A Simple and Fast Baseline for Tuning Large XGBoost Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

XGBoost, a scalable tree boosting algorithm, has proven effective for many prediction tasks of practical interest, especially using tabular datasets. Hyperparameter tuning can further improve the predictive performance, but training many models on large datasets can be time consuming. Owing to the discovery that (i) there is a strong linear relation between dataset size & training time, (ii) XGBoost models satisfy the *ranking hypothesis*, and (iii) lower-fidelity models can discover promising hyperparameter configurations, we show that uniform subsampling makes for a simple yet fast baseline to speed up the tuning of large XGBoost models using multi-fidelity hyperparameter optimization with data subsets as the fidelity dimension. We demonstrate the effectiveness of this baseline on large-scale tabular datasets ranging from $15 - 70$GB in size.

1 Introduction

Despite modern developments in deep learning models for tabular datasets [12, 22], XGBoost [4] has stood the test time of time and remains the favorite scalable tree boosting algorithm for a wide range of problems [21], including large-scale tabular datasets. Further performance gains can be realized by careful hyperparameter optimization (HPO) of XGBoost models.

One of the most successful HPO techniques is sequential Bayesian optimization (BO) [20]. BO has consistently proven to be the superior method for tuning black-box functions, as was also recently demonstrated by the NeurIPS 2020 Black-Box Optimization Challenge [24]. Its sequential nature is, however, limiting. XGBoost models with large-scale tabular datasets greater than 10GB in size, our focus in this work, come with significant computational costs — training a single model can be time consuming, and the full dataset may not even fit the memory.

In this work, we establish that uniformly subsampling large-scale tabular datasets provides a simple, fast, and surprisingly effective baseline for multi-fidelity hyperparameter optimization of XGBoost models. In particular, we show that:

- There is a strong linear relationship between the training time of XGBoost models and dataset size (in terms of the fraction of the full dataset). Naturally, training on smaller subsets provides substantial runtime gains.
- Hyperparameter configurations ranked by performance on lower-fidelity versions of XGBoost models (where the fidelity parameter is the fraction of the full dataset size) tend to maintain their relative ranking when trained on the full dataset. This hints that XGBoost models also satisfy the *ranking hypothesis* as discussed for neural network models in Bornschein et al. [2].
- Tuning lower-fidelity approximations of XGBoost models, with uniformly subsampling as little as 1% of the samples in the full dataset, leads to modest performance drops (often less

Submitted to the 5th Workshop on Meta-Learning at NeurIPS 2021, Sydney, Australia. Do not distribute.

than 0.5%) in terms of the validation score (e.g. AUROC) when compared to well-tuned models on the full dataset.

- For XGBoost models with large-scale tabular datasets, we demonstrate that Hyperband [14] is much more economical than an exhaustive randomized grid search in terms of the total wallclock time to achieve the same performance. Combining it with BO [6] allows us to squeeze out a few more runtime gains.

2 Motivations & Related Work

Our main inspiration comes from Bornschein et al. [2], which provides a detailed study of generalization performance of neural networks w.r.t dataset size, which complements existing studies w.r.t model size. The authors propose the *ranking hypothesis*: over-parameterized neural network models tend to maintain their relative ranking over a wide range of data subsets drawn from the same underlying data distribution.

Our focus, however, is the training of batch models like XGBoost [4] with very large datasets (often larger than 10GB). Neural networks already have the luxury of stochastic optimization using mini-batches of data, but XGBoost carries a few qualitative differences as it uses all data at once. It relies on boosting, i.e., greedily building an additive model by adding one base function at a time that learns only the residual predictive function. The number of boosting rounds can increase the model capacity. This is unlike neural networks, where the model capacity is fixed for a given architecture. Further, data (rows) and feature (columns) subsampling is already supported by XGBoost, but is not to be confused with our goals. XGBoost subsamples for the robustness of the constructed ensemble, whereas we are aiming for a simple approach to reduce the computational burden of tuning large XGBoost models without significantly compromising performance via lower fidelity approximations based on data subsets. Notably, He et al. [10] briefly describe the use of data subsampling in XGBoost models used as feature extractors for logistic regression, but do not fully explore the computational and performance benefits for tuning of XGBoost models.

