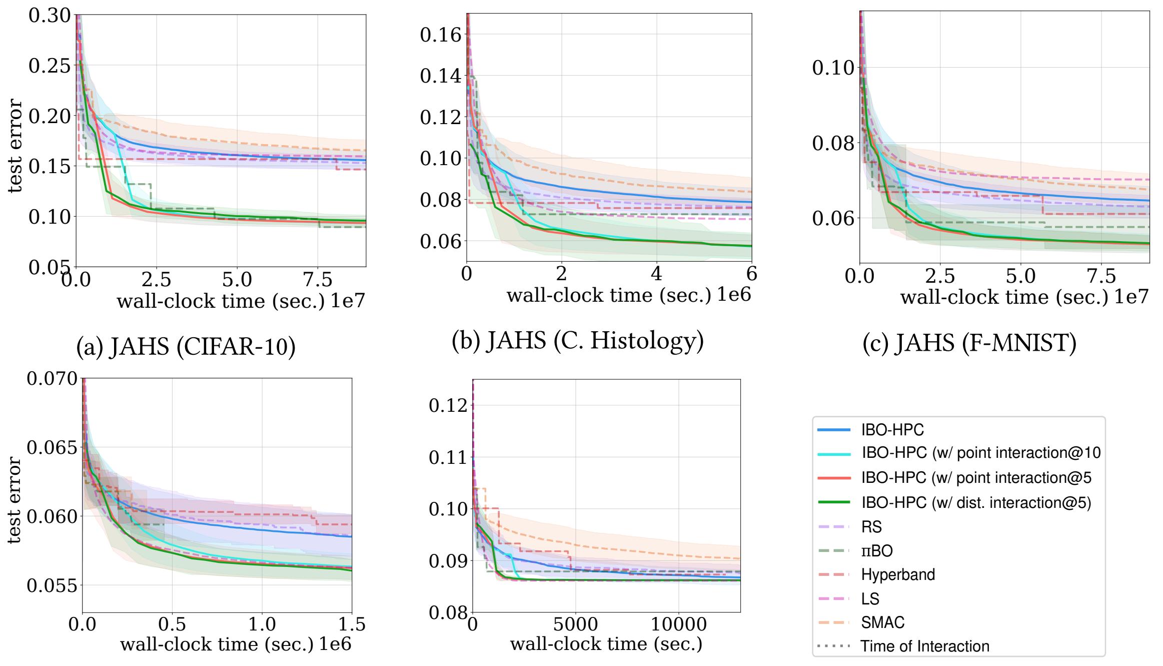
Hyperparameter Probabilistic Circuits (HPCs). Motivated by the lack of truly interactive Bayesian HPO methods, we seek a surrogate model that enables flexible interactions with the optimization procedure by providing an arbitrary amount of knowledge about hyperparameters at any time *during* the optimization. Probabilistic circuits (Choi et al., 2020) are computation graphs that compactly represent multivariate distributions, also in the challenging case of hybrid domains. They have been successfully applied on a variety of applications, e.g., in robotics (Zheng et al., 2018), computer vision (Stelzner et al., 2019), and time series (Yu et al., 2021). PCs can answer a wide range of probabilistic queries in a tractable fashion and (conditionally) generate new samples. These features make them a good candidate for our purpose, i.e., learning a joint distribution over the search space and efficiently querying it for interactive HPO.

More formally, a PC encodes a distribution over a set of random variables $\mathbf{Z}$ and is defined as a tuple $(\mathcal{G}, \phi)$ where $\mathcal{G} = (V, E)$ is a rooted, directed acyclic graph and $\phi : V \rightarrow 2^{\mathbf{Z}}$ 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 $\mathrm{ch}(\mathsf{N})$. Each leaf node L computes a distribution/density over its scope. 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., $\mathsf{P} = \prod_{\mathsf{N} \in \mathrm{ch}(\mathsf{P})} \mathsf{N}$. In our case, the random variables jointly model the hyperparameter search space, thus, we refer to these surrogates as hyperparameter probabilistic circuits (HPC). Figure 1 shows an example of a PC. 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 PC employing piecewise polynomial leaves which is decomposable and smooth. These properties guarantee to encode a valid distribution, and allow efficient inference and sampling with a variable amount of evidence such as expert feedback (Peharz et al., 2015).

