7th ICML Workshop on Automated Machine Learning (2020)

Collecting Empirical Data About Hyperparameters for Data Driven AutoML

Martin Binder MARTIN.BINDER@STAT.UNI-MUENCHEN.DE
Florian Pfisterer FLORIAN.PFISTERER@STAT.UNI-MUENCHEN.DE
Bernd Bischl BERND.BISCHL@STAT.UNI-MUENCHEN.DE
Department of Statistics, LMU Munich, Germany

Abstract

All optimization needs some kind of prior over the functions it is optimizing over. We used a large computing cluster to collect empirical data about the behavior of ML performance, by randomly sampling hyperparameter values and performing cross-validation. We also collected information about cross-validation error by performing some evaluations multiple times, and information about progression of performance with respect to training data size by performing some evaluations on data subsets. We present how we collected data, make some preliminary analyses on the surrogate models that can be built with them, and give an outlook over interesting analysis this should enable.

1. Introduction

Hyperparameter optimization (HPO) is an important aspect of automated machine learning (AutoML) and is often tackled as a black-box optimization problem. The No Free Lunch (NFL) theorem for optimization (Wolpert and Macready, 1997) states that the performance of optimization algorithms, averaged over all possible problem sets, is constant, and that it is therefore necessary to make a-priori assumptions about the characteristics of the problem at hand. A well-known manifestation of this is the superiority of random search over grid search on typical hyperparameter optimization problems, which is a consequence of them often having low effective dimensionality (Bergstra et al., 2011). Another instance of this is the favorable performance of model-based optimization (Snoek et al., 2012) for hyperparameter optimization, which often uses explicit Bayesian priors (Bayesian optimization) to achieve good performance with few function evaluations. We performed evaluations of different ML algorithms with randomized hyperparameters on a variety of datasets to gather empirical data about the influence of these hyperparameters on algorithm performance.

Model-based optimization makes use of surrogate models, which are regression models fitted to evaluated objective values of the problem. Surrogate models can also be used to benchmark or tune optimization algorithms (Eggensperger et al., 2013). With the large amount of performance data that we have collected it is possible to fit high fidelity surrogate models that can be used to analyse the behavior of classical machine learning algorithms with respect to their hyperparameters.

The importance of hyperparameter optimization for deep learning (neural architecture search, Elsken et al. (2019)) has been growing in recent years. Because of their nature – large hyperparameter spaces and very long model fitting times – they have led to more research into multi-fidelity approaches for HPO (Li et al., 2017; Tan and Le, 2019). Multi-fidelity approaches vary certain fidelity hyperparameters that trade-off between model performance evaluation cost and model performance. Here, an important prior assumption about the behavior of ML algorithms with respect to the fidelity parameters is that performance at low fidelity is somehow predictive for performance at high fidelity. Because this assumption is so crucial, it is important to conduct empirical studies, and to gather data on the effect of approaches towards trade-offs between model performance and computation time.

We have done a very large number of performance evaluations of machine learning algorithms on a diverse set of 119 datasets with randomly sampled hyperparameters to gather empirical data about the influence of hyperparameter configurations on machine learning performance. We performed some of these evaluations on different subsamples of the data, which can be used to empirically evaluate a sub-sampling approach to multi-fidelity optimization. We further used repeated cross-validation for some evaluations, to get an estimate of (some of the) uncertainty of the cross-validation estimators.

2. Related Work

The importance of gathering empirical data about hyperparameter influence on performance has been recognized and several projects exist that do this on different classes of machine learning algorithms and problems. Collections of experimental results like ours are often published with papers for new methods (Wistuba et al., 2015a; van Rijn and Hutter, 2018), but they are rarely as comprehensive and often only suffice for the particular task being presented. Especially in the context of deep learning, several collections of experimental data have been made available lately. Kühn et al. (2018) publish results on 38 tabular datasets across 6 algorithms for further analysis. NASBENCH-101 (Ying et al., 2019) provide a collection of experimental results across 423.000 convolutional neural network architectures on CIFAR-10 for faster and more reproducible analysis of neural architecture search strategies. (Metz et al., 2020) publish experimental results for neural network optimizers across 1162 diverse datasets and propose sets of default configurations.

