HPO-B: A Large-Scale Reproducible Benchmark for Black-Box HPO based on OpenML

Sebastian Pineda Arango[*]
University of Freiburg
pineda@cs.uni-freiburg.de

Hadi S. Jomaa[*]
University of Hildesheim
hsjomaa@ismll.uni-hildesheim.de

Martin Wistuba
Amazon Research
marwistu@amazon.com

Josif Grabocka
University of Freiburg
grabocka@cs.uni-freiburg.de

Abstract

Hyperparameter optimization (HPO) is a core problem for the machine learning community and remains largely unsolved due to the significant computational resources required to evaluate hyperparameter configurations. As a result, a series of recent related works have focused on the direction of transfer learning for quickly fine-tuning hyperparameters on a dataset. Unfortunately, the community does not have a common large-scale benchmark for comparing HPO algorithms. Instead, the *de facto* practice consists of empirical protocols on arbitrary small-scale meta-datasets that vary inconsistently across publications, making reproducibility a challenge. To resolve this major bottleneck and enable a fair and fast comparison of black-box HPO methods on a level playing field, we propose **HPO-B**, a new large-scale benchmark in the form of a collection of meta-datasets. Our benchmark is assembled and preprocessed from the OpenML repository and consists of 176 search spaces (algorithms) evaluated sparsely on 196 datasets with a total of 6.4 million hyperparameter evaluations. For ensuring reproducibility on our benchmark, we detail explicit experimental protocols, splits, and evaluation measures for comparing methods for both non-transfer, as well as, transfer learning HPO.

1 Introduction

Hyperparameter Optimization (HPO) is arguably the major open challenge for the machine learning community due to the expensive computational resources demanded to evaluate configurations. As a result, HPO and its broader umbrella research area, AutoML, have drawn particular interest over the past decade [2, 14, 26, 27]. Black-box HPO is a specific sub-problem that focuses on the case where the function to be optimized (e.g. the generalization performance of an algorithm) is unknown, non-differentiable with respect to the hyperparameters, and intermediate evaluation proxies are not computable (opposed to gray-box HPO [19] which accesses intermediate performance measurements).

Although black-box HPO is a core problem, existing solutions based on parametric surrogate models for estimating the performance of a configuration overfit the limited number of evaluated configurations. As a result, the AutoML community has recently invested efforts in resolving the sample-inefficiency of parametric surrogates via meta- and transfer-learning [10, 15, 23, 24, 29, 31, 34].

[*]Equal contribution

Submitted to the 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks. Do not distribute.

Unfortunately, despite the promising potential of transfer-learning in black-box HPO, the impact of such algorithms is hindered by their poor experimental reproducibility. Our personal prior research experience, as well as the feedback from the community, highlight that reproducing and generalizing the results of transfer-learning HPO methods is challenging. In essence, the problem arises when the results of a well-performing method in the experimental protocol of a publication either can not be replicated; or when the method underperforms in a slightly different empirical protocol. We believe that a way of resolving this negative *impasse* is to propose a new public large-scale benchmark for comparing HPO methods, where the exact training/validation/test splits of the meta-datasets, the exact evaluation protocol, and the performance measures are well-specified. The strategy of adopting benchmarks is a trend in related areas, such as in computer vision [7], or NAS [8, 36].

In this perspective, we present **HPO-B**[2], the largest public benchmark of meta-datasets for black-box HPO containing 6.4M hyperparameter evaluations across 176 search spaces (algorithms) and on 196 datasets in total. The collection is derived from the raw data of OpenML [28], but underwent an extensive process of cleaning, preprocessing and organization (Section 5). Additionally, we offer off-the-shelf ready variants of the benchmark that are adapted for both non-transfer, as well as transfer HPO experiments, together with the respective evaluation protocols (Section 6). This large, diverse, yet plug-and-play benchmark can significantly boost future research in black-box HPO.

2 Terminology

To help the reader navigate through our paper, we present the compact thesaurus of Table 1 for defining the vernacular of the HPO community.

Term	Definition
Configuration	Specific settings/values of hyperparameters
Search space	The domain of a configuration: scale and range of each hyperparameter's values
Response	The performance of an algorithm given a configuration and dataset
Surrogate	A (typically parametric) function that approximates the response
Seed	Set of initial configurations used to fit the initial surrogate model
Black-box	The response is an unknown and non-differentiable function of a configuration
Task	An HPO problem given a search space and a dataset
Evaluation	The measured response of a configuration on a dataset
Trial	An evaluation on a task during the HPO procedure
Meta-dataset	Collection of *recorded* evaluations from different tasks on a search space
Meta-instance	An evaluation in the meta-dataset for one of the tasks
Meta-feature	Descriptive attributes of a dataset
Source tasks	In a meta- or transfer-learning setup refers to the *known* tasks we *train from*
Target tasks	In a meta- or transfer-learning setup refers to the *new* tasks we *test on*
Benchmark	**New definition:** Collection of meta-datasets from different search spaces

Table 1: A thesaurus of the common HPO terminology used throughout this paper

3 Related Work

Non-transfer black-box HPO: The mainstream paradigm in HPO relies on surrogates to estimate the performance of hyperparameter configurations. For example, [2] were the first to propose Gaussian Processes (GP) as surrogates. The same authors also propose a Tree Parzen Estimator (TPE) for computing the non-parametric densities of the hyperparameters given the observed performances. Both approaches achieve a considerable lift over random [3] and manual search. To address the cubic run-time complexity of GPs concerning the number of evaluated configurations, DNGO [26] trains neural networks for generating adaptive basis functions of hyperparameters, in combination with a Bayesian linear regressor that models uncertainty. Alternatively, SMAC [14] represents the surrogate as a random forest, and BOHAMIANN [27] employs Bayesian neural networks instead of plain neural networks to estimate the uncertainty of a configuration's performance. For an extensive study

[2]The benchmark is publicly available at `https://github.com/releaunifreiburg/HPO-B`

on non-transfer Bayesian optimization techniques for HPO, we refer the readers to [5, 25] that study the impact of the underlying assumptions associated with black-box HPO algorithms.