Method. We now describe our method shown in Algorithm 1. As usual in Bayesian HPO, we start off by sampling J hyperparameter configurations from a prior distribution p, e.g., a uniform distribution. Each sampled configuration is evaluated by a function f **(Line 1-5)**. Typically f evaluates the performance of a model trained to solve a given problem instance $\mathbf{x}$ with the sampled configuration $\boldsymbol{\theta}$. This yields a performance score for each sampled $\boldsymbol{\theta}$ indicating how well a certain configuration works for the given problem. After evaluating each sampled $\boldsymbol{\theta}$ we obtain a set $\mathcal{D}$ of pairs $(\boldsymbol{\theta}, f_{\boldsymbol{\theta}}(\mathbf{x}))$ that we employ to fit a HPC s estimating the joint distribution $p(\mathcal{H}, F)$, where $\mathcal{H}$ is the set of hyperparameters and F is a random variable representing the evaluation score **(Line 7)**. IBO-HPC proceeds by selecting a batch of configurations that gets evaluated next. We target the configurations which are likely to achieve a better evaluation score. Thanks to the flexible inference of HPCs, we are no longer limited to a conditional posterior of the form $p(F|\mathcal{H})$, but we can derive arbitrary conditional distributions according to the partial evidence at hand (Peharz et al., 2015). Instead of employing an acquisition function such as Expected Improvement, we leverage the ability of HPCs to exactly compute the conditional distribution and sample R configurations from it that are evaluated next. We obtain a posterior distribution over the hyperparameter space by conditioning on the best score $f^* = \max_f \mathcal{D}$ observed so far and (optional) expert knowledge $\mathcal{K}$ **(Line 8-14)**. With Bayes rule and tractable marginal inference of HPCs, we obtain the conditional distribution as $p(\mathcal{H} \setminus \mathcal{K} | \mathcal{K}, F = f^*) = s(\mathcal{H} \setminus \mathcal{K}, \mathcal{K}, F = f^*)$. $\mathcal{H} \setminus \mathcal{K}$ indicates the set of random variables corresponding to the remaining hyperparameters that we want to conditionally sample. This distribution p can then be used to sample configurations from a promising region of the search space. Hence, conditioning can be viewed as the acquisition function to be in line with the BO framework: Configurations achieving a similar score as the current incumbent according to the HPC are likely to be evaluated in the next round. Due to the stochasticity of the sampling process, exploration is ensured. The procedure is concluded by updating the set of evaluations $\mathcal{D}$ that can be presented to the users **(Line 15-16)**. The algorithm runs until convergence or another condition for termination, e.g., time budget limit, is encountered. To note that a user might also provide an arbitrary evaluation score as feedback and obtain insights on the relationships between the configurations and f. Note that the knowledge $\mathcal{K}$ is assumed to be given in the form of conditioning values such as $H_i = h_i$ where H_i is the i-th hyperparameter being set to h_i in Algorithm 1. However, allowing distributions $H_i \sim p_{H_i}$ to be defined instead of mere conditioning can be easily achieved. In that case, for each H_i an expert can define a distribution. We independently sample R values and apply the conditional sampling scheme from **(Line 8-14)** where we sample one configuration from the conditional for each of the R sampled interactions.

After presenting IBO-HPC as an instance of an IBO algorithm, it remains to show that IBO-HPC formally conforms to the above Def. 2 of interactivity. For this scope, we define the *transform* operation mentioned in the definition to be the conditioning operation in HPCs. The following proposition states that IBO-HPC is indeed interactive following Def. 2 (for proof see Appendix A).

Proposition 1 (Conditioning transform). *Assuming a search space* Θ *over a subset of hyperparameters* $\hat{\mathcal{H}} \subseteq \mathcal{H}$, *a set of knowledge* $\mathcal{K} := \{H_i = h_i : H_i \in \hat{\mathcal{H}}, h_i \in \mathbf{H}_i\}$ *where* $\mathbf{H}_i$ *is the domain of hyperparam-*



(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 outperforms state of the art.** For 5 tasks across three challenging benchmarks, IBO-HPC is competitive with state-of-the-art methods when no expert knowledge is provided. It also meaningfully incorporates valuable expert knowledge. When interactions (vertical dotted line) with beneficial expert knowledge happen, it outperforms all the competitors in terms of convergence and solution quality. Early interactions (in brown at 5th, in blue at 10th iteration) speed convergence up. Note that we omit the standard deviation of LS for better readability, refer to Appendix B for the full plot.

eter H_i, and a HPC $s \in \mathcal{S}$ s.t. its scope includes all hyperparameters in $\mathcal{H}$, then, conditioning is a valid instance of the transform operation.

With this formal description of IBO-HPC interactivity, we can now proceed and evaluate our method empirically.

4 Experimental Evaluation

We now provide an extensive empirical evaluation of IBO-HPC and aim to answer the following research questions: **(Q1)** Can IBO-PC compete with state-of-the-art algorithms in HPO and NAS? **(Q2)** Does providing expert knowledge at various time points during optimization with IBO-HPC improve the convergence speed and quality of the solutions? **(Q3)** Is IBO-HPC capable of recovering from harmful user interactions?

Experimental Setup. 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), Hyperband (Li et al., 2018), and πBO (Hvarfner et al., 2022) which allows the incorporation of user knowledge. For our evaluation, we employ three real-world benchmarks, i.e., NAS-Bench-101 (Ying et al., 2019) and NAS-Bench-201 (Dong and Yang, 2020) as tabular NAS benchmarks, and JAHS (Bansal et al., 2022) as surrogate benchmark for joint architecture and hyperparameter search. We evaluate IBO-HPC on five image classification tasks: CIFAR-10 (JAHS, NAS-Bench-101, NAS-Bench-201), Fashion-MNIST (JAHS), and Colorectal Histology (JAHS). To make the optimization problem more challenging, we extended the search space definition of JAHS (see Appendix B). This is possible since JAHS is a surrogate benchmark. The incumbent solution is selected by using validation accuracy as evaluation score and we report the corresponding test regret in the following. All experiments were run on DGX-A100 machines and repeated with 500 seeds to account for random effects. We report the average test error and standard deviation (the latter in Appendix B) against the computational costs estimated by accumulating the wall-clock time of training and evaluating each sampled configuration. The wall clock time is provided by the employed benchmarks. To estimate the costs, we sum up the estimated wall-clock time for evaluating all the configurations sampled at each optimization iteration.

