# YAHPO Gym - An Efficient Multi-Objective Multi-Fidelity Benchmark for Hyperparameter Optimization

**Florian Pfisterer**[1] **Lennart Schneider**[1] **Julia Moosbauer**[1] **Martin Binder**[1] **Bernd Bischl**[1]

[1]LMU Munich

**Abstract** When developing and analyzing new hyperparameter optimization methods, it is vital to empirically evaluate and compare them on well-curated benchmark suites. In this work, we propose a new set of challenging and relevant benchmark problems motivated by desirable properties and requirements for such benchmarks. Our new surrogate-based benchmark collection consists of 14 scenarios that in total constitute over 700 multi-fidelity hyperparameter optimization problems, which all enable multi-objective hyperparameter optimization. Furthermore, we empirically compare surrogate-based benchmarks to the more widely-used tabular benchmarks, and demonstrate that the latter may produce unfaithful results regarding the performance ranking of HPO methods. We examine and compare our benchmark collection with respect to defined requirements and propose a single-objective as well as a multi-objective benchmark suite on which we compare 7 single-objective and 7 multi-objective optimizers in a benchmark experiment. Our software is available at [`https://github.com/slds-lmu/yahpo_gym`].

## 1 Introduction

Hyperparameter optimization (HPO) of machine learning (ML) models is a crucial step for achieving good predictive performance [30]. Over the last ten years, a large and still growing set of HPO tuning methods based on different principles has been developed [23, 49, 27]. A particularly interesting development are multi-fidelity methods, which make use of relatively cheap approximations of a given true objective, thereby achieving good performance relatively quickly [32, 15, 25], as well as multi-objective methods, which allow for simultaneous optimization of multiple objectives [29]. While different HPO methods found considerable adoption in practice, it is by no means clear which method performs best under which circumstances. In order to investigate this, it is necessary to evaluate these methods on testbeds that are ideally *i*) highly efficient, *ii*) include a sufficient amount of representative and diverse benchmark instances and *iii*) are easy to set up and integrate with different optimizer APIs. Furthermore, benchmarks have found use in *meta-learning* [53, 57, 44] and *meta-optimization* [36, 39]. In those settings, a larger number of potentially relevant optimization problems is required in order to obtain results that generalize beyond the set of (meta-)training instances. Simultaneously, those applications require a large number of evaluations that make obtaining real evaluations prohibitively expensive, indicating a need for benchmarks that are cheap to query.

Several benchmarks that aim to address this, each of which are collections of multiple benchmark instances, have been proposed [52, 10, 2, 13]. Benchmark instances can be classified into four categories: (i) synthetic functions, (ii) benchmarks incorporating *real* evaluations, (iii) *tabular* benchmarks based on pre-evaluated grid points, and (iv) *surrogate* benchmarks making use of meta-models that approximate the relationship between configurations and performance metrics. These categories have various advantages and drawbacks. Synthetic functions can be evaluated quickly but are often not representative for the type of problems encountered in practice; real evaluations on the other hand are often prohibitively expensive, especially in the context of larger benchmarks and neural architecture search. Tabular benchmarks, while cheap to evaluate, rely

Submitted to AutoML Conference 2022 © 2022 the authors, released under CC BY 4.0

on a pre-defined grid which changes the optimization problem and can potentially lead to biases. Surrogate benchmarks are also cheap to query but require high quality surrogates in order to avoid introducing bias. While benchmark suites have found some use in scientific publications, they are not used ubiquitously. This lack of permeation – and consequently the lack of a standard test bed – can result in researchers choosing benchmark problems that favor their own method, leading to the publication of biased results. The problem of *cherry picking*, also termed *rigging the lottery* [9], can be ameliorated through the use of standardized testing infrastructure along with a detailed definition of evaluation criteria that are widely adapted.

We therefore observe a clear need for benchmark libraries that provide unified interfaces to a variety of cheap to evaluate, realistic, and practically relevant benchmarking problems that are defined across diverse search spaces. In this work, we propose *YAHPO Gym*, a *surrogate-based* benchmark library including a collection of over 700 benchmark instances defined across 14 *scenarios*. Scenarios are comprised of evaluations of one given machine learning algorithm on different datasets (= instances) and therefore share the same search space and performance metrics. It contains a versioned set of surrogate models that allow for *multi-fidelity* evaluations of *multiple objectives*. Our library is licensed under the *GPL-v3* license and can be freely used and extended by the community. Usage and available functionality is extensively documented[1].

**Contributions**: We introduce YAHPO Gym, a surrogate-based benchmark for machine-learning HPO. We conceptually demonstrate that tabular benchmarks may induce bias in performance estimation and ranking of HPO methods, and that this happens to a lesser degree with surrogate benchmarks. We argue that our surrogate benchmark YAHPO Gym meets all desiderata for a good benchmark, providing faithful results, fast evaluation, relevant problems and realistic objective landscapes both on local as well as global scales. In order to demonstrate this, we conduct an extensive evaluation of the proposed surrogates indicating that our surrogate models indeed provide high quality approximations. We propose two benchmark suites for *single-objective* and *multi-objective* evaluation comprised of a subset of our instances and demonstrate how they can be used with YAHPO Gym in a *multi-fidelity* and a *multi-objective* optimization benchmark.

## 2 Related Work

Several efforts to provide unified testbeds for black-box optimization exist. For general purpose black-box optimization, COCO [21] provides a collection of various synthetic black-box benchmark functions, while *kurobako* [42] is a collection of various general black-box optimizers and benchmark problems. Similarly, *Bayesmark* [52] includes several benchmarks for Bayesian Optimization on real problems and *LassoBench* [47] provides a benchmark for high-dimensional optimization problems. *HPOlib* [10] was one of the first to propose a common test bed for empirically assessing the performance of HPO methods. It provides a common API to access synthetic test functions, real-world HPO problems, tabular benchmarks as well as some surrogate benchmarks and found use in empirical benchmark studies [5]. Its successor *HPOBench* [13] offers similar capabilities, focussing on reproducible containerized benchmarks. It offers 12 benchmark scenarios and more than 100 test instances. Recently, [2] introduced *HPO-B*, a large-scale reproducible (tabular) benchmark for black-box HPO based on OpenML [54]. *HPO-B*[2] relies on 16 search spaces that were evaluated sparsely on 101 datasets. *PROFET* [28] in contrast is not based on real datasets but uses a generative meta-model to generate synthetic but realistic benchmark instances. In the past, tabular benchmarks have been used frequently to speed up experiments in the context of HPO [49, 16, 55, 17] and Neural Architecture Search (NAS) (c.f. [37]). Eggensperger et al. [11] compared different instance surrogate models for 9 different HPO problems and

[1]Documentation and data are available at `https://github.com/slds-lmu/yahpo_gym`

[2]We consider the published v2 version for comparison. Surrogates are only available in the v3 version.