Transfer black-box HPO: To expedite HPO, it is important to leverage information from existing evaluations of configurations from prior tasks. A common approach is to capture the similarity between datasets using meta-features (i.e. descriptive dataset characteristics). Meta-features have been used as a warm-start initialization technique [11, 16], or as part of the surrogate directly [1]. Transfer learning is also explored through the weighted combination of surrogates, such as in TST-R [34], RGPE [10], and TAF-R [35]. Another direction is learning a shared surrogate across tasks. ABLR optimizes a shared hyperparameter embedding with separate Bayesian linear regressors per task [20], while GCP [23] maps the hyperparameter response to a shared distribution with a Gaussian Copula process. Furthermore, FSBO [31] meta-learns a deep-kernel Gaussian Process surrogate, whereas DMFBS incorporates the dataset context through end-to-end meta-feature networks [16].

Meta-datasets: The work by Wistuba et al. [33] popularised the usage of meta-dataset benchmarks with pre-computed evaluations for the hyperparameters of SVM (288 configurations) and Adaboost (108 configurations) on 50 datasets; a benchmark that inspired multiple follow-up works [10, 30]. Existing attempts to provide HPO benchmarks deal only with the non-transfer black-box HPO setup [9]. As they contain results for one or very few datasets per search space, they cannot be used for the evaluation of transfer black-box HPO methods. Nevertheless, there is a trend in using evaluations of search spaces from the OpenML repository [12], which contains evaluations reported by an open community, as well as large-scale experiments contributed by specific research labs [4, 18]. However, the choice of OpenML search spaces in publications is ad-hoc: one related work uses SVM and XGBoost [20], a second uses GLMNet and SVM [31], while a third paper uses XGBoost, Random Forest and SVM [21]. We assess that the community *i)* inconsistently cherry-picks (assuming *bona fides*) search spaces, with *ii)* arbitrary train/validation/test splits of the tasks within the meta-dataset, and *iii)* inconsistent preprocessing of hyperparameters and responses. In our experiments, we observed that existing methods do not generalize well on new meta-datasets (Section 7).

Our Novelty: As a remedy, we propose a novel benchmark derived from OpenML [12], that resolves the existing reproducibility issues of existing non-transfer and transfer black-box HPO methods, by ensuring a fairly-reproducible empirical protocol. The contributions of our benchmark are multi-fold. First of all, we remove the confounding factors induced by different meta-dataset preprocessing pipelines (e.g. hyperparameter scaling and transformations, missing value imputations, one-hot encodings, etc.). Secondly, we provide a specified collection of search spaces, with specified datasets and evaluations. Furthermore, for transfer learning HPO methods, we also provide pre-defined training/validation/testing splits of tasks. For experiments on the test tasks, we additionally provide 5 seeds (i.e. 5 sets of initial hyperparameters to fit the initial surrogate) with 5 hyperparameter configurations, each. We also highlight recommended empirical measures for comparing HPO methods and assessing their statistical significance in Section 6. In that manner, the results of different papers that use our benchmark can be compared directly without fearing the confounding factors. Table 2 presents a summary of the descriptive statistics of meta-datasets from prior literature. To the best of our awareness, the proposed benchmark is also richer (in the number of search spaces and their dimensionality) and larger (in the number of evaluations) than all the prior protocols.

Paper	Venue/Year	# Search Spaces	# Datasets	# HPs	# Evals.
[1]	ICML '13	1	29	2	3K
[33]	DSAA '15	2	50	2, 4	20K
[11]	AAAI '15	3	57	4, 5	93K
[32]	ECML-PKDD '15	17	59	1-7	1.3M
[20]	NeurIPS '18	2	30	4, 10	655K
[23]	ICML '20	4	26	6, 9	343K
[16]	DMKD '21	1	120	7	414K
[31]	ICLR '21	3	80	2, 4	864K
Our HPO-B-v1	-	176	196	1-53	6.39M
Our HPO-B-v2/-v3	-	16	101	2-18	6.34M

Table 2: Summary statistics for various meta-datasets considered in prior works.

4 A Brief Explanation of Bayesian Optimization Concepts

As we often refer to HPO methods, in this section we present a brief coverage of Bayesian optimization as the most popular HPO method for black-box optimization. HPO aims at minimizing the function $f : \mathcal{X} \rightarrow \mathbb{R}$ which maps each hyperparameter configuration $\mathbf{x} \in \mathcal{X}$ to the validation loss obtained when training the machine learning model using $\mathbf{x}$. Bayesian optimization keeps track of all evaluated hyperparameter configurations in a history $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_i$, where $y_i \sim \mathcal{N}(f(\mathbf{x}_i), \sigma_n^2)$ is the (noisy) response. A probabilistic model, the so-called surrogate model, is used to approximate the behavior of the response function. Gaussian Processes are a common choice for the surrogate model [22].

Bayesian optimization is an iterative process that alternates between updating the surrogate model as described above and selecting the next hyperparameter configuration. The latter is done by finding the configuration which maximizes an acquisition function, which scores each feasible hyperparameter configuration using the surrogate model by finding a trade-off between exploration and exploitation. Arguably, the most popular acquisition function is the Expected Improvement [17].