For the experiments, beneficial and harmful expert interactions have been chosen. Both were identified by querying each benchmark with $10k$ randomly chosen configurations where the best and worst configurations were kept. Only a few hyperparameters were fixed for the beneficial configuration to demonstrate that it is enough to incorporate user knowledge for a small subset of hyperparameters to gain performance. In the case of the harmful interaction, most hyperparameters were set to poor values to demonstrate that IBO-HPC recovers even if a large amount of misleading information is provided, but removed after 15 iterations. We additionally evaluate IBO-HPC in cases where users specify a distribution over possible interactions. This allows a fair comparison to πBO as users are required to define a prior over hyperparameter configurations. The interactions employed above are used to define a distribution strongly favoring the found interaction. For example, if an edge $e_{ij} = 1$ in an interaction (i.e., the edge is supposed to exist in the architecture), a prior is defined s.t. $e_{ij} = 1$ is 1000 times more likely than $e_{ij} = 0$. The defined distributions were used for IBO-HPC and πBO. For a detailed list of interactions, refer to Appendix B.

Following Algorithm 1, at each iteration of IBO-HPC, we conditionally sample $J = 20$ configurations from the surrogate HPC (or uniform distribution in the first iteration). These are then evaluated and added to the dataset of evaluations. Then, the surrogate is learned on this updated set and the procedure reiterated.

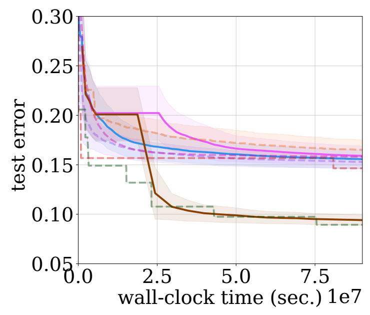
4.1 (Q1) IBO-HPC is Competitive in HPO & NAS

To demonstrate the effectiveness of IBO-HPC, we run IBO-HPC on all tasks with no expert interaction. We show the results in Fig. 2. We find that the performance of IBO-HPC without user interaction is competitive or superior to current state-of-the-art HPO algorithms on most tasks. However, on NAS local search, known to be a strong baseline for NAS (White et al., 2020; Den Ottelander et al., 2021), yields the best results. We attribute this to the discrete nature of the search space where LS is capable of efficiently navigating towards good solutions. Notably, for joint architecture and hyperparameter search spaces, IBO-HPC outperforms all state-of-the-art HPO algorithms as well as LS. These results show that IBO-HPC can also handle more complex and realistic cases well. Besides the quality of the final solution, we also observe that our method comes with a similar convergence speed as the other methods. Thus, we can state that IBO-HPC achieves state-of-the-art results in HPO and NAS when no expert interaction takes place, and answer **(Q1)** affirmatively.

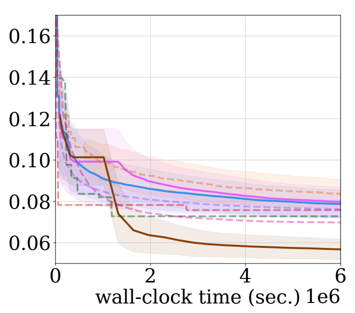
4.2 (Q2, Q3) Interactive and Resilient HPO & NAS with IBO-HPC

The main benefit of IBO-HPC is the incorporation of a variable amount of user knowledge at any time during optimization. We further validate how providing valuable user knowledge positively affects the optimization with IBO-HPC, and how IBO-HPC can recover from harmful interactions.

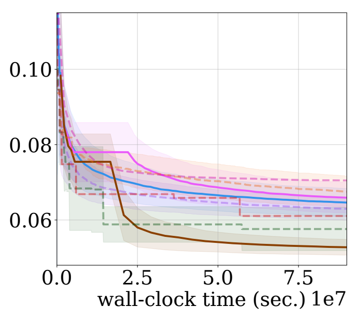
Interactions. We retrieve beneficial/harmful knowledge from the benchmarks by obtaining hyperparameter configurations that yield high/low test performance. Then, we provide parts of these configurations as user knowledge at one or more iterations. For NAS-Bench-101 we fix the architecture skeleton, for NAS-Bench-201 we fix half of the operations per cell, and for JAHS we fix the resolution, depth multiplier, and width multiplier as expert knowledge. Besides fixing certain hyperparameters, we allow users to specify a distribution over interactions. The same