Experimental data is already being used to improve optimization performance. For one, the HPOLib benchmark suite for black-box optimization (Eggensperger et al., 2013), which can be used by researchers to evaluate their black box optimization algorithms, uses surrogate models for some benchmarks (Eggensperger et al., 2015; Klein et al., 2019). Optimization algorithms may get tuned on, or at least chosen by their performance on, these surrogate models. The experimental data – on which the models are based – are thus influencing the implicit prior assumptions made by these algorithms in an indirect way. A more direct influence of experimental data on optimization is meta-learning (Brazdil et al., 2008; Vanschoren, 2019), e.g. by studying the importance of various hyperparameters (van Rijn and Hutter, 2018; Probst et al., 2018), or try to find initial configurations (Wistuba et al., 2015b; Pfisterer et al., 2018; van Rijn et al., 2018) that perform well on many datasets.

3. Setup

We executed a large quantity of machine learning performance evaluations on a large grid of computers of 3168 compute nodes, each with 48 physical (96 logical) CPU cores and 80 GB working memory, for 48 hours.

The evaluations were performed on a collection of 119 classification task datasets chosen from the OpenML-CC18 (Bischl et al., 2017) benchmark suite, as well as the AutoML benchmark (Gijsbers et al., 2019). These datasets cover a large variety of different challenges for machine learning, such as many missing values, large cardinality of factorial features, large number of features, or great imbalance of outcome classes. We obtained datasets, as well as resampling splits from OpenML (Vanschoren et al., 2014).

We investigate an array of classical machine learning algorithms: decision trees, random forests, svm, gradient boosting, approximate KNN (Malkov and Yashunin, 2020), fully connected neural networks, and regularized logistic regression (Zou and Hastie, 2005). Because not all these algorithms can natively handle all datasets, we performed data-preprocessing: missing value imputation, factorial feature cardinality reduction, and factor one-hot encoding among others. The specific search spaces (including software libraries used) as well as the preprocessing setup are detailed in Appendix A and B, respectively.

Values for each hyperparameter were sampled independently from individual univariate distributions. A common problem in previous data collections is a limited hyperparameter search space. If good values lie on the border of investigated hyperparameter spaces (e.g. large number of gradient boosting iterations), then meta-learning approaches might miss important facts about algorithm behavior, such as eventual overfitting beyond the investigated region. Using too broad limits for sampling, on the other hand, can lead to many evaluations in uninteresting regions where the algorithm crashes or predicts constant. We alleviate this problem by defining intervals to sample from that were chosen informally and from previous experience. However, we purposely sample outside of these intervals with a small probability in order to obtain a more complete picture w.r.t. algorithm behavior beyond the regions. For this we sample a mixture distribution: uniformly distributed inside a specified range with probability 5/6, and normally distributed centered in the middle of this range and standard deviation half the range width with probability 1/6. The investigator-chosen ranges were therefore *soft bounds* that contain about $5/6 + 1/6 \times 68\% = 95\%$ of all sampled points[1]. We made the prior assumption that many hyperparameters contain more variation close to 0 than further away from it. These were sampled as described here, but on a log-scale (including mixture uniform-normal sampling) and then exponentiated. Hyperparameters with nominal discrete values were sampled uniformly. The specific hyperparameters, their bounds and transformation are listed in Appendix A.

The same hyperparameter samples were used on all datasets. This makes it possible to investigate the direct effect of dataset properties on machine learning performance while keeping hyperparameters constant, without having to resort to surrogate modelling.

Performance evaluation was performed using 10-fold cross-validation, using pre-defined, balanced (with respect to the outcome class) cross-validation folds provided by OpenML. To make it possible to investigate the effect of subsampling before model training, or the effect of the specific resampling split on the performance estimate, we also performed what we call *super-evals* (supererogatory evaluations) on 10% of all sampled hyperparameter points. (Which configurations were super-evaluated was kept constant across all datasets). For super-evals, we employed: (1) the 10-fold cross-validation used on all other configuration points; (2) another 10-times-10-fold repeated cross-validation; and (3) a sequence of

1. Some hyperparameter also presented *hard bounds*, e.g. when negative values are forbidden, in which case hyperparameter values were sampled from a truncated distributions.