The efficiency of Bayesian optimization depends on the surrogate model's ability to approximate the response function. However, this is a challenging task since every optimization starts with no or little knowledge about the response function. To overcome this cold-start problem, transfer methods have been proposed, where the transfer scenario slightly changes the problem definition. The main objective remains finding the configuration $\mathbf{x}_\star \in \mathcal{X}^{(\tilde{s})}$ which optimizes the target response function $f^{(\tilde{s},\tilde{t})}$ for a given search space $\mathcal{X}^{(\tilde{s})}$. The knowledge about the target response is now denoted by $\mathcal{D}^{(\tilde{s},\tilde{t})}$. Additionally, side information, the so-called *meta-dataset*, for S-many search spaces and T-many datasets, $\mathcal{D} = \bigcup_{s=1..S,t=1..T} \mathcal{D}^{(s,t)}$, is available. Here, $\mathcal{D}^{(s,t)} = \{(\mathbf{x}_i^{(s,t)}, y_i^{(s,t)})\}_i$, $\mathbf{x}_i^{(s,t)} \in \mathcal{X}^{(s)}$, and $y_i^{(s,t)}$ is the noisy observation of the source response function $f_i^{(s,t)} = f^{(t)}(\mathbf{x}_i^{(s,t)})$.

5 Benchmark Description

The benchmark HPO-B is a collection of meta-datasets collected from OpenML [12] with a diverse set of search spaces. We present three different versions of the meta-data set, as follows:

- **HPO-B-v1:** The raw benchmark of all 176 meta-datasets;
- **HPO-B-v2:** Subset of 16 meta-datasets with the most frequent search spaces;
- **HPO-B-v3:** Split of HPO-B-v2 into training, validation and testing.

When assembling the benchmark HPO-B-v1 we noticed that most of the evaluations are reported for a handful of popular search spaces, in particular, we noticed that 9% of the top meta-datasets include 99.3% of the evaluations. As a result, we created a second version HPO-B-v2 that includes only the frequent meta-datasets that have at least 10 datasets with at least 100 evaluations per dataset (Section 5.1). Furthermore, as we clarified in Section 3 a major reproducibility issue of the related work on transfer HPO is the lack of clear training, validation, and test splits for the meta-datasets. To resolve this issue, we additionally created HPO-B-v3 as a derivation of HPO-B-v2 with pre-defined splits of the training, validation, and testing tasks for every meta-dataset, in addition to providing initial configurations (seeds) for the test tasks. The three versions were designed to fulfill concrete purposes with regards to different types of HPO methods. For non-transfer black-box HPO methods, we recommend using HPO-B-v2 which offers a large pool of HPO tasks. Naturally, for transfer HPO tasks we recommend using HPO-B-v3 where meta-datasets are split into training, validation, and testing. We still are releasing the large HPO-B-v1 benchmark to anticipate next-generation methods for heterogeneous transfer learning techniques that meta-learn surrogates across different search spaces, where all 176 meta-datasets might be useful despite most of them having few evaluations.

Concretely, HPO-B-v3 contains the set of filtered search spaces of HPO-B-v2, which are specially split into *four* sets: meta-train, meta-validation. meta-test and an augmented version of the meta-train dataset. Every split contains different datasets from the same search space. We distributed the datasets per search space as 80% of the datasets to meta-train, 10% to meta-validation, and 10% to meta-test, respectively. A special, augmented version of the meta-train is created by adding all other search space evaluations from HPO-B-v1 that are not part of HPO-B-v3. On the other hand, in HPO-B-v3 we also provide seeds for initializing the HPO. They are presented as five different sets of five initial configurations to be used by a particular HPO method. By providing five different seeds we decrease the random effect of the specific initial configurations. To ease the comparison among HPO methods, we suggest using the recommended initial configurations for testing. Although, we admit that some algorithms proposing novel warm-starting strategies might need to bypass the recommended initial configurations.

5.1 Benchmark summary

The created benchmark contains 6,394,555 total evaluations across 176 search spaces that are sparsely evaluated on 196 datasets. By accounting for the search spaces that comply with our filtering criteria (at least 10 datasets with 100 evaluations), we obtain HPO-B-v2 with 16 different search spaces and 6,347,916 evaluations on 101 datasets. Notice that the benchmark does not include evaluations for all datasets in every search space. The number of dimensions, datasets, and evaluations per search space is listed in Table 3. An additional description of the rest of all the search spaces in HPO-B-v1 is presented in the Appendix. In addition, Table 3 shows the description of the meta-dataset splits according to the HPO-B-v3.

Search Space	ID	#HPs	Meta-Train		Meta-Validation		Meta-Test	
			#Evals.	#DS	#Evals.	#DS	#Evals.	#DS
rpart.preproc(16)	4796	3	10694	36	1198	4	1200	4
svm (6)	5527	8	385115	51	196213	6	354316	6
rpart (29)	5636	6	503439	54	184204	7	339301	6
rpart (31)	5859	6	58809	56	17248	7	21060	6
glmnet (4)	5860	2	3100	27	598	3	857	3
svm (7)	5891	8	44091	51	13008	6	17293	6
xgboost (4)	5906	16	2289	24	584	3	513	2
ranger (9)	5965	10	414678	60	73006	7	83597	7
ranger (5)	5970	2	68300	55	18511	7	19023	6
xgboost (6)	5971	16	44401	52	11492	6	19637	6
glmnet (11)	6766	2	599056	51	210298	6	310114	6
xgboost (9)	6767	18	491497	52	211498	7	299709	6
ranger (13)	6794	10	591831	52	230100	6	406145	6
ranger (15)	7607	9	18686	58	4203	7	5028	7
ranger (16)	7609	9	41631	59	8215	7	9689	7
ranger (7)	5889	6	1433	20	410	2	598	2

Table 3: Description of the search spaces in HPO-B-v3; "#HPs" stands for the number of hyperparameters, "#Evals." for the number of evaluations in a search space, while "#DS" for the number of datasets across which the evaluations are collected. The search spaces are named with the respective OpenML version number (in parenthesis), and their original names are preceded by *mlr.classif.*

5.2 Preprocessing

The OpenML-Python API [13] was used to download the experiment data from OpenML [12]. We have collected all evaluations (referred to as runs in OpenML) tagged with `Verified_Supervised_Classification` available until April 15, 2021.