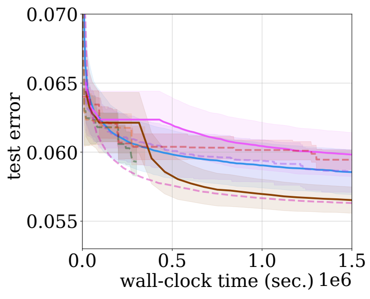
(a) JAHS (CIFAR-10)



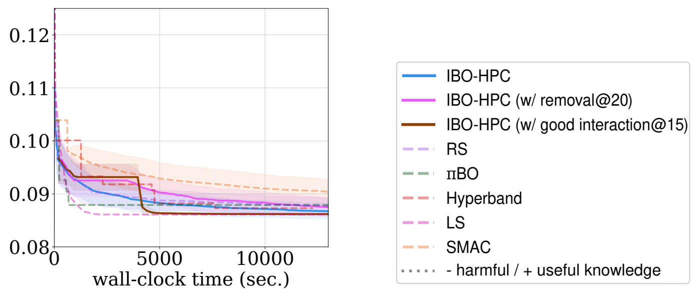
(b) JAHS (C. Histology)



(c) JAHS (F-MNIST)



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



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

Figure 3: **IBO-HPC recovers from harmful interactions**. Harmful feedback is provided at the 5th iteration of the search limiting the performance of the optimization routine. After removing harmful expert knowledge at iteration 20 (blue dotted vertical line), IBO-HPC (blue) converges towards a good solution. Similarly, if beneficial expert knowledge is provided at iteration 15 (brown dotted vertical line), IBO-HPC (brown) outperforms all other methods on most tasks. We omit the standard deviation of LS for better readability, refer to Appendix B for details.

distribution is used as a prior for πBO for fair comparison. For that, we reuse the interactions. However, we define a distribution s.t. the values included in the interaction have a high probability of being sampled while all other possible choices have a low probability. For details, see Appendix B. However, as πBO only allows to define expert knowledge in terms of prior distribution, we define the priors s.t. sampling the interaction used for IBO-HPC is 1000 times more likely than other possible interactions. In Figure 2, a clear positive effect of providing beneficial expert knowledge to IBO-HPC can be seen. This holds for very early interactions (after 5 iterations) and later interactions (after 10 iterations). Remarkably, it is possible to observe a clear benefit in terms of convergence speed and improvement in terms of solution quality, especially for larger and more complex search spaces. Not only does IBO-HPC outperform state-of-the-art HPO algorithms not capable of incorporating expert knowledge, but it also outperforms πBO which is specifically designed to incorporate expert knowledge as prior distribution. Besides being able to incorporate feedback at any point during optimization, another major feature of IBO-HPC is its flexibility. In fact, during optimization experts can interact multiple times, arbitrarily often. To demonstrate that IBO-HPC handles these multiple interactions well, we perform the same experiments as above with instead performing two interactions, one harmful at an early stage (after 5 iterations) and one useful after, at a later stage (after 15 iterations). As expected, the early harmful interaction blocks IBO-HPC at a certain performance level. Later, the useful interaction comes in allowing IBO-HPC to catch up with the competitors or even outperform them showing that IBO-HPC is able to leverage valuable feedback also in critical conditions. Results are shown in Fig. 3.

Speed-up. To provide evidence on the benefit of good expert interactions in terms of convergence speed, we show the average improvement in terms of convergence speed of IBO-HPC with (useful) expert interaction vs. IBO-HPC with no expert feedback during optimization. For the analysis, we run IBO-HPC and obtained the best evaluation result achieved as well as the time needed to get to this result (denoted as t_w). Then, we run IBO-HPC with a useful interaction (an early one after 5 iterations and a later one after 10 iterations) and measured the estimated wall-clock time until IBO-HPC finds an equally well or better-performing configuration (denoted as t_i). We then compute $\frac{t_w}{t_i}$ as the performance speed-up for each run. The results are shown in Fig. 4 and reveal that useful expert interactions lead to a remarkable median speed-up of 2 to 10×. These results clearly show that IBO-HPC is able to leverage expert knowledge improving performance and saving resources. Thus, we can answer **(Q2)** positively.

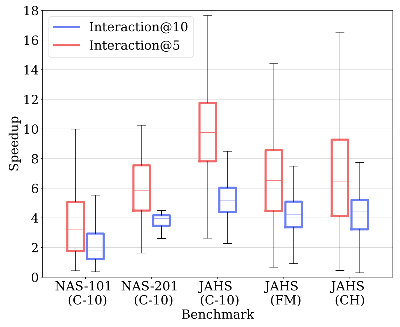


Figure 4: **Expert interaction leads to speed-ups**. Beneficial interactions lead to significant speed-up in terms of convergence, from 2 to 10×.

Harmful Interactions. Expert feedback could also be misleading for the optimization process, thus, an interactive HPO algorithm should recover from such harmful interactions. Here, we demonstrate that IBO-HPC recovers from harmful interactions employing the same protocol as the previous scenarios. However, this time, we retrieve from each benchmark a configuration with a low test performance. For NAS-Bench-101 and NAS-Bench-201 we take the same hyperparameters as above and we set them to the values of the selected bad configuration. For JAHS we additionally fix the activation function, learning rate, and four out of six operations in the architecture to sub-optimal values to ensure a significant harmful effect on the interaction. The interaction is removed after 15 iterations and the optimization is terminated with no further user interaction. Fig. 3 shows that IBO-HPC is able to recover from harmful interactions and to catch up with the competitor methods that have not received any harmful feedback. Thus, we can answer **(Q3)** affirmatively.