Algorihtm	N	# of DS	avg. N	min N	max N	OpenML ID
glmnet	104820	114	919	58	2235	42578
ranger	278863	119	2343	356	4754	42580
knn	111753	116	963	146	2792	42582
rpart	92067	115	801	85	1382	42583
svm	540576	106	5100	174	13352	42577
xgboost	2955210	119	24834	2294	41147	42584
feed-forward NN	171691	107	1605	730	4133	42579

Table 1: Information about generated data: Experiment counts across different algorithms, number of datasets for which data could be generated, average, min, and max number of values per dataset, and OpenML ID of the performance data. Values not counting super-evals.

cross-validations with reduced training set size using subsampling (see Appendix C for the subsample sizes), simulating one approach to multi-fidelity cross-validation. The specific subsamples are a tower of subsets of the training sets used for (1) and fixed for each dataset.

Performance was evaluated by training the machine learning methods on the chosen training data subset and predicting on all other data-samples. Individual evaluation threads were limited w.r.t. the amount of working memory (up to 55 GB; memory limits were different depending on the algorithm and dataset) they could consume, and the amount of time they could use for a single cross-validation fold (4 hours). Available resources (both time and memory) for each learner and dataset were determined beforehand in a small pilot experiment. Therefore, the amount of data generated varies across them. A small number of datasets were too large for some ML algorithms and no data could be generated with the randomly sampled hyperparameters given the time or memory constraints.

Besides machine learning performance data, we also collected information about the working memory required for each configuration, as well as the training and prediction runtime. This data can be used to investigate memory and time requirements for different algorithms based on dataset properties and hyperparameters, and may help to improve AutoML systems to better adhere to runtime and memory limits in the future. Table 1 gives an overview of the generated data.

4. Analysis

Several routes for the analysis of this data can be envisioned. In this work we decide to study the quality of resulting surrogate models and the effect of the number of evaluated configurations on that quality. We use the collected data to fit random forest surrogate models. The usefulness of a surrogate model depends on its predictive performance, which we can estimate using cross-validation.

It is often not clear how much experimental data is needed to build sufficiently accurate surrogate models. The large amount of data that we have gathered makes it possible to analyse the behavior of surrogate model fidelity with respect to data size, and to estimate the marginal gain in model quality that would have been possible if even more data had been collected. We set the resampling error, i.e. the error introduced by using surrogate models