We processed the raw data as follows. While the hyperparameter configuration was directly available for many evaluations, some of them had to be parsed from WEKA arguments (e.g. `weka.filters.unsupervised.attribute.RandomProjection -P 16.0 -R 42 -D Sparse1`). A small percentage (<0.001%) of these were too complex in structure to be automatically parsed, so they were discarded. Duplicate responses for the same hyperparameter configuration have been resolved by keeping only one random response. Finally, all tasks with fewer than five observations were also discarded.

All categorical hyperparameters were one-hot encoded, taking into account all categories that occur in the different datasets for a search space. Missing values have been replaced with zeros and the corresponding missing indicator (a new feature) has been set to one. Hyperparameters that had the same value for all configurations in a search space were dropped. We manually decided which hyperparameters required log-scaling by inspecting the distributions of each hyperparameter in each space (considerable manual effort). Finally, the hyperparameter ranges were scaled to $[0, 1]$.

5.3 Benchmark JSON schema

The benchmark is offered as easily accessible JSON files. The first-level key of each JSON schema corresponds to the search space ID, whereas the second-level key specifies the dataset ID. By accessing the JSON schema with the search space s and the dataset t, we obtain the meta-dataset $\mathcal{D}^{(s,t)} = \{(\mathbf{x}_i^{(s,t)}, y_i^{(s,t)})\}_i$, $\mathbf{x}_i^{(s,t)} \in \mathcal{X}^{(s)}$. The meta-dataset exhibits the following structure, where N denotes the number of evaluations available for the specific task:

$$\texttt{\{search\_space\_ID: \{dataset\_ID:\{}\mathbf{X}\texttt{:[[}\mathbf{x}_1\texttt{],...,[}\mathbf{x}_N\texttt{]], }y\texttt{:[[}y_1\texttt{],...,[}y_N\texttt{]]\}\}\}}$$

The initialization seeds are similarly provided as a JSON schema, where the third-level subschema has 5 keys whose values are the indices of the samples to use as initial configurations.

6 Recommended Experimental Protocol

One of the primary purposes of HPO-B is to standardize and facilitate the comparison between HPO techniques on *a level playing field*. In this section, we provide two specific recommendations: which benchmark to use for a type of algorithm and what metrics to use for comparing results.

Evaluation Metrics We define the average normalized regret at trial e (a.k.a. average distance to the minimum) as $\min_{x \in \mathcal{X}_e^{(s,t)}} \left(f^{(s,t)}(x) - y^*_{\text{min}}\right) / \left(y^*_{\text{max}} - y^*_{\text{min}}\right)$ with $\mathcal{X}_e^{(s,t)}$ as the set of hyperparameters that have been selected by a HPO method up to trial e, with y^*_{min} and y^*_{max} as the best and worst responses, respectively. The average rank represents the mean across tasks of the ranks of competing methods computed using the test accuracies of the best configuration until the e-th trial. Results across different search spaces are computed by a simple mean over the search-space-specific results.

Non-Transfer Black-Box HPO Methods should be compared on all the tasks in HPO-B-v2 and for each of the five initial configurations. The authors of future papers should report the normalized regret and the mean ranks for all trials from 1 to 100 (excluding the seeds). We recommend that the authors show both aggregated and per search-space (possibly moved to the appendix) mean regret and mean rank curves for trials ranging from 1 to 100. In other words, as many runs as the number of tasks for a given space times the number of initialization seeds. To assess the statistical significance of methods, we recommend that critical difference diagrams [6] be computed for the ranks of all runs @25, @50, and @100 trials.

Transfer Black-Box HPO Methods should be compared on the meta-data splits contained in HPO-B-v3. All competing methods should use exactly the evaluations of the provided meta-train datasets for meta- and transfer-learning their method, and tune the hyper-hyperparameters on the evaluations of the provided meta-validation datasets. In the end, the competing methods should be tested on the provided evaluations of the meta-test tasks. As our benchmark does not have pre-computed responses for all possible configurations in a space, the authors need to adapt their HPO acquisitions and suggest the next configuration only from the set of the pre-computed configurations for each specific meta-test task. Additionally, we recommend that the authors present (see details in the paragraph above) regret and rank plots, besides the critical difference diagrams @25, @50, and @100 trials. If a future transfer HPO method proposes a novel strategy for initializing configurations, for the sake of reproducibility we still recommend showing additional results with our initial configurations.

7 Experimental Results

The benchmark is intended to serve as a new standard for evaluating non-transfer and transfer black-box HPO methods. In the following, we will compare different methods according to our recommended protocol described in Section 6. This is intended to demonstrate the usefulness of our meta-dataset, while at the same time serving as an example for the aforementioned recommendations on comparing baselines and presenting results.