A large fraction of the literature has focused their analysis on tuning large neural networks models with stochastic training using subsets of data [9, 3, 16]. Most recently, Klein et al. [13] propose FABOLAS, a general framework to model the loss and training time as a function of the dataset size, inspired by multi-task BO [23] where the tasks are now continuous. The evaluation, however, is still focused on neural network models. Most notably, the analysis highlights the importance of using wallclock times for comparing HPO algorithms for practical usage, which we also do in this work.

With the success of Bornschein et al. [2], one may be tempted to believe that the inductive biases of neural networks are aligned with natural data like images, which form the bulk of the benchmarks, and are therefore amenable to training using subsets. Tabular datasets, however, are not expected to have such easily exploitable biases, as has been shown by previous work [21, 12, 22]. Surprisingly, to the contrary, we empirically demonstrate that batch trained models like XGBoost are also amenable to training with uniformly sampled subsets of large datasets.

3 Background

There are two broad approaches to achieve scalable and efficient hyperparameter optimization: (i) modeling the landscape of hyperparameters to model's performance, we can be more efficient about *configuration selection*, e.g., through BO [20]; and (ii) adaptively allocating computational resources, we can evaluate a large number of hyperparameters, and be more efficient about *configuration evaluation*, e.g., through Hyperband [14]. Both approaches can also be combined for further practical gains [6].

Bayesian Optimization (BO) We can formulate the performance of any machine learning model as a function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is the search space of hyperparameter configurations. The hyperparameter optimization (HPO) problem can then be defined as the search for the optimal configuration $x_\star \in \mathcal{X}$, where $x_\star = \arg\max_{x \in \mathcal{X}} f(x)$ gives us the globally optimal score value (e.g., validation accuracy for classification models). BO models this function using a probabilistic model, $p(f \mid \mathcal{D})$, conditioned on the evaluations $\mathcal{D} = \{(x_0, f(x_0)), \dots\}$ observed so far [8]. We can use this model to query a new configuration x' that maximizes an *acquisition function* $a(x)$ of interest, i.e., $x' = \arg\max_{x \in \mathcal{X}} a(x)$. A common choice of the probabilistic model is *Gaussian processes* [19], and the acquisition function is the *Expected Improvement* (EI) $a(x) = \mathbb{E}_{p(f|\mathcal{D})}[\max(0, f(x) - f(x^\star))]$ [15], where $x^\star$ is the best configuration seen so far. For subsequent trials, we refit the model with $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x', f(x'))\}$, and repeat the acquisition step.

Multi-Fidelity Hyperparameter Optimization (HPO) For a given black box function $f(x)$, multi-fidelity optimization aims to learn from an augmented function $f(x, r)$ with a fidelity parameter $r \in [r_{\min}, r_{\max}]$, such that $f(x) = f(x, r_{\max})$ [23]. In the case of HPO, r represents the computational resource. The expectation is that low fidelity approximations of the true function, i.e., $r < r_{\max}$, are computationally much cheaper, but informative towards learning $f(x)$. Popular choices of the fidelity parameter r include the number of epochs when training neural networks, or the fraction of the full dataset used for training the model.

Hyperband (HB) Framing the hyperparameter optimization as a multi-armed bandit problem, Hyperband [14] is an approach towards multi-fidelity HPO that builds upon repeated trials of *Successive Halving* (SH) [11]. For a total computational budget B, SH uses the average budget B/r for each hyperparameter configuration B/r, where r is fixed a priori. This leads to a trade off — a small value of r would allow many evaluations, but at lower and less reliable fidelities, whereas a large value of r would allow only a small number of reliable evaluations. Hyperband instead uses multiple trials of SH ("*brackets*") for different values of r ("*rung levels*"). At each bracket, Hyperband ranks the different configurations, only allowing the top $1/\eta$ fraction to continue to a higher rung level. Hyperband relies on random draws of the hyperparameter configurations for convergence to the global optimum, and often works very well for small to medium computational budget. By accounting for information from existing evaluations, BOHB [6] improves upon Hyperband by combining BO with HB.

4 Experiments

Datasets In our benchmark study, we focus on large-scale tabular datasets which are at least 10 GB in raw size. The actual size after feature preprocessing is often much larger. We keep the feature preprocessing to a minimum as provided by AWS Sagemaker [5, 18], which includes converting text features into tf-idf vectorization [1], categorical variables into one-hot representation, and splitting datetime variables across days/weeks/months.[1] We include datasets that are both classification and regression to demonstrate the generality of our results. The complete list of benchmark datasets and their key details are provided in Table 1. For brevity, we only show results using a subset of the datasets, and the remainder of the figures are available in Appendix A.