5 Conclusion

In this work, we have introduced a definition of interactive BO and a novel interactive BO method named IBO-HPC which leverages the flexible inference of probabilistic circuits to easily incorporate expert feedback at any time during the optimization. IBO-HPC is competitive with state-of-the-art methods when no expert knowledge is given and it can outperform competitors accelerating the optimization when valuable expert knowledge is available, thus, saving resources. Furthermore, IBO-HPC is resilient since it can recover the optimization after harmful feedback is provided.

Limitations & Future Work. Whereas IBO-HPC enables truly interactive BO, the necessity to retrain the surrogate model at each iteration remains. Thus, prospective directions could explore methods of continual learning (Mundt et al., 2023) to increase the overall efficiency and knowledge reuse over different HPO and NAS settings. Also, currently, our method requires users to recognize and remove harmful interactions before recovering from such interactions. This limitation could be tackled similarly to Hvarfner et al. (2022). Moreover, to model hybrid domains, we relied on HPCs employing piecewise polynomials that might not be sufficient to model complex distributions. Therefore, more sophisticated alternatives could further improve performance.

6 Broader Impact Statement

We believe that this work presents no notable negative impacts to society or the environment. We would like to point out that, as we empirically demonstrated, expert feedback can help to quickly guide the HPO procedure towards promising regions in the hyperparameter search space. Thus, it can make HPO and NAS more energy-efficient, reducing its environmental impact.

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).

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*.

Domhan, T., Springenberg, J. T., and Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

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

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. (2019). Scalable global optimization via local bayesian optimization. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Falkner, S., Klein, A., and Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning (ICML)*.

Giovanelli, J., Tornede, A., Tornede, T., and Lindauer, M. (2023). Interactive hyperparameter optimization in multi-objective problems via preference learning. *arXiv preprint arXiv:2309.03581*.

Hutter, F., Hoos, H., and Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. *International Conference on Machine Learning (ICML)*.

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*.

Hutter, F., Kotthoff, L., and Vanschoren, J. (2019). *Automated Machine Learning: Methods, Systems, Challenges*. Springer Nature.

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)*.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. (2017). Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

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)*.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18:1–52.

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*.

Moosbauer, J., Herbinger, J., Casalicchio, G., Lindauer, M., and Bischl, B. (2021). Explaining hyperparameter optimization via partial dependence plots. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mundt, M., Hong, Y., Pliushch, I., and Ramesh, V. (2023). A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*, 160:306–336.

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)*.

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)*.

Poon, H. and Domingos, P. (2011). Sum-product networks: A new deep architecture. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

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.

Segel, S., Graf, H., Tornede, A., Bischl, B., and Lindauer, M. (2023). Symbolic explanations for hyperparameter optimization. In *International Conference on Automated Machine Learning*.

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)*.

Stelzner, K., Peharz, R., and Kersting, K. (2019). Faster attend-infer-repeat with tractable probabilistic models. In *International Conference on Machine Learning (ICML)*.

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)*.

Watanabe, S., Bansal, A., and Hutter, F. (2023). PED-ANOVA: efficiently quantifying hyperparameter importance in arbitrary subspaces. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

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

White, C., Safari, M., Sukthanker, R., Ru, B., Elsken, T., Zela, A., Dey, D., and Hutter, F. (2023). Neural architecture search: Insights from 1000 papers. *arXiv preprint arXiv:2301.08727*.

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)*.

Yu, Z., Ventola, F., and Kersting, K. (2021). Whittle networks: A deep likelihood model for time series. In *International Conference on Machine Learning (ICML)*.

Zheng, K., Pronobis, A., and Rao, R. P. N. (2018). Learning graph-structured sum-product networks for probabilistic semantic maps. In *AAAI Conference on Artificial Intelligence*.

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

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We have provided a formal definition of interactive Bayesian HPO in Section 3 and an extensive experimental evaluation in Section 4 showing its benefits.

   (b) Did you describe the limitations of your work? [Yes] See Section 5.

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Yes, in Section 6 we briefly explained that in our view there are no potential negative impacts and we pointed out that incorporating expert knowledge with the introduced method could be beneficial for the environment since it could save computations.

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` [Yes]

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [Yes] All experiments were executed on the same well-known challenging benchmarks and tasks.

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [Yes] We provide code of the implementation and the listed interactions with additional experimental details in the Appendix B and in Section 4.

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes] Yes, we executed each experiment with 500 different random seeds.

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [Yes] Following common practice, we only report the mean test regret / error, however, we provide variance regarding the speed-ups (Fig. 4).

   (e) Did you report the statistical significance of your results? [No] We did not report statistical significance as our results clearly show the benefit of incorporating expert knowledge.

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes] We used NAS-Bench-101, NAS-Bench-201, and JAHS.