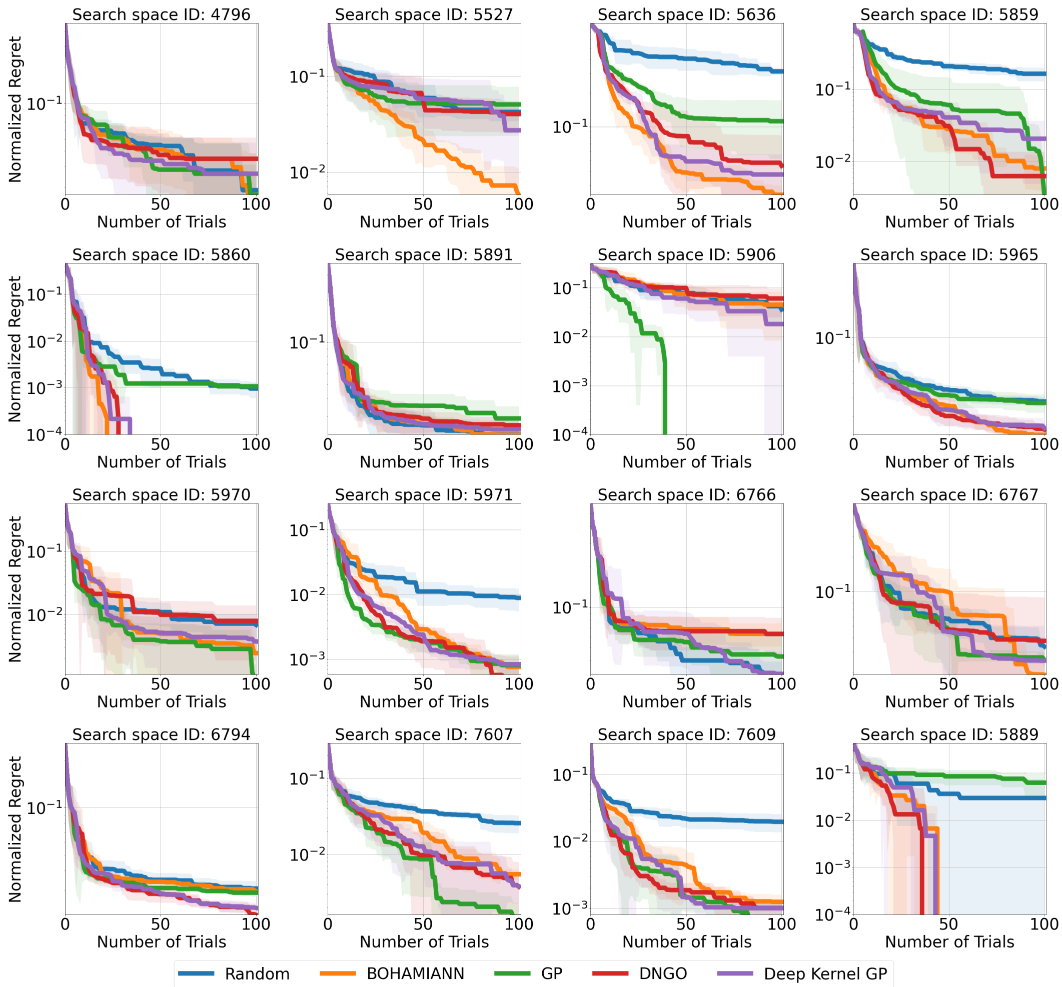

Figure 1: **Normalized regret** comparison of **non-transfer** black-box HPO methods on HPO-B-v2

7.1 Non-transfer Black-Box HPO

First, we compare Random Search, DNGO, BOHAMIANN, Gaussian Process (GP) with Matérn 3/2 kernel, and Deep Gaussian Process (FSBO [31] without pre-training) on HPO-B-v2 in the non-transfer scenario. As recommended by us earlier, in Figure 1 we report aggregated results for normalized regret, average rank, and critical difference plots. In addition, we report in Figure 2 the aggregated normalized regret per search space. The values in the figures for the number of trials equal to 0 correspond to the result after the five initialization steps. According to Figure 2, BOHAMIANN and Deep GP achieve comparable aggregated normalized regret across all search spaces, which suggests that both methods are equally well-suited for the tasks. The average rank and the critical difference plot paint a different picture, in which Deep GP and DNGO achieve better results. This discrepancy arises because each metric measures different performance aspects on different tasks, so it's important to report both. As can be seen in Figure 10, Deep GP achieves better results than the GP in most of the tasks, which leads to a better average ranking. However, as we can see in Figure 1, the regrets are observed at heterogeneous scales that can skew the overall averages. In some cases where BOHAMIANN outperforms Deep GP (e.g. search spaces 5527, 5859, and 5636), the difference in normalized regret is evident, due to the nature of the search space, whereas in cases where it is the other way around, however, the difference is only slightly less evident (e.g. search spaces 4796, 5906, and 7609). An important aspect of HPO is the choice of the surrogate function and acquisition. Figure 3 presents an ablation of typical combinations and shows the accuracy of the Boosted Tree as a surrogate.