Table 1: Comparison of HPO Benchmark Suites.

| Suite | Types | #Collections | #HPs | MF | MO | TF | Async | H | Time† | Memory† |
|---|---|---|---|---|---|---|---|---|---|---|
| YAHPO Gym | S | 14 | 2-38 | ✓ | ✓ | ✓ | (-) | ✓ | $0.4^{*}s$ | 0.1 GB |
| HPOBench | R/T/S | 12 | 4-26 | ✓ | ✓ | (-) | – | (-) | 12.2s | 0.2 GB |
| HPO-B (v2) | T/(S) | 16 | 2-18 | – | – | ✓ | – | – | 18.8s | 3.7 GB |

MF: Multi-fidelity; MO: Multi-objective, TF: Transfer-HPO, Async: Asynchronous evaluation; H: Hierarchical search spaces.
✓: fully supported; (-): partially supported; -: not supported; R/T/S:real/tabular/surrogate.
†: Runtime and memory footprint for 300 iterations of random search on an SVM instance. ∗: allowing for batched evaluation, YAHPO Gym takes only 0.13$s$).

concluded that the results of benchmarks run on surrogate models generally closely mimic those of benchmarks using the actual evaluations that they are derived from, if performance measures of the surrogate models indicate that they predict the underlying objective values sufficiently well (cross-validated Spearman's $\rho$ between 0.9 and 1 [11]). Similar observations have been made in the context of algorithm configuration [12] and NAS [48].

We compare YAHPO Gym with the recently published benchmarks HPOBench [13] and HPO-B [2] in Table 1. Our library relies on high quality surrogates that allow for *multi-fidelity* as well as *multi-objective* evaluation. While existing benchmark suites could in principle be used to construct multi-objective benchmarks, they do not offer full support: HPOBench contains only few instances that allow evaluating multiple metrics and offers no unified API to query those, while HPO-B does not support multiple objectives at all. Furthermore, neither propose a concrete evaluation protocol, opening up a multiplicity of (benchmark) design choices which can lead to inconclusive results (c.f. [41]). Instead of relying on *containerization* to allow for portability, our library relies on neural network surrogates compressed using *ONNX* [3], allowing for reproducibility and portability while simultaneously being extremely fast and efficient due to minimal overhead. This is demonstrated in a small experiment where we measure runtime and memory consumption for evaluating 300 random configurations on SVM search spaces also shown in Table 1, demonstrating that our software is more time and memory efficient. While YAHPO Gym provides the flexibility to design and execute any subset of the provided benchmarks, we also propose two fully specified testbeds for single- and multi-objective optimization that were specifically selected to cover a diverse set of relevant instances while being less extensive. Seed details in **??**.

## 3 Background

### 3.1 Hyperparameter Optimization

An ML *learner* or *inducer* $\mathcal{I}$ configured by hyperparameters $\boldsymbol{\lambda} \in \Lambda$ maps a data set $\mathcal{D} \in \mathbb{D}$ to a model $\hat{f}$, i.e., $\mathcal{I} : \mathbb{D} \times \Lambda \rightarrow \mathcal{H}, (\mathcal{D}, \boldsymbol{\lambda}) \mapsto \hat{f}$. Hyperparameter optimization (HPO) methods for ML aim to identify a well-performing hyperparameter configuration (HPC) $\boldsymbol{\lambda} \in \tilde{\Lambda}$ for $\mathcal{I}_{\boldsymbol{\lambda}}$ [7]. Typically, the considered search space $\tilde{\Lambda} \subset \Lambda$ is a subspace of the set of all possible HPCs: $\tilde{\Lambda} = \tilde{\Lambda}_1 \times \tilde{\Lambda}_2 \times \cdots \times \tilde{\Lambda}_d$, where $\tilde{\Lambda}_i$ is a bounded subset of the domain of the $i$-th hyperparameter $\Lambda_i$. This $\tilde{\Lambda}_i$ can be either real, integer, or category valued, and the search space can contain dependent hyperparameters, leading to a possibly hierarchical search space. We formally define the (potentially multi-objective) HPO problem as:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \tilde{\Lambda}} c(\boldsymbol{\lambda}), \quad \text{with} \quad c : \tilde{\Lambda} \rightarrow \mathbb{R}^m, \tag{1}$$

where $\boldsymbol{\lambda}^*$ denotes the theoretical optimum and $c$ maps an arbitrary HPC to (possibly multiple) target metrics. The classical HPO problem is defined as $\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \tilde{\Lambda}} \widehat{\mathrm{GE}}(\boldsymbol{\lambda})$, i.e., the goal is to minimize the estimated generalization error, see [7] for further details. Instead of optimizing only

for predictive performance, other metrics such as model sparsity or computational efficiency of prediction (e.g., MACs and FLOPs or model size and memory usage) could be included, resulting in a multi-objective HPO problem [46, 22, 6, 43, 20]. $c(\boldsymbol{\lambda})$ is a black-box function, as it usually has no closed-form mathematical representation, and analytic gradient information is generally not available. Furthermore, the evaluation of $c(\boldsymbol{\lambda})$ can take a significant amount of time. Therefore, the minimization of $c(\boldsymbol{\lambda})$ forms an *expensive black-box* optimization problem.