Table 1: For our study, we consider large tabular datasets with a raw size of approximately 10 GB, whose sizes after feature preprocessing are noted below. The number of rows are represented by N, and the number of raw features by D.

Dataset	Kind	N	D	Processed Size (GB)
adform	Binary Classification	23,999,936	108	56.2
adfraud	Binary Classification	149,813,196	9	30.2
lendingc	Binary Classification	1,760,668	990	29.3
codes	10-Way Classification	22,889,691	9	15.9
taxifare	Regression	44,936,324	17	69
reddit-score	Regression	36,008,714	18	56.6
census-income	Regression	2,452,939	789	38.2

Evaluation For all regression datasets, we maximize the R^2 score. For all binary classification datasets, we use the weighted AUC score, and for multiclass classification datasets, we use the one-vs-rest formulation of weighted AUC score [7]. We use the implementation provided by Pedregosa et al. [17]. All evaluation scores need to be maximized by the HPO algorithm.

[1]The complete set of feature processors and their implementation is available at https://github.com/aws/sagemaker-scikit-learn-extension

HPO Tuning For our study, we focus on multi-fidelity HPO with Hyperband (HB) [14] and BOHB [6]. All results are compared to an exhaustive randomized grid search as the gold standard, where we run each algorithm for a total budget of approximately 60000 seconds ($\sim$ 17 hours) on AWS Sagemaker [5, 18] using `m5.12/24xlarge` CPU instances. As noted earlier, we use the fraction of the full dataset size as the fidelity parameter r, which is chosen from the set $\mathcal{R} = \{1/100, 1/10, 1/4, 1/2, 3/4, 1\}$. This choice is of practical consequence as we describe in Section 4.1. Table 2 provides the details of the tuned hyperparameters.[2]

Table 2: The set of XGBoost hyperparameters tuned are in the table below, with their considered ranges. For reference, the corresponding XGBoost hyperparameter names are provided alongside the sampling distribution used to sample the range.

Hyperparameter	XGBoost Parameter	Distribution (Range)
Learning Rate	`eta`	$\log\text{-uniform}(10^{-3}, 1.)$
ℓ_1 Regularization	`alpha`	$\log\text{-uniform}(10^{-6}, 2.)$
ℓ_2 Regularization	`lambda`	$\log\text{-uniform}(10^{-6}, 2.)$
Min. Split Loss	`gamma`	$\log\text{-uniform}(10^{-6}, 64.)$
Row Subsample Ratio	`subsample`	$\text{uniform}(0.5, 1.)$
Column Subsample Ratio	`col_subsample`	$\text{uniform}(0.3, 1.)$
Max. Tree Depth	`max_depth`	$\log\text{-randint}(2, 8)$
Boosting Rounds	`num_round`	$\log\text{-randint}(2, 1024)$

4.1 Data Subsampling and Training Runtime

We find that the relationship between the training time of a single XGBoost model and the dataset size is roughly linear. This observation is consistent across all our benchmark datasets when we consider the fraction of the dataset size $r \in \mathcal{R}$, as visualized in Figure 1. Reducing the fraction r further to 1/1,000 or 1/10,000 does not provide proportional gains to be meaningful in practice (often amounting to less than 500ms per run).