   (g) Did you compare performance over time and describe how you selected the maximum duration? [Yes] See Section 4.

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We did not collect any of these metrics.

   (i) Did you run ablation studies to assess the impact of different components of your approach? [Yes] In our case, the ablation study consisted in providing a variable amount of expert feedback (including no feedback scenario) at different times.

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] We provide code and logs (the URLs) and experimental details in the paper and in the Appendix B.

(b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes] See anonymous GitHub repository.

(c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes] See anonymous GitHub repository.

(d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes] Yes, see Footnote 1.

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes] We include all of our logs and the code to generate plots is available on the anonymous GitHub repository.

4. If you used existing assets (e.g., code, data, models)…

   (a) Did you cite the creators of used assets? [Yes] See Section 4.

   (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [N/A]

   (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you created/released new assets (e.g., code, data, models)…

   (a) Did you mention the license of the new assets (e.g., as part of your code submission)? [No] We will choose an open-source license later on.

   (b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes] We provided the URLs, see Footnote 1.

6. If you used crowdsourcing or conducted research with human subjects…

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

7. If you included theoretical results…

   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix A.

   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.

A Proofs

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

A.1 Conditioning as Valid Transform Operation

Proposition 2 (Conditioning transform). *Assuming a search space Θ over a subset of hyperparameters $\hat{\mathcal{H}} \subseteq \mathcal{H}$, a set of knowledge $\mathcal{K} := \{H_i = h_i : H_i \in \hat{\mathcal{H}}, h_i \in \mathbf{H}_i\}$ where $\mathbf{H}_i$ is the domain of hyperparameter H_i, and a HPC $s \in \mathcal{S}$ s.t. its scope includes all hyperparameters in $\mathcal{H}$, then, conditioning is a valid instance of the transform operation.*

Proof. We have to show that **(1)** conditioning yields a valid HPC again, **(2)** conditioning yields a valid search space, and **(3)** conditioning is reversible. Since HPCs encode joint distributions and each conditional can be represented as $p(H_1, \ldots, H_{n-1} | H_n = h_n) = \frac{p(H_1,\ldots,H_n)}{\int_{h_1} \cdots \int_{h_{n-1}} p(H_1,\ldots,H_{n-1},H_n=h_n)}$, only the evaluation of the HPC s changes, but not the HPC itself, i.e. after conditioning we still have the same HPC s. Thus, **(1)** is fulfilled. The Bayes' rule implies that **(2)** is fulfilled as well since conditioning with HPCs results in a distribution over a subset of hyperparameters. Finally, **(3)** directly follows from **(1)**, as removing the conditions yields the same HPC s and recovers the original search space. □

B Experimental Details

Here we present additional details of our empirical evaluation.

B.1 Search Space Extension of JAHS

To make the HPO problem 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 tested explicitly in the benchmark. We defined three search spaces for JAHS which are presented in the following table.

	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.

B.2 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 harmful 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 harmful interaction, followed by a beneficial interaction and a no interaction (for recovery).

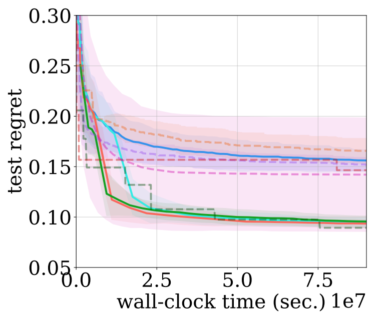
```
[
    {
        "type": "bad",
        "kind": "point",
       "intervention": [0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1],
        "iteration": 5
    },
    {
        "type": "good",
        "kind": "point",
       "intervention": [1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 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 harmful interaction, followed by a beneficial interaction and a no interaction (for recovery).

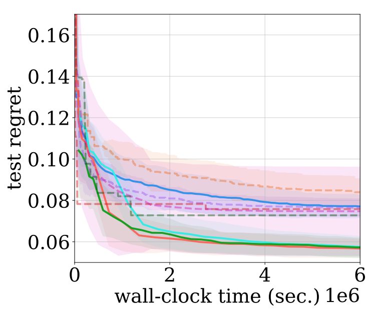
```
[
    {
        "type": "good",
        "kind": "point",
        "intervention": {"Op_0": 2, "Op_1": 2, "Op_2": 0},
        "iteration": 5
    },
    {
        "type": "bad",
        "kind": "point",
        "intervention": {"Op_0": 1, "Op_1": 2, "Op_2": 1},
        "iteration": 5
    },
    {