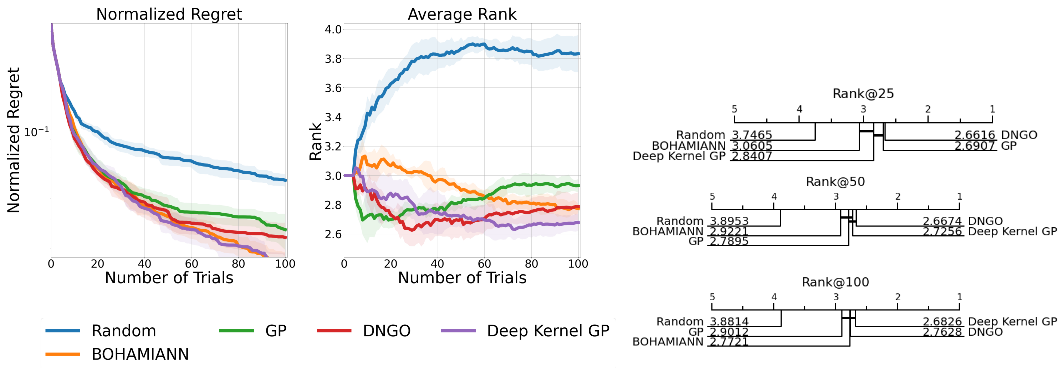

Figure 2: **Aggregated** comparisons of normalized regret and mean ranks across all search spaces for the **non-transfer** HPO methods on HPO-B-v2

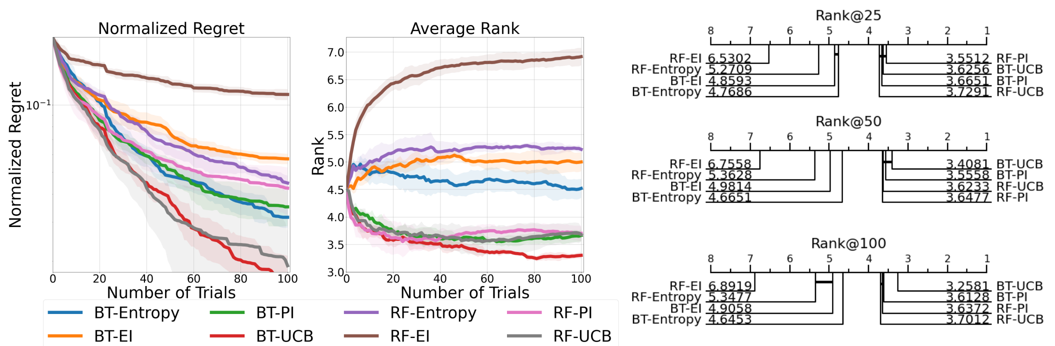

Figure 3: **Aggregated** comparisons of different surrogates and acquisition functions for **transfer** HPO methods on HPO-B-v2; BT stands for Boosted Trees, RF for Random Forests, EI for Expected Improvement, and UCB for Upper Confidence Bound.

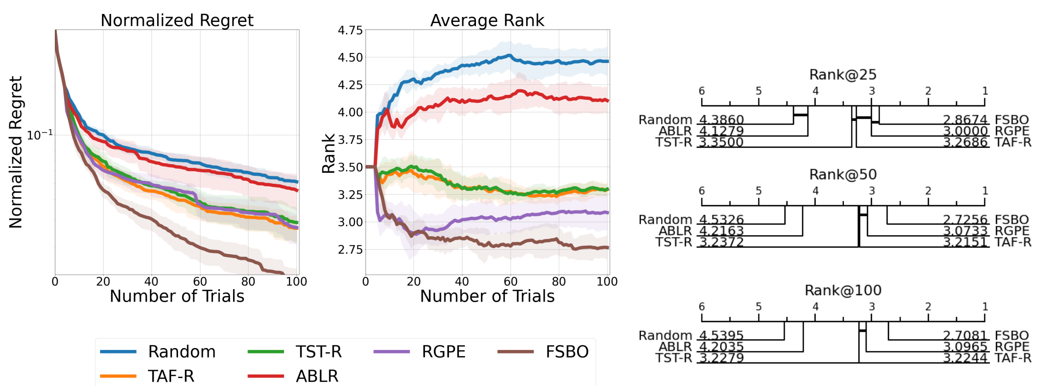

Figure 4: **Aggregated** comparisons of normalized regret and mean ranks across all search spaces for the **transfer learning** HPO methods on HPO-B-v3

7.2 Transfer Black-Box HPO

Finally, we compare RGPE [10], ABLR [20], TST-R [34], TAF-R [35], and FSBO [31] on HPO-B-v3 in the transfer scenario. All hyper-hyperparameters were optimized on the meta-validation datasets and we report results aggregated across all test search spaces in terms of normalized regret and average rank in Figure 4. The results per search space for normalized regret and average rank are given in Figure 5 and Figure 11, respectively. FSBO shows improvements over all the compared methods for the normalized regret metric and average rank metric. On the other hand, RGPE is seemingly performing similar to TST-R and TAF-R for the average regret, but performs significantly