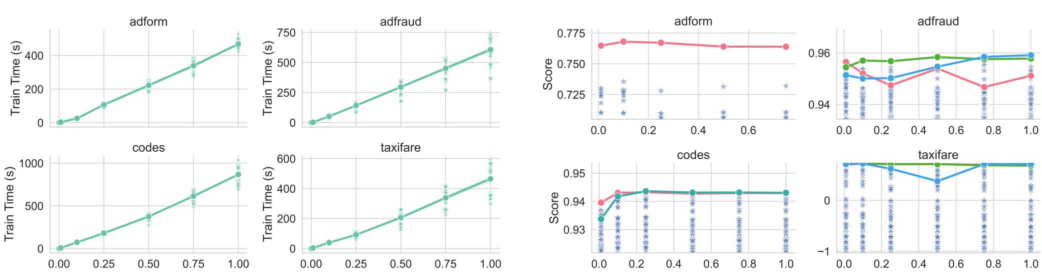


Figure 1: In these plots, $\star$'s denote individual runs corresponding to different hyperparameter configurations. **(Left)** We find a linear relationship between XGBoost training time and the dataset size fraction $r \in \mathcal{R}$. This has practical consequences (Section 4.1). **(Right)** XGBoost models satisfy the *ranking hypothesis* [2], making them amenable to multi-fidelity HPO (Section 4.2). For each dataset, we pick the best performing configuration at each fraction r, and see how it performs across all other fractions, connected via a line of the same color. Well-performing configurations in lower-fidelity models typically maintain performance on the full-fidelity model too. We crop the bottom quantile for better legibility. For instance, on the dataset `adform`, there is only a single line, indicating that the best performing configuration through all fidelities r is the same.

This observation has two important practical consequences: (i) the HPO tuning algorithm can now benefit from faster runs of the lower fidelity models, and a model using 1/10 the data can train roughly 10 times faster than the full-fidelity model; (ii) the linear relationship can be successfully exploited by Hyperband for efficient resource allocation, since the algorithm expects the relationship

[2] See https://xgboost.readthedocs.io/en/latest/parameter.html#general-parameters for the XGBoost hyperparameters.

to be roughly linear. A large deviation from linearity would break the assumptions such that lower-fidelity models end up getting disproportionately larger time than desired, defeating the resource allocation strategy of Hyperband.

4.2 The Ranking Hypothesis

Bornschein et al. [2] note that overparameterized neural network architectures seem to maintain their relative ranking in terms of generalization, when trained on arbitrarily small subsets of data. This is termed as the *ranking hypothesis*, and established empirically. Neural networks are trained using stochastic minibatches of $i.i.d$ data, and a priori, it would appear that batch models like XGBoost would not satisfy the ranking hypothesis. Surprisingly, however, XGBoost models satisfy the ranking hypothesis for all our benchmarks considered. Observing Figure 1 more closely reveals that the trend may not always be monotonic from the lowest fidelity to the highest fidelity and configurations may switch ranks. Nevertheless, overall we notice that well-performing lower fidelity runs tend to perform well also with the full dataset.

As a consequence of this property, we can afford to use far fewer computational resources to discover competitive hyperparameter configurations. Moreover, this property is necessary when using adaptive resource allocation HPO algorithms such as Hyperband [14]. In practice, we find that optimizing using very small data subsets (say $r = 10^{-5}$) can lead to over-regularized XGBoost models, whose configurations do not perform as well when retrained on the full dataset. This highlights that the choice of the minimum fidelity level $r_{\min}$ is crucial to a successful multi-fidelity HPO. Our experiments in Section 4.3 and the visuals in Figure 2 reveal that $r = 1/100$ works for most datasets.

4.3 Relative Performance of Lower-Fidelity Models

Considering the best-scoring configuration on the full dataset as the reference, we quantify the relative performance of lower-fidelity models on the validation set in two ways — (i) *Without retraining*: pick best tuned model at each fidelity r, and directly compute the score. (ii) *With retraining*: pick the configuration corresponding to the best tuned model at each fidelity r, and retrain with the full dataset. To test the performance limits of the models, in addition to $\mathcal{R}$, we test on $r \in \{10^{-4}, 10^{-5}\}$ as well. The results are visualized in Figure 2.