        "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
    }
]
```

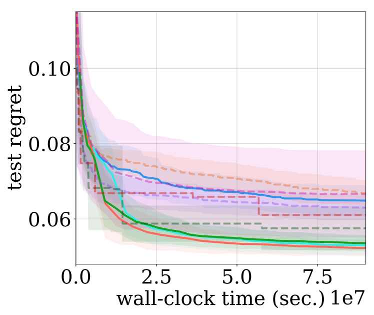
B.3 Results



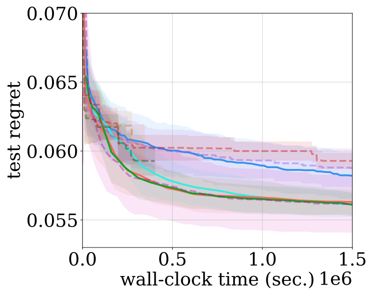
(a) JAHS (CIFAR-10)



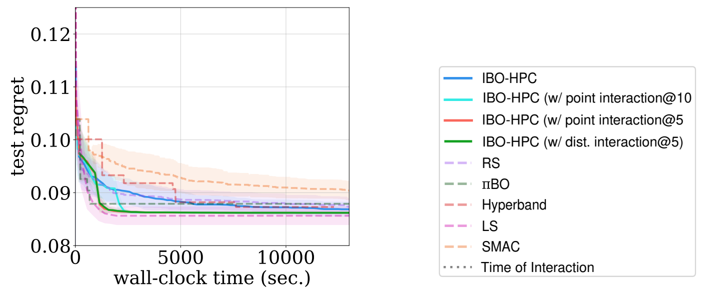
(b) JAHS (C. Histology)



(c) JAHS (F-MNIST)



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



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

Figure 5: **IBO-HPC outperforms existing (interactive) HPO algorithms**. IBO-HPC is competitive with existing state-of-the-art approaches on standard HPO tasks while outperforming πBO in 4/5 tasks when user interaction is provided, also in form of a distribution (green line). This further emphasizes the flexibility of IBO-HPC since it allows to incorporate user feedback in several forms. Notably, LS has a high standard deviation on almost all tasks. We conjecture that this is due to the local nature of the optimization process.

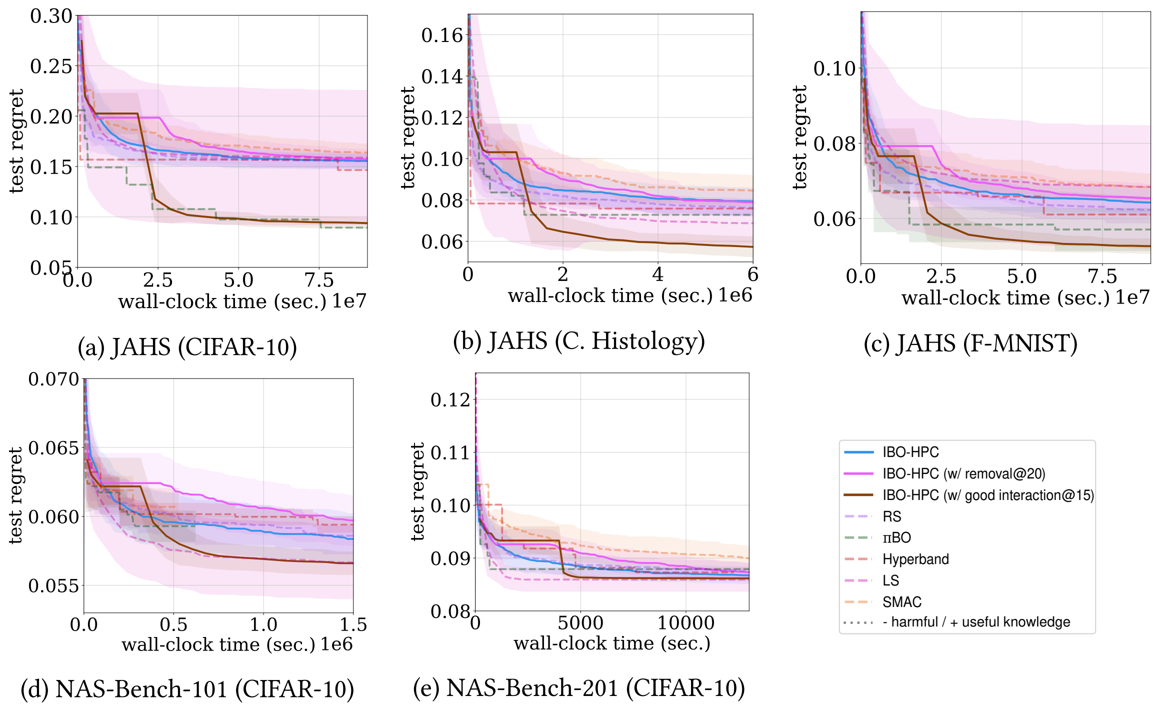


Figure 6: **IBO-HPC recovers from harmful interactions**. User knowledge might be even harmful during optimization. Once users recognize that a set interaction is harmful, IBO-HPC recovers after removal of the harmful interaction. Notably, LS has a high standard deviation on almost all tasks. We conjecture that this is due to the local nature of the optimization process.