Many HPO problems allow for approximations of the objective to a varying fidelity, making *multi-fidelity optimization* a viable option [32, 46, 25]. E.g., in the context of fitting neural networks, it is possible to stop or pause training runs early when performance does not indicate a promising final result [50]. Another possibility is given by reducing the fraction of the dataset $\mathcal{D}_{\text{train}}$ used for training [27], since the complexity of evaluating $c(\boldsymbol{\lambda})$ is often at least linear in $|\mathcal{D}_{\text{train}}|$. Formally, the possibility of multi-fidelity evaluation can be represented in the form of a "budget" hyperparameter which we denote by $\lambda_{\text{budget}}$ as a component of $\boldsymbol{\lambda}$.

### 3.2 Hyperparameter Optimization Benchmarks

Benchmark suites are comprised of a set of benchmark *instances* that each define an optimization problem to be solved. We formally define benchmark instances adapted from [13] as:

**Definition 1 (Benchmark Instance)** *A benchmark instance consists of a function $g : \Lambda \rightarrow \mathbb{R}^m, m \in \mathbb{N}^+$, and a bounded hyperparameter space $\tilde{\Lambda}$ which is the Cartesian product of hyperparameters $\tilde{\Lambda}_1, \ldots, \tilde{\Lambda}_d$. Multi-fidelity benchmarks can be queried at lower fidelities by varying the budget parameter $\tilde{\Lambda}_{budget} \in \tilde{\Lambda}$.While hyperparameters $\tilde{\Lambda}_i$ can be continuous, integer, ordinal or categorical, we require at least ordinal scales for the fidelity parameter(s) $\Lambda_{budget}$. We call a benchmark instance multi-objective if the number of objectives $p > 1$ and single-objective otherwise.*

We consider HPO benchmark instances estimating the generalization error $g(\boldsymbol{\lambda}) = \widehat{\text{GE}}(\mathcal{I}, \mathcal{J}, \rho, \boldsymbol{\lambda})$ given an inducer $\mathcal{I}$, resampling $\mathcal{J}$, and performance metric(s) $\rho$, along with other possibly relevant metrics (computational cost, memory, ...). *Real* instances are based on actually performing these evaluations during the benchmark, while *tabular* instances are based on a fixed set of pre-recorded evaluations. Instances based on *surrogates* in turn approximate the functional relationship between $\boldsymbol{\lambda}$ and $g(\boldsymbol{\lambda})$. For clarity, we provide more precise definitions of *synthetic, tabular* and *surrogate* instances in Section **??** of the supplement. *Real* instances rely on live evaluations of the generalization error and are therefore often prohibitively computationally expensive, especially when considering larger benchmarks or meta-learning scenarios across many tasks [53, 44, 18]. Practitioners therefore often rely on *tabular* or *surrogate* benchmarks for large benchmark studies because they are often cheaper to evaluate by orders of magnitude. For *tabular* benchmarks, a large collection of pre-computed hyperparameter performance mappings is provided, which serves as a look-up table during runs of HPO methods. This has the downside of constraining the search space to precomputed evaluations, essentially turning the optimization problem from a *continuous/mixed space* to a *discrete* optimization problem. *Surrogate* benchmarks can strike a balance between the efficiency and faithful approximation to the real problem by learning the functional relationship between hyperparameters and performance values yielding an approximation $\hat{g}(\boldsymbol{\lambda})$ of $g(\boldsymbol{\lambda})$. This allows evaluations across the full search space $\Lambda$ while being considerably cheaper to evaluate. The usefulness of surrogates in turn relies on the approximation quality of the surrogate model. We present an in-depth analysis of approximation qualities of the surrogates employed in YAHPO Gym in **??**.

**Definition 2 (Benchmark Scenario)** *A benchmark scenario consists of a set of $K$ functions $g_k : \Lambda \rightarrow \mathcal{Y} \subseteq \mathbb{R}^m, m \in \mathbb{N}^+, k \in \{1, ..., K\}$ corresponding to a set of* Benchmark Instances. *Each instance within a scenario shares the same bounded hyperparameter space $\tilde{\Lambda}$ (and therefore fidelity parameters) as well as the same co-domain $\mathcal{Y}$.*

Table 2: YAHPO Gym Benchmarks.

| Scenario | Search Space | # Instances | Target Metrics | Fidelity | H |
|---|---|---|---|---|---|
| rbv2_super | 38D: Mixed | 103 | 9: perf(6) + rt(2) + mem | fraction | ✓ |
| rbv2_svm | 6D: Mixed | 106 | 9: perf(6) + rt(2) + mem | fraction | ✓ |
| rbv2_rpart | 5D: Mixed | 117 | 9: perf(6) + rt(2) + mem | fraction | |
| rbv2_aknn | 6D: Mixed | 118 | 9: perf(6) + rt(2) + mem | fraction | |
| rbv2_glmnet | 3D: Mixed | 115 | 9: perf(6) + rt(2) + mem | fraction | |
| rbv2_ranger | 8D: Mixed | 119 | 9: perf(6) + rt(2) + mem | fraction | ✓ |
| rbv2_xgboost | 14D: Mixed | 119 | 9: perf(6) + rt(2) + mem | fraction | ✓ |
| nb301 | 34D: Categ. | 1 | 2: perf(1) + rt(1) | epoch | ✓ |
| lcbench | 7D: Cont. | 34 | 6: perf(5) + rt(1) | epoch | |
| iaml_super | 28D: Mixed | 4 | 12: perf(4) + inp(3) + rt(2) + mem(3) | fraction | ✓ |
| iaml_rpart | 4D: Cont. | 4 | 12: perf(4) + inp(3) + rt(2) + mem(3) | fraction | |
| iaml_glmnet | 2D: Cont. | 4 | 12: perf(4) + inp(3) + rt(2) + mem(3) | fraction | |
| iaml_ranger | 8D: Mixed | 4 | 12: perf(4) + inp(3) + rt(2) + mem(3) | fraction | ✓ |
| iaml_xgboost | 13D: Mixed | 4 | 12: perf(4) + inp(3) + rt(2) + mem(3) | fraction | ✓ |

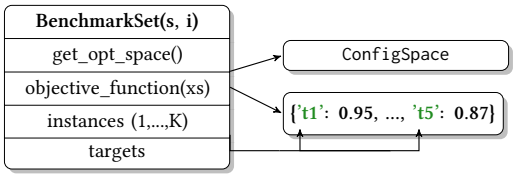
Mixed = numeric and categorical hyperparameters; perf = performance measures; rt = train/predict time; mem = memory consumption; inp = interpretability measures; H = Hierarchical search space. We do not include the fidelity parameter in the search space dimensionality.