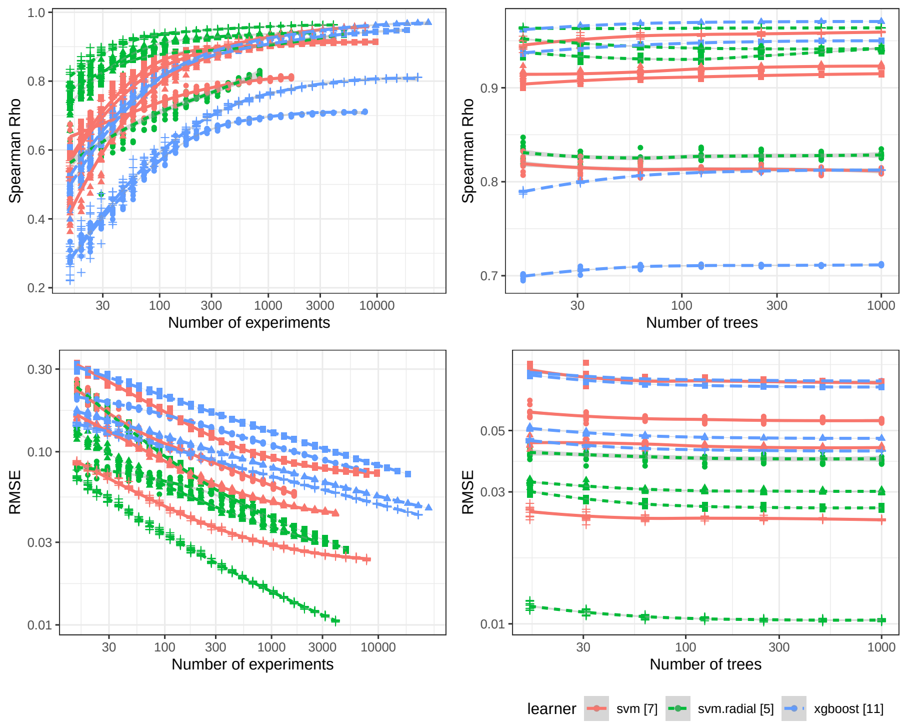

Figure 1: Resampling error of surrogate models with respect to number of included performance data points (left) and number of trees for the surrogate model (right) across 4 different datasets. Algorithm svm is restricted to the radial kernel. It can be seen that surrogate model performance varies with dataset as well as the learning algorithm.

instead of actual function evaluations, in relationship to the random noise introduced by having finitely many trees in the random forest model, by analysing the progression of error with different number of trees in the model.

Figure 1 shows the (ten-fold cross-validation) resampling error, both in terms of root mean square error (RMSE) as well as Spearman rank correlation (Spearman Rho), and how it progresses with different training set sizes, exemplary for a few datasets and algorithms evaluated on them. Furthermore, the resampling error w.r.t. the number of trees used in the surrogate model is shown. It becomes obvious that a few hundred random forest trees are enough for all shown datasets, and that the limiting factor is the sample size.

4.1 Hyperparameter Response Surface

Figure 2 shows the average and standard deviation of the normalized accuracy for the SVM `cost` and `gamma` parameters evaluated on a grid with resolution 200. Accuracy was normalized to $[0, 1]$ for each task to improve commensurability. Standard deviation seems to be low in areas with generally good performance and high for larger values of `gamma`. As we gathered a large number of evaluations for each dataset, we can build high-fidelity surrogate models which allow for a more realistic analysis of corresponding response surface.

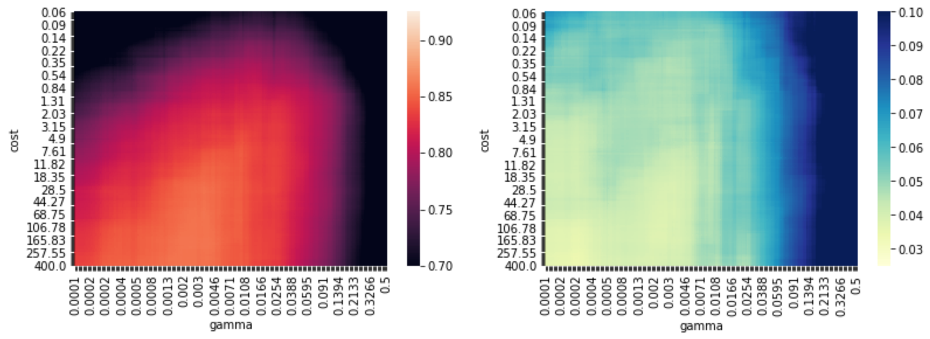

Figure 2: Response surfaces of surrogate models for the performance of SVM with respect to its `gamma` and `cost` parameters. Mean (left) and standard deviation (right) of normalized classification accuracy across the 106 datasets for which SVM generated results.

5. Further Analysis & Future Work

We presented the large collection of resampling performance data which we have generated, and showed some statistics and illustrative analysis results. There are, however, many directions to analyse and make use of the produced hyperparameter data in this work. We would like to use this data in order to reproduce and extend the results of several methods for the analysis of hyperparameter importance (van Rijn and Hutter, 2018; Probst et al., 2018). Furthermore the collected data allows for further investigation on hyperparameter response surfaces. Meta-learning often assumes that knowledge for the optimization of one task can be transferred to others. This would require a certain degree of similarity across response surfaces, which can be empirically validated using our data. The runtime and memory measurement also allows to build models predicting resource requirements (c.f. Hutter et al. (2014)). This could be used to improve multi-fidelity approaches or to perform automated scheduling for parallel workflows in AutoML systems.