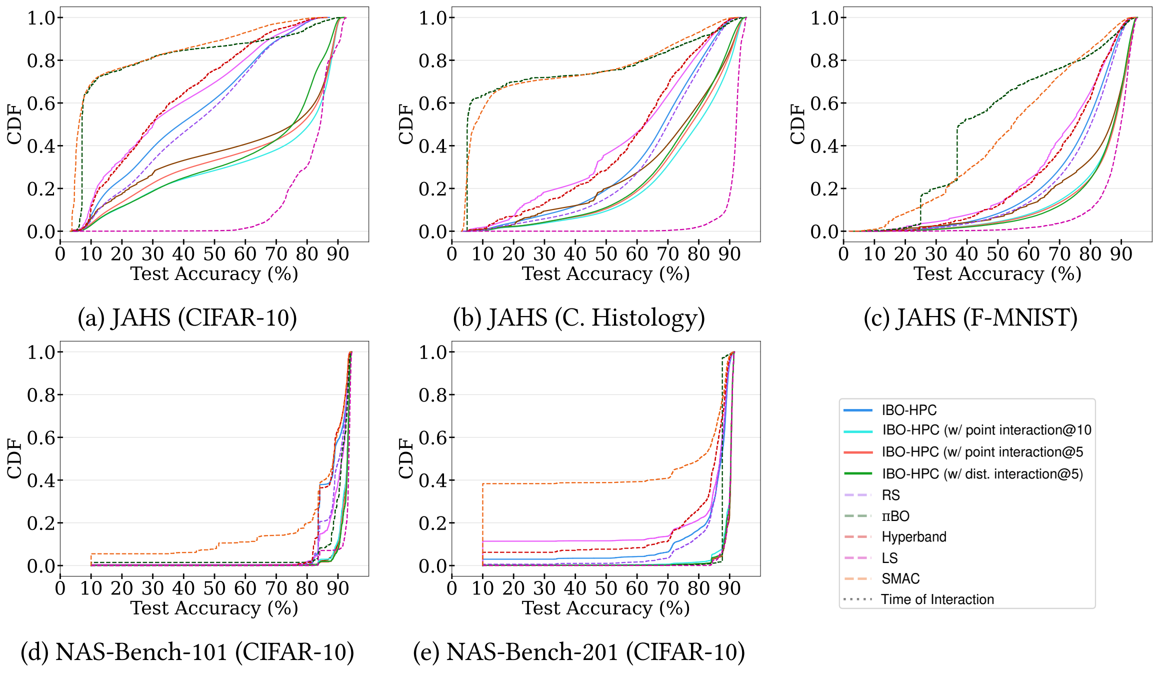


Figure 7: **CDF of Test Accuracy**. IBO-HPC samples more good performing configurations than most of the baseline on most of the tasks. Thus, IBO-HPC invests more computational resources in good configurations than other methods.

C Working Example

In the following we consider a more detailed example of our proposed method from a user perspective. We assume that we only optimize 3 hyperparameters here, W, N and R which correspond to the hyperparameters W, N and Resolution in the JAHS benchmark.

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 Figure 8). From then on, the model only optimizes the remaining hyperparameter R (green in Figure 8), using conditional sampling from the resulting conditional distribution that the HPC represents after the interaction.

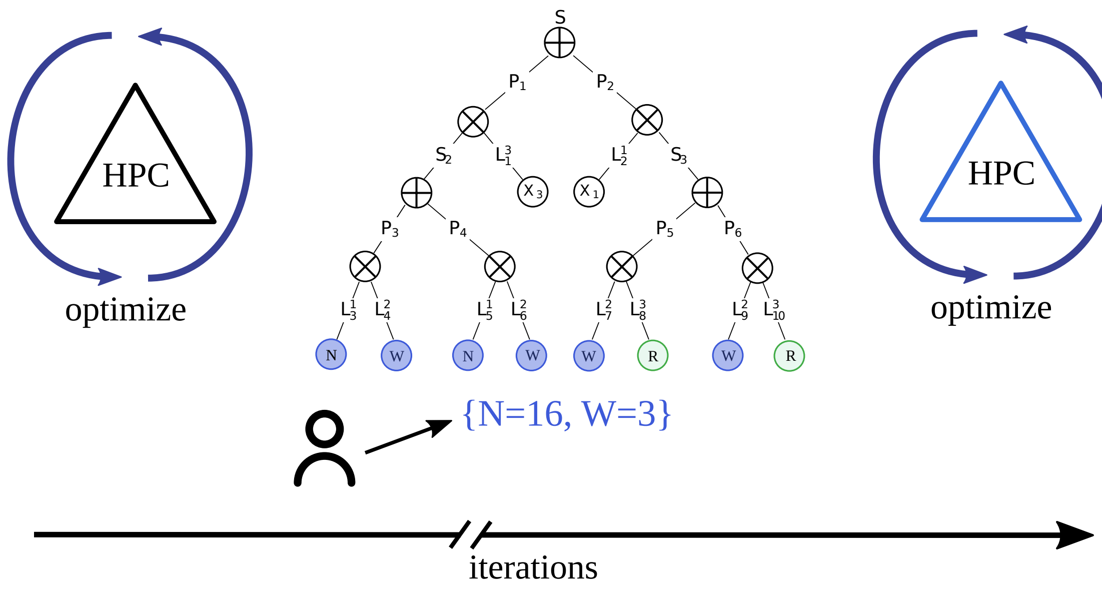


Figure 8: **Example of IBO-HPC**. A user specifies certain aspects of the hyperparameter search space during optimization. Afterward, IBO-HPC takes user knowledge into account when sampling new configuration candidates.