A scenario is therefore a collection of instances sharing the same search space and objective(s), e.g., allowing for hyperparameter transfer learning between instances of the scenario. *Benchmark Suites* in turn are sets of instances that do not need to share the same objectives, but instead can consist of instances stemming from different scenarios.

## 4 YAHPO Gym

Motivated by the need for efficient and faithful benchmarks for HPO, we develop YAHPO Gym based on a set of *Criteria for HPO Benchmarks* discussed in **??**. YAHPO Gym is explicitly designed to use surrogate-based benchmarks only. It consists of a collection of 14 *scenarios* that can be evaluated across a total of $\sim 700$ instances. Each benchmark instance consists of an objective function that is parameterized in the form of a `ConfigSpace` Python object [34], making the search space computer-readable and readily usable with a range of existing HPO implementations. The objective function generates a prediction using the instance surrogate model, which is a compressed neural network. Table 2 provides an overview of all benchmark scenarios available in YAHPO Gym. We describe data sources as well as the full search spaces in **??**. We want to highlight the *rbv2_super* collection, which reflects an AutoML pipeline: It is, to our knowledge, the first available benchmark simulating a combined algorithm and hyperparameter selection problem [51] in the form of a high dimensional hierarchical search space by introducing the algorithm as an additional tunable hyperparameter.

In YAHPO Gym, every scenario allows for querying objective values at lower fidelities, enabling efficient benchmarking of multi-fidelity HPO methods. Analogously, every benchmark allows for returning multiple target metrics as criteria, enabling benchmarking of multi-objective HPO methods. Finally, almost all benchmark scenarios provide problems on a large number of instances (ranging from 34 to 119), allowing for benchmarking of transfer-learning HPO methods. Predictions as well as sampling can be made reproducible through seeding. In order to achieve *portability* while still being *efficient*, YAHPO Gym uses fitted neural networks compressed via `ONNX` [3] as surrogate models. Our neural networks are `ResNets` for tabular data [19] consisting of up to 8 layers with a width of up to 512 and hyperparameters individually tuned for each scenario. We refer the reader to **??** for details regarding architecture and fitting procedure. Surrogate models have very small memory and inference time overhead and are compatible across platforms




(a) YAHPO Gym's core functionality (**s**: scenario, **i**: instance, **xs**: configuration). Evaluating `objective_function` for a given configuration **xs** returns a dictionary of predicted metrics for a given scenario and instance.

```python
from yahpo_gym import *
b = BenchmarkSet('lcbench', instance='3945')
# Sample a point from the ConfigSpace
xs = b.get_opt_space().sample_configuration(1)
# Evaluate the configuration
b.objective_function(xs)
```

(b) Python code for instantiating a benchmark instance, sampling a new configuration and evaluating the objective function.

and operating systems. In contrast to other benchmarks, evaluating $c(\boldsymbol{\lambda})$ requires only $10 - 100$ ms and only 100 MB of memory. In fact, YAHPO Gym's current infrastructure is so lightweight, it can easily be integrated in any existing toolbox or benchmark suite.

### 4.1 Suites: YAHPO-SO & YAHPO-MO

Together with YAHPO Gym, we propose two carefully selected *benchmark suites*. They constitute a proposal for surrogate-based benchmarks of HPO problems. We call those YAHPO-SO (single-objective, 20 instances) and YAHPO-MO (multi-objective, 25 instances). Together with the set of instances, we provide specific evaluation criteria, such as the budget available for optimization and number of stochastic replications as well as metrics to be used and fully specified search spaces which can be obtained from our software. Instances were selected across all scenarios taking into account approximation quality of the underlying surrogate and diversity. We consider those benchmarks a first draft for such a benchmark set (version **v1**) and explicitly invite the community to jointly work on a larger, more comprehensively evaluated set of benchmark instances. Details with respect to how instances were selected, and a full list of included instances, can be found in Section **??** in the Supplement. We conduct a benchmark providing anytime performance for a large variety of baselines on the proposed benchmark suites.

## 5 Tabular or Surrogate Benchmarks?

Consider the true objective $c(\boldsymbol{\lambda})$ of a *real* benchmark instance with $c : \tilde{\Lambda} \rightarrow \mathbb{R}$ in the single-objective setting. In a *tabular* benchmark, the domain of the objective function is implicitly discretized into a finite grid $\tilde{\Lambda}_{\text{discrete}}$ of the original domain and pre-evaluated at these points and the benchmark objective $\hat{c}_{\text{tabular}}(\boldsymbol{\lambda})$ is thus the original $c(\boldsymbol{\lambda})$ restricted to $\tilde{\Lambda}_{\text{discrete}}$. The extent to which discretization affects the faithfulness of tabular benchmarks depends on the nature and dimensionality of the search space: It disregards local structure in the response function and might even impose fixed fidelity schedules, should evaluations not be available at all budget levels. In order to assess the magnitude of this effect, we investigate the practical effects of discretization in the following experiment by comparing 8 black-box optimizers on *tabular*, *surrogate* and *real* versions of 5 synthethic multi-fidelity functions of varying dimensionality (Branin2D, Currin2D, Hartmann3D/6D, and Borehole8D [25]). The tabular benchmark is constructed by drawing and evaluating $10^6$ points from a grid. Surrogates are then fitted using those points. We compare Random search (`RS`), several versions of Bayesian optimization (`BO`) and Hyperband (`HB`, [32]) across all settings. `BO` is configured with algorithm surrogate model either a Gaussian process (`BO_GP`), ensemble of feed-forward neural networks (`NN`, [56]) or random forest (`BO_RF`, [8]) and acquisition function optimizer either Nelder-Mead/exhaustive search[3] (`*_DF` [40]) or random search (`*_RS`). We describe additional details regarding the benchmark setup in **??** and briefly present results: Figure 2 shows the anytime performance and mean rank of each HPO method split for the real,