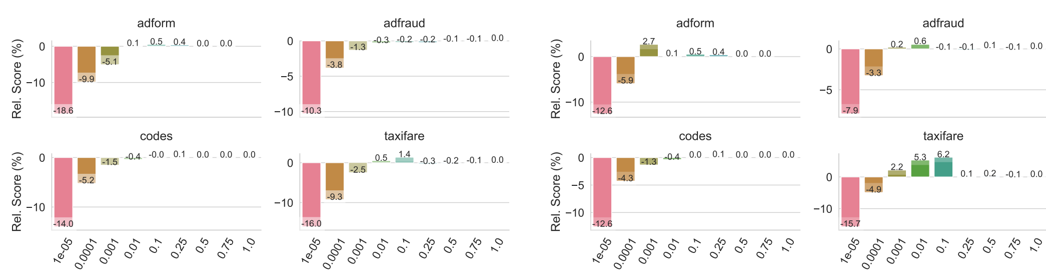


Figure 2: We compare the relative performance of lower-fidelity XGBoost models to the full-fidelity model trained with the full dataset (Section 4.3). In addition, to push r to its limit, we include $\{10^{-3}, 10^{-4}, 10^{-5}\}$, which suffer a much greater performance drop. **(Left)** *Without retraining*, lower fidelity models as low as $r = 0.01$ (i.e., 1% of the training size) can sustain a reasonably low drop in the validation score. The model effectively breaks when using even smaller subsets. **(Right)** *With retraining*, we find that the hyperparameter configurations of the lower-fidelity models can further close the generalization gap, often performing better potentially due to the regularization effect of using data subsets.

In summary, we find that we are able to sustain (on average across benchmark datasets) as low as 3.3% drop in performance when training with as little as 1% ($r = 1/100$) of the full training dataset, sampled uniformly at random. Further, retraining with the full dataset reduces the generalization gap to just 1.4%. Higher fidelity models are even better, sustaining less than 0.5% error on average. This result is of immense practical value as we can discover competing configurations with far lower computational costs, as we demonstrate in Section 4.4.

4.4 Economical HPO

By virtue of the facts that, (i) training on data subsets leads to proportionately faster training time (Section 4.1), (ii) XGBoost models satisfy the ranking hypothesis for all practical purposes (Section 4.2), and (iii) lower-fidelity models can discover high performing configurations Section 4.3, it is now reasonable to expect benefits in the computational cost of hyperparameter optimization, especially in terms of the total wallclock time.

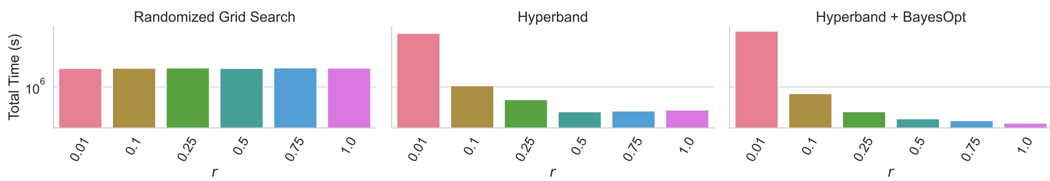


Figure 3: Owing to XGBoost models training faster with data subsets, satisfying the ranking hypothesis, and maintaining high-performance with lower-fidelity approximations, we are able to achieve significantly faster wallclock times for the HPO of large-scale XGBoost models, shown here for the `adform` dataset (Section 4.4). The y-axis is $\log$-scaled. Notably, Hyperband spends much more time training lower fidelity models, but is able to try many more configurations. Randomized grid search instead spends time in higher-fidelity configurations, which is wasteful if the configuration is not promising.

To validate this, we compare random search to both our subsampling-based Hyperband proposal and to its BO extension in Figure 3. The results show that higher-fidelity models now take far less wallclock time, and that we can tune large-scale XGBoost models considerably faster. Further, we find that combining BO with Hyperband, as in Falkner et al. [6], can provide further marginal improvements in the wallclock time of model tuning. Unlike randomized grid search, which would allocate roughly the same time to more expensive higher-fidelity configurations, smarter resource allocation as in Hyperband [14] and smarter candidate configuration as with BO [6] can provide meaningful computational cost savings.

5 Conclusions & Future Outlook

XGBoost remains an effective model choice for many practical problems in the industry, but catering to very large datasets is a computational challenge for such batch models, i.e., models which do not employ minibatching of the data for stochastic optimization. Further, small changes in XGBoost hyperparameters can have large effects; for instance, changing the tree depth can drastically change the learned predictor, which one could expect to be a consequence of data subsampling.

Our work instead provides surprising evidence to the contrary — XGBoost satisfies many of the favorable properties that allow us to exploit multi-fidelity hyperparameter optimization towards faster tuning, most importantly the ability to discover promising hyperparameter configurations with subsets of data as small as 1% of the total size, constructed simply by uniform sampling.

Limitations & Future Work The simplicity and speed of uniform sampling of the dataset is the key strength of our proposed baseline for multi-fidelity hyperparameter optimization. While this may be enough for curated datasets, it also remains fundamentally limited in its ability to always provide a representative subset for any dataset in the wild. Therefore, much of our future effort lies in finding reliable ways to summarize datasets using informative samples.