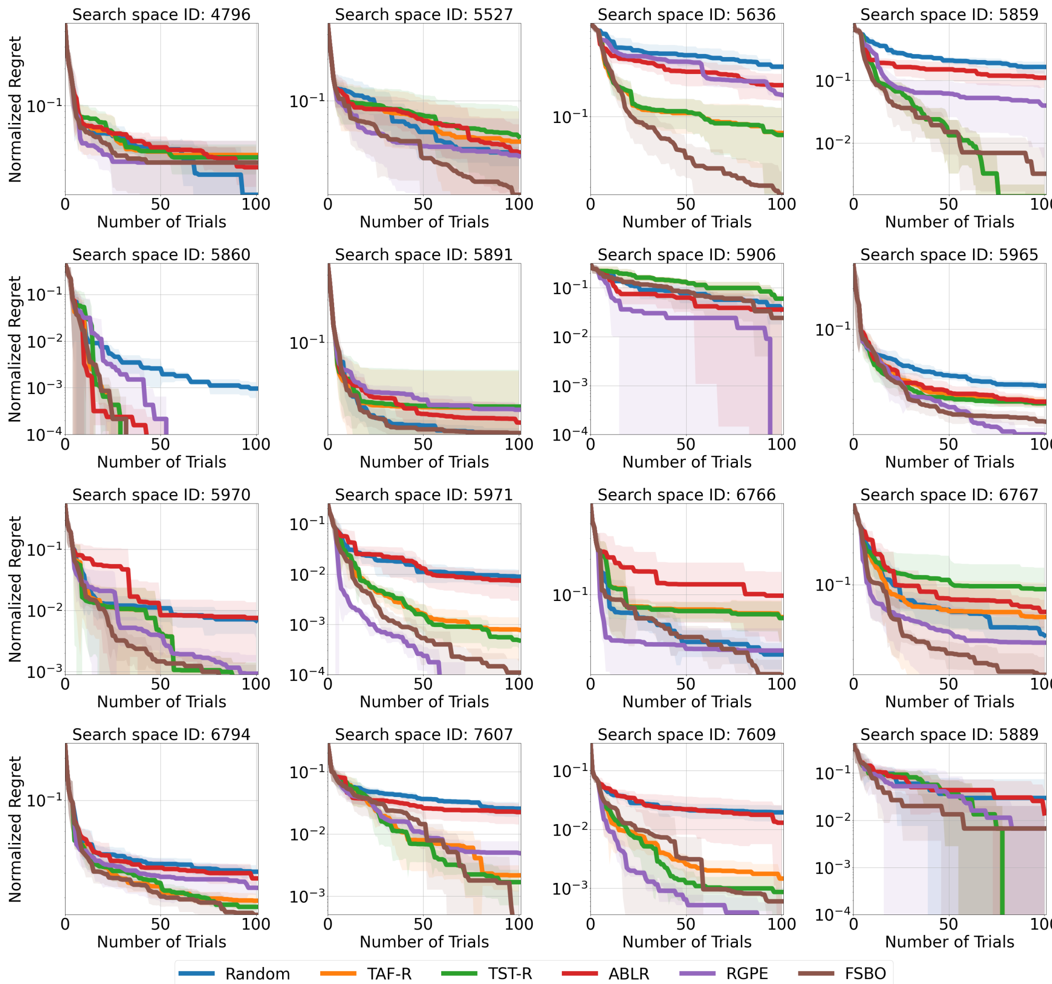

Figure 5: **Normalized regret** comparison of **transfer learning** HPO methods on HPO-B-v3

better for the average rank metric. The explanation is the same as for our last experiment and can mainly be traced back to the strong performance of RGPE in search spaces 5971 and 5906. Such behaviors strengthen our recommendations of Section 6 for showing results in terms of both the ranks and the normalized regrets, as well as the ranks' statistical significance.

7.3 Comparing Non-Transfer vs. Transfer Black-Box HPO

We provide a cumulative comparison of both non-transfer and transfer black-box methods in Figure 6, for demonstrating the benefit of transfer learning in HPO-B-v3. We see that the transfer methods (FSBO, RGPE, TST-R, TAF-R) achieve significantly better performances than the non-transfer techniques (GP, DNGO, BOHAMIANN, Deep Kernel GP). On the average rank plot and the associated Critical Difference diagrams, we notice that FSBO [31] achieves significantly better results than all baselines, followed by RGPE [10]. A detailed comparison of the ranks per search-space is presented in the supplementary material. In particular, the direct gain of transfer learning can observed by the dominance that FSBO has over *Deep Kernel GP*, considering that both use exactly the same surrogate model and the same acquisition function. In comparison, the deep kernel parameters in FSBO are initialized from the solution of a meta-learning optimization conducted on the meta-train tasks of HPO-B-v3 (transfer), while the parameters of *Deep Kernel GP* are initialized randomly (no transfer).

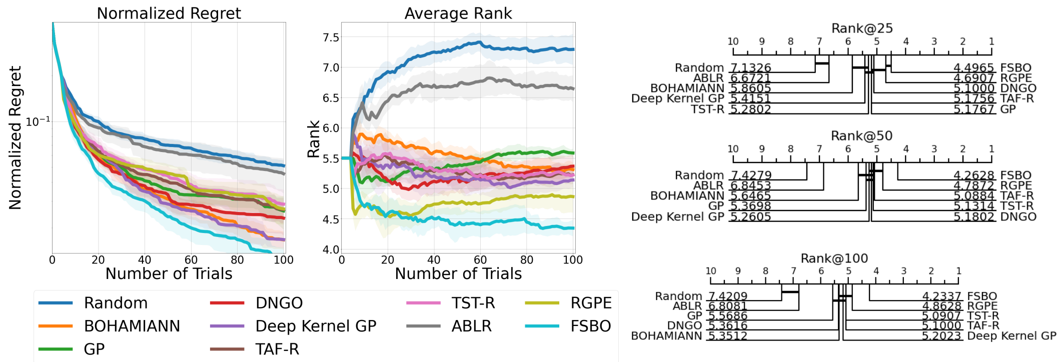

Figure 6: Comparisons of normalized regret and mean ranks across all search spaces for the **transfer learning** and **non-transfer** HPO methods on HPO-B-v3

8 Discussing the Limitations of HPO-B

HPO-B relies on OpenML evaluations that are not exhaustively computed for all the possible hyperparameter configurations of a search space. In that context, an acquisition function can suggest a next configuration only from the set of those configurations that were evaluated on a particular task. A possible way to tackle this limitation is fitting surrogate models on the evaluated configurations, and using the estimation of the surrogate as the response of a new configuration for which no actual evaluation exists. However, the choice of the surrogate model might add noisy/confounding effects to the evaluations, as there are open questions on the capacity of the surrogate model (i.e. do we need different surrogate complexities for different tasks/datasets?), or the choice of the loss function for training the surrogate. Moreover, as the majority of HPO-B tasks have an abundant number of evaluations (see Appendix E for details), it is highly likely that a well-performing configuration is already present in the set of existing evaluations.