[3] for tabular benchmarks

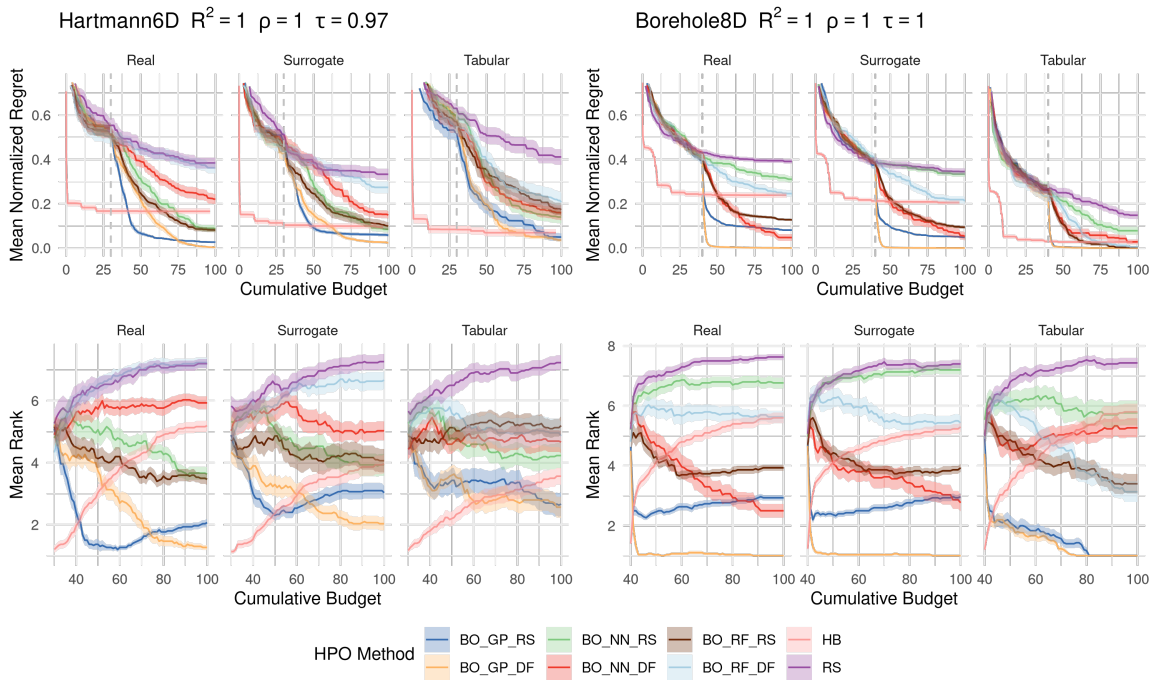



Figure 2: Mean normalized regret (top) and mean ranks (bottom) of different HPO methods on different benchmarks. Ribbons represent standard errors. The gray vertical line indicates the cumulative budget used for the initial design of BO methods. Performance measures of the surrogate benchmarks are stated after the benchmark function. 30 replications.

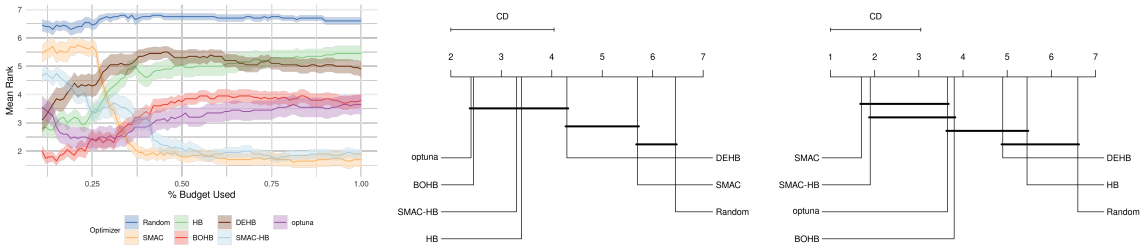
surrogate, and tabular benchmark on the Hartmann6D and Borehole8D test functions. We observe very similar performance traces of HPO methods on surrogate versions of benchmarks compared to real versions (Figure 2, top). However, in tabular benchmarks, we notice that for some problems, the BO methods converge substantially faster to a lower mean normalized regret (especially for `BO_GP_*`), which can possibly be explained by the much simpler infill optimization problem solved in the tabular case. Moreover, Hyperband appears to consistently perform better on tabular benchmarks. We further investigate average rankings over all replications (Figure 2, bottom). Each benchmark function yields an average ranking of HPO methods (e.g., with respect to final performance). Using consensus rankings, we can arrive at a single ranking over all benchmark functions [38] for a given benchmark type. We use the optimization based symmetric difference (SD) [26] minimizing rank reversals to compare both the surrogate and tabular inferred consensus rankings with the "ground truth" real function consensus ranking. We observe that consensus rankings obtained using surrogate benchmarks (permutation order 2) match more closely than tabular benchmarks (permutation order 5). We again provide additional details in **??**.