Societal Impact By using uniform subsampling to construct data subsets, the results presented in this work rely on the often commonly used assumption in machine learning that data is $i.i.d.$ and covers the true underlying data distribution reasonably well. If the dataset has unfavorable biases towards certain subpopulations, those may be exacerbated by simple uniform subsampling. Better subsampling methods accounting for such scenarios must be a consideration for practical usage of our proposed baseline when such assumptions are violated.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval - the concepts and technology behind search, Second edition. 2011. 3

[2] J. Bornschein, Francesco Visin, and Simon Osindero. Small Data, Big Decisions: Model Selection in the Small-Data Regime. In *ICML*, 2020. 1, 2, 4, 5

[3] L. Bottou. Stochastic Gradient Descent Tricks. In *Neural Networks: Tricks of the Trade*, 2012. 2

[4] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. 1, 2

[5] Piali Das, Valerio Perrone, Nikita Ivkin, Tanya Bansal, Zohar Karnin, Huibin Shen, Iaroslav Shcherbatyi, Yotam Elor, Wilton Wu, Aida Zolic, Thibaut Lienart, Alex Tang, Amr Ahmed, Jean Baptiste Faddoul, Rodolphe Jenatton, Fela Winkelmolen, Philip Gautier, Leo Dirac, Andre Perunicic, Miroslav Miladinovic, Giovanni Zappella, Cédric Archambeau, Matthias Seeger, Bhaskar Dutt, and Laurence Rouesnel. Amazon SageMaker Autopilot: a white box AutoML solution at scale. *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, 2020. 3, 4

[6] S. Falkner, Aaron Klein, and F. Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *ICML*, 2018. 2, 3, 4, 6

[7] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognit. Lett.*, 27:861–874, 2006. 3

[8] P. Frazier. A Tutorial on Bayesian Optimization. *ArXiv*, abs/1807.02811, 2018. 3

[9] J. Hartmanis and J. V. Leeuwen. Neural Networks: Tricks of the Trade. In *Lecture Notes in Computer Science*, 2002. 2

[10] Xinran He, Junfeng Pan, Ou Jin, T. Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, R. Herbrich, S. Bowers, and J. Q. Candela. Practical Lessons from Predicting Clicks on Ads at Facebook. In *ADKDD'14*, 2014. 2

[11] Kevin G. Jamieson and Ameet S. Talwalkar. Non-stochastic Best Arm Identification and Hyperparameter Optimization. *ArXiv*, abs/1502.07943, 2016. 3

[12] Arlind Kadra, M. Lindauer, F. Hutter, and Josif Grabocka. Regularization is all you Need: Simple Neural Nets can Excel on Tabular Data. *ArXiv*, abs/2106.11189, 2021. 1, 2

[13] Aaron Klein, S. Falkner, Simon Bartels, Philipp Hennig, and F. Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. *ArXiv*, abs/1605.07079, 2017. 2

[14] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet S. Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *J. Mach. Learn. Res.*, 18:185:1–185:52, 2017. 2, 3, 4, 5, 6

[15] J. Mockus. On Bayesian Methods for Seeking the Extremum and their Application. In *IFIP Congress*, 1977. 3

[16] T. Nickson, Michael A. Osborne, S. Reece, and S. Roberts. Automated Machine Learning using Stochastic Algorithm Tuning. 2014. 2

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 3

[18] Valerio Perrone, Huibin Shen, Aida Zolic, I. Shcherbatyi, A. Ahmed, Tanya Bansal, Michele Donini, Fela Winkelmolen, Rodolphe Jenatton, J. Faddoul, Barbara Pogorzelska, Miroslav Miladinovic, K. Kenthapadi, M. Seeger, and C. Archambeau. Amazon SageMaker Automatic Model Tuning: Scalable Gradient-Free Optimization. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021. 3, 4

[19] C. Rasmussen and Christopher K. I. Williams. Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). 2005. 3

[20] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016. doi: 10.1109/JPROC.2015.2494218. 1, 2

[21] Ravid Shwartz-Ziv and A. Armon. Tabular Data: Deep Learning is Not All You Need. *ArXiv*, abs/2106.03253, 2021. 1, 2

[22] G. Somepalli, Micah Goldblum, Avi Schwarzschild, C. B. Bruss, and T. Goldstein. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. *ArXiv*, abs/2106.01342, 2021. 1, 2