Another limitation of HPO-B is that it only covers black-box HPO tasks, instead of other HPO problems, such as grey-box HPO, or pipeline optimization for AutoML libraries. In addition, HPO-B is restricted by the nature of search spaces found in OpenML, which contains evaluations for well-established machine learning algorithms for tabular data, but lacks state-of-the-art deep learning methods or tasks on image or text data.

9 Conclusions

Recent HPO and transfer-learning HPO papers inconsistently use different meta-datasets, arbitrary train/validation/test splits, as well as ad-hoc preprocessing, which makes it hard to reproduce the published results. To resolve this bottleneck, we propose HPO-B, a novel benchmark based on the OpenML repository, that contains meta-datasets from 176 search spaces, 196 datasets, and a total of 6.4 million evaluations. For promoting reproducibility at a *level playing field* we also provide initial configuration seeds, as well as predefined training, validation and testing splits. Our benchmark contains pre-processed meta-datasets and a clear set of HPO tasks and exact splits, therefore, it enables future benchmark results to be directly comparable. We believe our benchmark has the potential to become the *de facto* standard for experimentation in the realm of black-box HPO.

Acknowledgements

The research of Hadi S. Jomaa is co-funded by the industry project "IIP-Ecosphere: Next Level Ecosphere for Intelligent Industrial Production". Prof. Grabocka is also thankful to the Eva Mayr-Stihl Foundation for their generous research grant, as well as to the Ministry of Science, Research and the Arts of the German state of Baden-Württemberg for funding his professorship. In addition, we thank Arlind Kadra for his assistance in interfacing with the OpenML Python package.

References

[1] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *International conference on machine learning*, pages 199–207, 2013.

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, volume 24. Neural Information Processing Systems Foundation, 2011.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[4] Martin Binder, Florian Pfisterer, and Bernd Bischl. Collecting empirical data about hyperparameters for datadriven automl. In *ICML Workshop on Automated Machine Learning*, 2020.

[5] Alexander I Cowen-Rivers, Wenlong Lyu, Zhi Wang, Rasul Tutunov, Hao Jianye, Jun Wang, and Haitham Bou Ammar. Hebo: Heteroscedastic evolutionary bayesian optimisation. *arXiv preprint arXiv:2012.03826*, 2020. winning submission to the NeurIPS 2020 Black Box Optimisation Challenge.

[6] Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020.

[9] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, volume 10, page 3, 2013.

[10] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for bayesian optimization. *arXiv preprint arXiv:1802.02219*, 2018.

[11] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[12] Matthias Feurer, Jan N. van Rijn, Arlind Kadra, Pieter Gijsbers, Neeratyoy Mallik, Sahithya Ravi, Andreas Mueller, Joaquin Vanschoren, and Frank Hutter. Openml-python: an extensible python api for openml. *arXiv*, 1911.02490, 2020.

[13] Matthias Feurer, Jan N. van Rijn, Arlind Kadra, Pieter Gijsbers, Neeratyoy Mallik, Sahithya Ravi, Andreas Müller, Joaquin Vanschoren, and Frank Hutter. Openml-python: an extensible python api for openml. *arXiv:1911.02490*, 2019.

[14] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[15] Hadi S Jomaa, Josif Grabocka, and Lars Schmidt-Thieme. Hyp-rl: Hyperparameter optimization by reinforcement learning. *arXiv preprint arXiv:1906.11527*, 2019.

[16] Hadi S Jomaa, Lars Schmidt-Thieme, and Josif Grabocka. Dataset2vec: Learning dataset meta-features. *Data Mining and Knowledge Discovery*, 35(3):964–985, 2021.

[17] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *J. Global Optimization*, 13(4):455–492, 1998.

[18] Daniel Kühn, Philipp Probst, Janek Thomas, and Bernd Bischl. Automatic exploration of machine learning experiments on openml. *CoRR*, abs/1806.10961, 2018.

[19] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[20] Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, pages 6845–6855, 2018.

[21] Valerio Perrone and Huibin Shen. Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12751–12761, 2019.

[22] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

[23] David Salinas, Huibin Shen, and Valerio Perrone. A quantile-based approach for hyperparameter transfer learning. In *International Conference on Machine Learning*, pages 8438–8448. PMLR, 2020.

[24] Nicolas Schilling, Martin Wistuba, Lucas Drumond, and Lars Schmidt-Thieme. Hyperparameter optimization with factorized multilayer perceptrons. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 87–103. Springer, 2015.

[25] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.

[26] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

[27] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4141–4149, 2016.

[28] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[29] 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. *arXiv preprint arXiv:1904.02642*, 2019.

[30] 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, 2020.

[31] Martin Wistuba and Josif Grabocka. Few-shot bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representations*, 2021.

[32] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 104–119. Springer, 2015.

[33] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *International Conference on Data Science and Advanced Analytics*, pages 1–10, 2015.

[34] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Two-stage transfer surrogate model for automatic hyperparameter optimization. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 199–214. Springer, 2016.

[35] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78, 2018.

[36] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114. PMLR, 09–15 Jun 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

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]
   (c) Did you discuss any potential negative social impacts of your work? [N/A]
   (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 (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See our repository link in the introduction.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [No] We used pre-defined configurations from previous work.
   (c) Did you report error bars (e.g., concerning the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of computing and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
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]
   (b) Did you mention the license of the assets? [No] They are included in the cited publications.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We are using open-sourced assets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or researched 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]