## 6 A Benchmark of HPO Methods on YAHPO Gym

We now demonstrate how YAHPO Gym can be used in practice to benchmark different HPO methods. We benchmark 7 single-objective HPO methods on YAHPO-SO and 7 multi-objective HPO methods on YAHPO-MO and want to answer the following research questions: (**RQ1**) *Do multi-fidelity (single-objective) HPO methods improve over full-fidelity methods?* (**RQ2**) *Do advanced multi-objective HPO methods improve over Random Search?*

### 6.1 RQ1: Do multi-fidelity (single-objective) HPO methods improve over full-fidelity methods?

We compare Random Search and SMAC [35] to the multi-fidelity methods Hyperband [32], BOHB [15], DEHB [4], SMAC-HB [35] and optuna ([1]; TPE sampler and median pruner following successive halving steps). More details on the experimental setup and HPO methods is given in **??**. All optimizers are run for a total budget of $\lceil 20 + 40 \cdot \sqrt{\text{SEARCH_SPACE_DIM}} \rceil$ full-fidelity evaluations with 30 replications. Figure 3a shows the average rank of HPO methods with respect to their anytime performance. Figure 3b and Figure 3c show critical difference plots ($\alpha = 0.05$) of mean ranks after 25% and 100% of the optimization budget. The corresponding Friedman tests indicate significant differences ($p < 0.001$) in both cases. We observe that all multi-fidelity optimizers outperform Random Search with respect to intermediate performance (25% of optimization budget) and optuna, BOHB, SMAC-HB and Hyperband also outperform SMAC. With respect to final performance, SMAC takes the lead closely followed by SMAC-HB with other multi-fidelity optimizers slightly falling behind. We conclude that multi-fidelity HPO methods indeed improve over full-fidelity methods, but only with respect to intermediate performance. Our results are in line with what has been reported in other benchmarks [13] with the exception that optuna seems more competitive in our benchmark, while DEHB is less competitive. One reason for this difference might be that we include hierarchical search spaces in contrast to previous work.




(a) Mean ranks of HPO methods. x-axis starts after 10%.

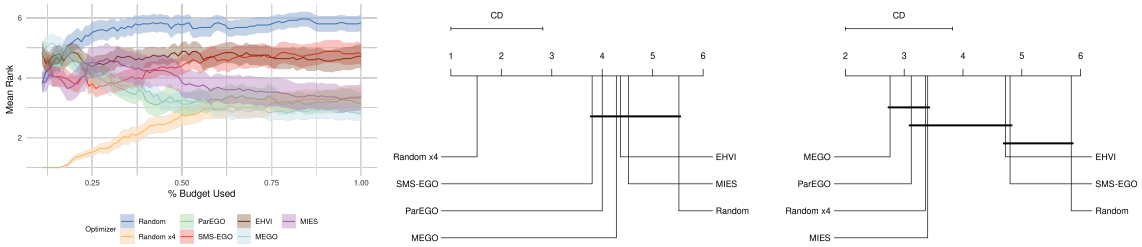
(b) Critical differences plot for mean ranks of HPO methods after 25% of the optimization budget.

(c) Critical differences plot for mean ranks of of HPO methods after 100% of the optimization budget.

Figure 3: Results of YAHPO-SO single-objective benchmark across 7 optimizers (20 Instances).

### 6.2 RQ2: Do advanced multi-objective HPO methods improve over Random Search?

We compare Random search, Random search x4 (random search with quadrupled budget), ParEGO [29], SMS-EGO [45], EHVI [14], MEGO [24] and MIES [33] on multi-objective HPO problems with $2 - 4$ objectives. More details on the experimental setup and HPO methods is given in **??**. All optimizers are run for a total budget of $\lceil 20 + 40 \cdot \sqrt{\text{SEARCH_SPACE_DIM}} \rceil$ full-fidelity evaluations for 30 replications. Figure 4a shows the average rank of HPO methods with respect to their anytime performance (determined based on the normalized Hypervolume Indicator). Figure 4b and Figure 4c show critical difference plots ($\alpha = 0.05$) of these ranks after 25% and 100% of the optimization budget. The corresponding Friedman tests indicate significant differences ($p < 0.001$) in both cases. We observe that not all methods significantly improve over Random Search with respect to final performance, i.e., EHVI and SMS-EGO fail to do so. Especially with respect to intermediate performance (25% of optimization budget), Random x4 outperforms all competitors (evaluating 4 times more configurations due to its parallelism). However, with respect to final performance, MEGO, ParEGO and MIES yield similar performance catching up to Random x4. We conclude that, in general, advanced multi-objective HPO methods improve over Random Search but also want to highlight that optimizer performance strongly varies with respect to the different benchmark instances.




(a) Mean ranks of HPO methods. x-axis starts after 10% of the optimization budget has been used.

(b) Critical differences plot for differences in ranks of HPO methods after 25% of optimization budget.

(c) Critical differences plot for differences in ranks of HPO methods after 100% of optimization budget.

Figure 4: Results of the YAHPO-MO multi-objective benchmark across 7 optimizers (25 Instances).

In total, both benchmarks described in this section took the equivalent of 138.7 CPU days using YAHPO Gym. We estimate that the YAHPO-SO benchmark, would take 15.34 CPU **years** when running real benchmarks, while our benchmark using YAHPO Gym took only 388 CPU **hours**, essentially speeding up evaluation by a factor of $\sim 350$.

## 7 Conclusions, Limitations and Broader Impact

We present YAHPO Gym, a multi-fidelity, multi-objective benchmark for HPO. Our benchmark is based on surrogates, which strike a favorable trade-off between faithfulness and efficiency, which we demonstrate in various experiments throughout our paper before conducting a large scale benchmark of modern single- and multi-objective optimizers. An as of yet under-explored domain are asynchronous optimization algorithms, which have recently gained popularity [31]. This has been studied in surrogate-based benchmarks by predicting runtimes and pausing the objective function for the predicted runtime, lowering computational demand for benchmarks but leading to a large waiting time [15]. In future work we plan on introducing faster-than-real time asynchronous benchmarking based on predicted runtimes.

**Limitations**. YAHPO Gym is based on surrogate models and therefore heavily relies on the faithfulness of those models in order to allow for valid conclusions. We have comprehensively evaluated surrogate models and provide a detailed report of performance metrics, hoping to demonstrate the faithfulness of our surrogates, but can only do so to a certain degree. We are furthermore aware that the real HPO problems modeled in our surrogates are in fact stochastic, and results can vary depending on randomness of the fitting procedure, data splits or initialization. We therefore provide a set of 'noisy' surrogate models that intend to model the stochasticity of the problems using an ensemble of neural networks, but simultaneously allow for full control of the stochastic process by using random seeds.