[23] Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-Task Bayesian Optimization. In *NIPS*, 2013. 2, 3

[24] Ryan Turner, David Eriksson, M. McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and I. Guyon. Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. In *NeurIPS*, 2020. 1

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]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (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 (either in the supplemental material or as a URL)? [No] The code and the data are proprietary.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We found the performances to have negligible variance across seeds in our experiments.
   (d) 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]
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? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (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 conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional Figures

A.1 Data Subsampling and Training Runtime

In continuation to Section 4.1, we provide the training runtime plots for the remainder of our benchmark datasets (see Table 1) in Figure 4.

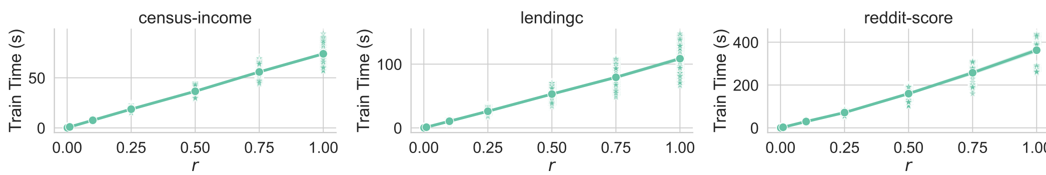


Figure 4: As in Figure 1(a), for the remainder of our benchmark datasets too, we find a strong linear relationship.

A.2 The Ranking Hypothesis

For the remainder of the datasets of our benchmark in Figure 5, we are able to demonstrate *the ranking hypothesis* is satisfied. The consequences of this are discussed in Section 4.2.

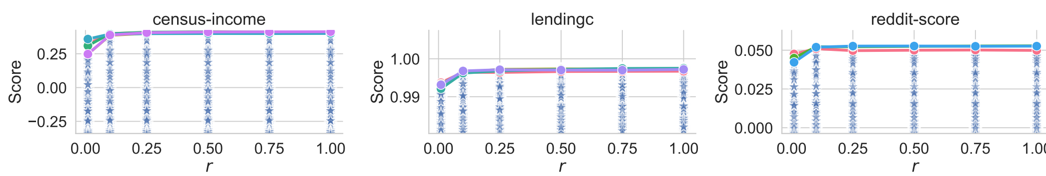
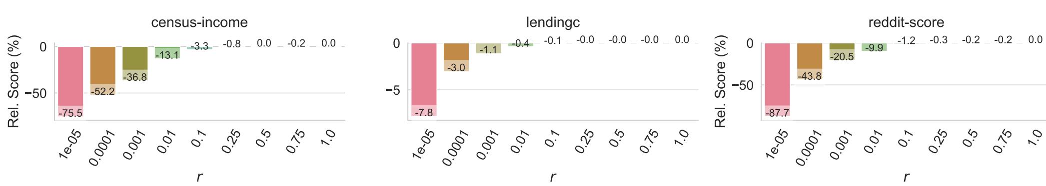


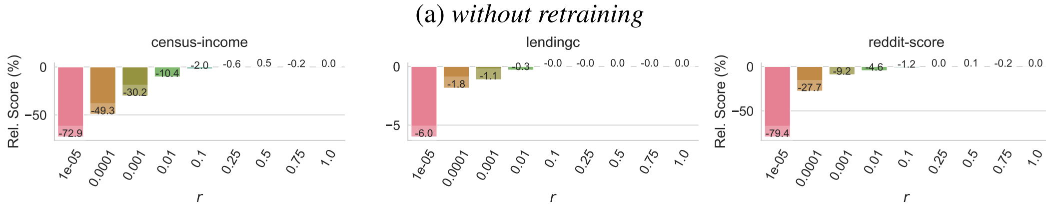
Figure 5: As in Figure 1(b), the remainder of our benchmark datasets satisfy *the ranking hypothesis* as well.

A.3 Relative Performance of Lower-Fidelity Models

For the remainder of the datasets, we make a similar assessment *without retraining* and *with retraining*, as described in Section 4.3.



(a) *without retraining*



(b) *with retraining*

Figure 6: As in Figure 2, the remainder of our benchmark datasets also show similar trends in performance with uniformly subsampled datasets. Here again, we show much lower values of r to push the subsampling to its limits.