**Broader Impact**. This manuscript presents a set of surrogate-based benchmarks for HPO. As such, our work does not have direct implications on society or individuals, but can lead to such indirectly if new methods are developed based on it. We would like to emphasize the possible **societal & environmental benefits**. First, we hope our benchmarks can improve the state of benchmarking in hyperparameter optimization contexts, leading to better tracking of progress in the discipline. Second, and more important, we hope that experiments based on YAHPO Gym can drastically reduce **computational cost** of hyperparameter optimization experiments. This type of experiments is usually extremely expensive, if real experiments are run for the evaluation of each HPC, which can be sped up by large factors if cheap approximations through surrogates are available.

## 8 Reproducibility 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]

   (b) Did you describe the limitations of your work? [Yes] See Section 7

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 7

   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results…

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments…

   (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 version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] The full code for experiments, figures and table can be obtained from the following GitHub repositories:

      i. Software: `https://github.com/slds-lmu/yahpo_gym`

      ii. Documentation: `https://slds-lmu.github.io/yahpo_gym/`

      iii. Surrogates & Search Spaces: `https://github.com/slds-lmu/yahpo_data`

      iv. Code for Results: `https://slds-lmu.github.io/yahpo_exps`

   (b) Did you include the raw results of running the given instructions on the given code and data? [Yes] We make the full data used to train our surrogates available at `https://syncandshare.lrz.de/getlink/fiCMkzqj1bv1LfCUyvZKmLvd/`

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [Yes] See `https://slds-lmu.github.io/yahpo_exps`

   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [Yes]

   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes] See **??** for search spaces, the code repository as well as the software repository for further fixed hyperparameters

   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes] Yes, this is explicitly guaranteed by our software.

   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes] Partially, see sections throughout the supplementary material.

   (h) Did you use the same evaluation protocol for the methods being compared? [Yes] Yes

   (i) Did you compare performance over time? [Yes] Anytime performances are reported in all relevant figures throughout the paper.

(j) Did you perform multiple runs of your experiments and report random seeds? [Yes] We perform 30 replications for each experiments. Random seeds can be obtained from the acompanying code.

(k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All figures reporting experimental results include error bars.

(l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes] Surrogate benchmarks

(m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We state the total computation as well as $CO_2$ equivalent in the respective section and briefly summarize here: Tuning and fitting surrogates required a total of 45 GPU-days (116 kg $CO_2$-equivalent) on NVIDIA DGX-A100 instances) while the main experiments require 138.7 CPU days across all replications (262 kg $CO_2$ equivalent). The tabular vs surrogate benchmark required 22 CPU-hours (2kg $CO_2$) equivalent.

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [Yes] We report tuning of surrogates in the supplementary material.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets…

(a) If your work uses existing assets, did you cite the creators? [Yes] Yes, throughout the paper and explicitly in **??** for datasets we base our surrogates on.

(b) Did you mention the license of the assets? [Yes] Yes, see **??**.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Yes, trained surrogates are available at `https://github.com/slds-lmu/yahpo_data`.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Data is meta-data about ML experiments and we do not consider any personal data. All used data is available via OSS Licenses and no consent was required.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Data is only metadata about ML experiments.

5. 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] No crowd sourcing.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No IRB was required.

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

## References


[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] Sebastian Pineda Arango, Hadi S. Jomaa, Martin Wistuba, and Josif Grabocka. HPO-B: A Large-Scale Reproducible Benchmark for Black-Box HPO based on OpenML. *arXiv:2106.06257 [cs]*, 2021.

[3] ONNX authors. ONNX. `https://github.com/onnx/onnx`, 2022.

[4] Noor Awad, Neeratyoy Mallik, and Frank Hutter. DEHB: Evolutionary Hyberband for Scalable, Robust and Efficient Hyperparameter Optimization. *arXiv:2105.09821 [cs]*, 2021.

[5] James Bergstra, Brent Komer, Chris Eliasmith, and David Warde-Farley. Preliminary Evaluation of Hyperopt Algorithms on HPOLib. In *ICML Workshop on Automatic Machine Learning*, 2014.

[6] Martin Binder, Julia Moosbauer, Janek Thomas, and Bernd Bischl. Multi-objective Hyperparameter Tuning and Feature Selection Using Filter Ensembles. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 471479, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, Difan Deng, and Marius Lindauer. Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges. *arXiv:2107.05847 [cs, stat]*, July 2021.

[8] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[9] Mostafa Dehghani, Yi Tay, Alexey A. Gritsenko, Zhe Zhao, Neil Houlsby, Fernando Diaz, Donald Metzler, and Oriol Vinyals. The Benchmark Lottery. *arXiv:2107.07002 [cs]*, 2021.

[10] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters. In *NIPS Workshop on Bayesian Optimization in Theory and Practice*, 2013.

[11] Katharina Eggensperger, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Efficient Benchmarking of Hyperparameter Optimizers via Surrogates. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1114–1120, Austin, Texas, 2015. AAAI Press.

[12] Katharina Eggensperger, Marius Lindauer, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Efficient Benchmarking of Algorithm Configurators via Model-based Surrogates. *Machine Learning*, 107(1):15–41, 2018.

[13] Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, René Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. Hpobench: A collection of reproducible multi-fidelity benchmark problems for hpo, 2021.

[14] Michael T. M. Emmerich. Single- and multi-objective evolutionary design optimization assisted by Gaussian random field metamodels. *PhD Dissertation*, 2005.

[15] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, July 2018.

[16] M. Feurer, T. Springenberg, and F. Hutter. Initializing Bayesian Hyperparameter Optimization via Meta-learning. In B. Bonet and S. Koenig, editors, *Proceedings of the Twenty-Ninth National Conference on Artificial Intelligence (AAAI15)*, volume 15, pages 1128–1135. AAAI Press, 2015.

[17] Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. Practical Transfer Learning for Bayesian Optimization. *arXiv:1802.02219 [cs, stat]*, 2021.

[18] Pieter Gijsbers, Florian Pfisterer, Jan N. van Rijn, Bernd Bischl, and Joaquin Vanschoren. Meta-Learning for Symbolic Hyperparameter Defaults. *arXiv:2106.05767 [cs, stat]*, 2021.

[19] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34, 2021.

[20] J. Guerrero-Viu, S. Hauns, S. Izquierdo, G. Miotto, S. Schrodi, A. Biedenkapp, T. Elsken, D. Deng, M Lindauer, and F. Hutter. Bag of Baselines for Multi-objective Joint Neural Architecture Search and Hyperparameter Optimization. arXiv:2105.01015 [cs.LG], 2021.

[21] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

[22] Daniel Horn and Bernd Bischl. Multi-objective Parameter Configuration of Machine Learning Algorithms using Model-based Optimization. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

[23] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-based Optimization for General Algorithm Configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 507–523, Berlin, Heidelberg, 2011. Springer.

[24] S. Jeong and S. Obayashi. Efficient global optimization (EGO) for multi-objective problem and data mining. In *2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2138–2145, 2005.

[25] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabas Poczos. Multi-fidelity Bayesian Optimisation with Continuous Approximations. *arXiv:1703.06240 [stat]*, 2017.

[26] John G. Kemeny and James Laurie Snell. *Mathematical Models in the Social Sciences*. MIT Press, Cambridge, MA, USA, 1972.

[27] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536. PMLR, 2017.

[28] Aaron Klein, Zhenwen Dai, Frank Hutter, Neil Lawrence, and Javier Gonzalez. Meta-surrogate benchmarking for hyperparameter optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[29] J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[30] N. Lavesson and P. Davidsson. Quantifying the impact of learning algorithm parameter tuning. In *Proc. of AAAI*, volume 6, pages 395–400, 2006.

[31] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Bentzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A System for Massively Parallel Hyperparameter Tuning. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 230–246, 2020.

[32] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

[33] Rui Li, Michael T.M. Emmerich, Jeroen Eggermont, Thomas Bäck, M. Schütz, J. Dijkstra, and J.H.C. Reiber. Mixed integer evolution strategies for parameter optimization. *Evolutionary Computation*, 21(1):2964, 2013.

[34] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, J. Marben, P. Müller, and F. Hutter. BOAH: A Tool Suite for Multi-fidelity Bayesian Optimization & Analysis of Hyperparameters. arXiv:1908.06756 [cs.LG], 2019.

[35] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *arXiv:2109.09831 [cs, stat]*, 2022. arXiv: 2109.09831.

[36] Marius Lindauer, Matthias Feurer, Katharina Eggensperger, André Biedenkapp, and Frank Hutter. Towards assessing the impact of bayesian optimization's own hyperparameters. *arXiv preprint arXiv:1908.06674*, 2019.

[37] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable Architecture Search. In *Proceedings of the International Conference on Learning Representations*, 2019.

[38] Olaf Mersmann, Heike Trautmann, Boris Naujoks, and Claus Weihs. Benchmarking Evolutionary Multiobjective Optimization Algorithms. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[39] Julia Moosbauer, Martin Binder, Lennart Schneider, Florian Pfisterer, Marc Becker, Michel Lang, Lars Kotthoff, and Bernd Bischl. Automated benchmark-driven design and explanation of hyperparameter optimizers, 2021.

[40] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1965.

[41] Christina Nießl, Moritz Herrmann, Chiara Wiedemann, Giuseppe Casalicchio, and Anne-Laure Boulesteix. Over-optimism in benchmark studies and the multiplicity of design and analysis options when interpreting their results. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1441, 2021.

[42] Takeru Ohta and Hiroyuki Vincent Yamazaki. Kurobako. `https://github.com/optuna/kurobako`, 2022.

[43] Maryam Parsa, John P. Mitchell, Catherine D. Schuman, Robert M. Patton, Thomas E. Potok, and Kaushik Roy. Bayesian Multi-objective Hyperparameter Optimization for Accurate, Fast, and Efficient Neural Network Accelerator Design. *Frontiers in Neuroscience*, 14:667, 2020.

[44] Florian Pfisterer, Jan N. van Rijn, Philipp Probst, Andreas C. Müller, and Bernd Bischl. Learning Multiple Defaults for Machine Learning Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, pages 241–242, New York, NY, USA, 2021. Association for Computing Machinery.

[45] Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection. In Günter Rudolph, Thomas Jansen, Nicola Beume, Simon Lucas, and Carlo Poloni, editors, *Parallel Problem Solving from Nature PPSN X*, Lecture Notes in Computer Science, pages 784–794. Springer, 2008.

[46] R. Schmucker, M. Donini, V. Perrone, M. B. Zafar, and C. Archambeaut. Multi-objective Multi-fidelity Hyperparameter Optimization with Application to Fairness. In *NeurIPS Workshop on Meta-Learning*, volume 2, 2020.

[47] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. Lassobench: A high-dimensional hyperparameter optimization benchmark suite for lasso. *arXiv preprint arXiv:2111.02790*, 2021.

[48] J. Siems, L. Zimmer, A. Zela, J. Lukasik, M. Keuper, and F. Hutter. NAS-Bench-301 and the Case for Surrogate Benchmarks for Neural Architecture Search. arXiv:2008.09777 [cs.LG], 2020.

[49] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[50] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-Thaw Bayesian Optimization. *arXiv:1406.3896 [cs, stat]*, 2014.

[51] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013.

[52] Ryan Turner. Uber. bayesopt benchmark. `https://github.com/uber/bayesmark`, 2022.

[53] Joaquin Vanschoren. Meta-Learning. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automated Machine Learning: Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 35–61. Springer International Publishing, Cham, 2019.

[54] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luís Torgo. OpenML: Networked Science in Machine Learning. *SIGKDD Explor.*, 15(2):49–60, 2013.

[55] Michael Volpp, Lukas P Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-learning Acquisition Functions for Transfer Learning in Bayesian Optimization. *International Conference on Learning Representations*, 2020.

[56] C. White, W. Neiswanger, and Y. Savani. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search. arXiv:1910.11858 [cs.LG], 2019.

[57] Martin Wistuba, N. Schilling, and L. Schmidt-Thieme. Learning Hyperparameter Optimization Initializations. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2015.