A.4 Economical HPO

We provide the cumulative tuning time plots, as in Section 4.4, for the remainder of the datasets in Figure 7.

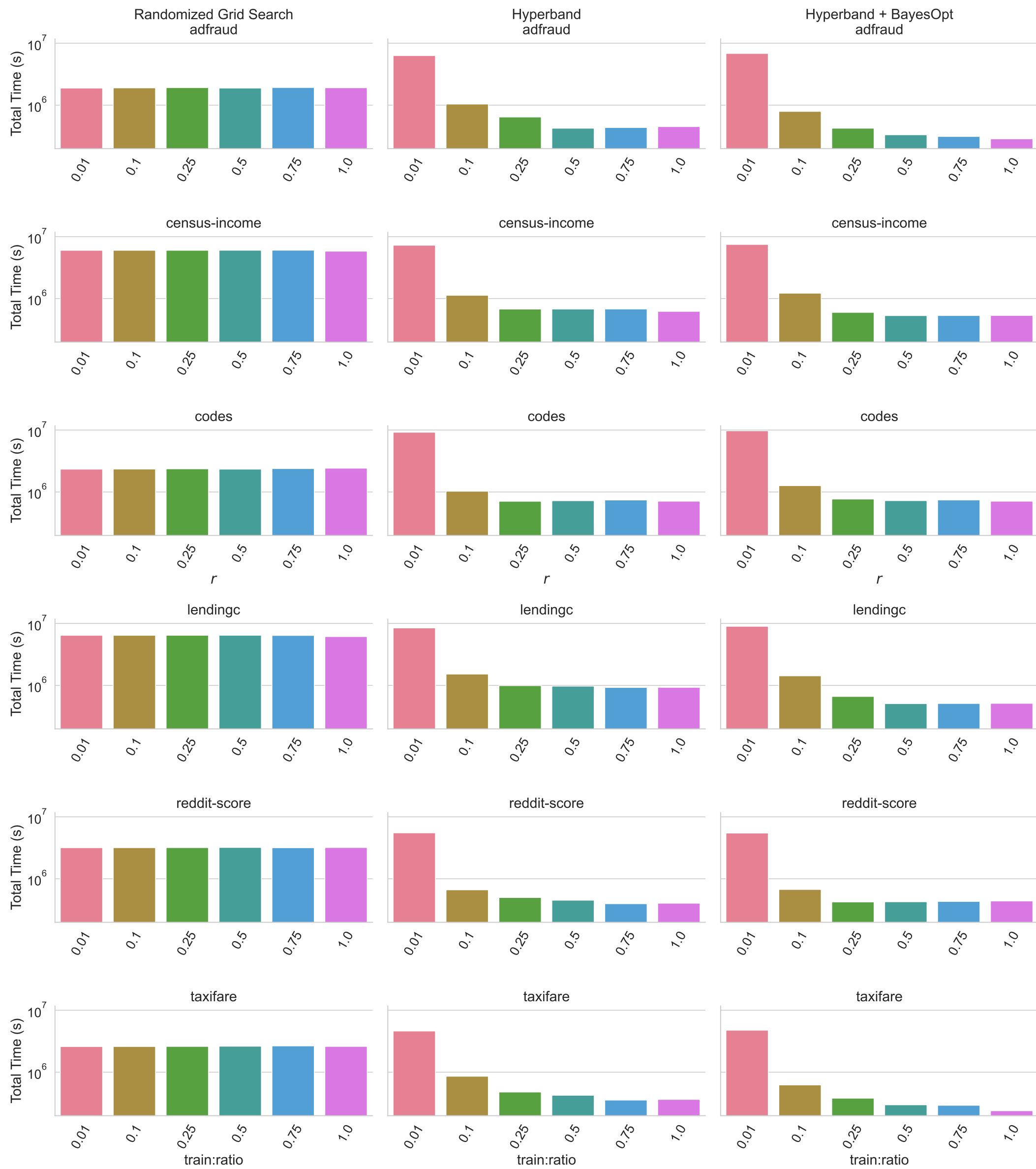


Figure 7: As in Figure 3, the remainder of our benchmark datasets reveal similar runtime trends, where combining with Bayesian optimization can often have practical benefits. More importantly, by virtue of the Hyperband resource scheduling, we are able to test many more configurations and only spend higher resources on the most promising ones.