We make the generated data publicly available on OpenML, with IDs listed in Table 1. The code used to generate the the data is available online[2].

Acknowledgments

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V.[3] for funding this project by providing computing time on the GCS Supercomputer SuperMUC-NG at Leibniz Supercomputing Centre[4].

This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

2. `https://github.com/compstat-lmu/randombot_ng/`
3. `www.gauss-centre.eu`
4. `www.lrz.de`

Appendix A. Search Spaces

The following tables list the learning algorithms and their respective sampled hyperparameter bounds. Besides their natural hyperparameters, algorithms are also equipped with the `num.impute.selected.cpo` hyperparameter, which controls imputation for missing values in numeric features: mean, median, or histogram sampling imputation. All algorithms were used as implemented or interfaced to the R (R Core Team, 2019) programming language for statistics and the `mlr` framework for machine learning in R (Bischl et al., 2016), using the `mlrCPO` (Binder et al., 2020) set of composable preprocessing operators for preprocessing. "KerasFF" is a fully connected neural network implemented via `keras` (Chollet et al., 2015) with hyperparameters controlling the architecture (number of neurons and layers, magnitude of dropout, ...) and the optimizer (learning rate, weight decay, ...). We additionally vary the network's seed in order to obtain more reliable estimates with respect to randomness induced by different weight initializations. "RcppHNSW" is an approximate k-nearest neighbor implementation based on hierarchical navigable small world graphs (Malkov and Yashunin, 2020). All other learners directly interface existing implementations, information on their hyperparameters and meaning can be obtained from the respective software's documentation: xgboost: (Chen and Guestrin, 2016), Random Forest: (Wright and Ziegler, 2017), Elastic Net: (Friedman et al., 2010) and Decision Trees: (Therneau and Atkinson, 2018).

Hyperparameter	Range
epochs	$[2^3, 2^7](log)$
optimizer	sgd, rmsprop, adam
lr	$[5^{-5}, 5^0](log)$
decay	$[5^{-8}, 5^0](log)$
momentum	$[5^{-8}, 5^0](log)$
layers	[1, 4]
batchnorm_dropout	batchnorm, dropout, none
input_dropout_rate	$[3^{-5/2}, 3^0](log)$
dropout_rate	$[3^{-5/2}, 3^0](log)$
units_layer1	$[2^3, 2^9](log)$
units_layer2	$[2^3, 2^9](log)$
units_layer3	$[2^3, 2^9](log)$
units_layer4	$[2^3, 2^9](log)$
act_layer	relu, tanh
init_layer	glorot_normal, glorot_uniform, he_normal, he_uniform
l1_reg_layer	$[5^{-10}, 5^{-2}](log)$
l2_reg_layer	$[5^{-10}, 5^{-2}](log)$
learning_rate_scheduler	TRUE, FALSE
init_seed	1, 11, 101, 131, 499
num.impute.selected.cpo	impute.mean, impute.median, impute.hist

Table 2: Sample bounds of the KerasFF learning algorithm.

Hyperparameter	Range
k	[1, 50]
distance	l2, cosine, ip
M	[18, 50]
ef	$[2^3, 2^8](log)$
ef_construction	$[2^4, 2^9](log)$
num.impute.selected.cpo	impute.mean, impute.median, impute.hist

Table 3: Sample bounds of the RcppHNSW learning algorithm.

Hyperparameter	Range
nrounds	$[2^3, 2^{11}](log)$
eta	$[2^{-10}, 2^0](log)$
gamma	$[2^{-15}, 2^3](log)$
lambda	$[2^{-10}, 2^{10}](log)$
alpha	$[2^{-10}, 2^{10}](log)$
subsample	[0.1, 1]
max_depth	[1, 15]
min_child_weight	$[2^0, 2^7](log)$
colsample_bytree	[0.01, 1]
colsample_bylevel	[0.01, 1]
num.impute.selected.cpo	impute.mean, impute.median, impute.hist

Table 4: Sample bounds of the XGBoost learning algorithm.

Hyperparameter	Range
num.trees	[1, 2000]
replace	TRUE, FALSE
sample.fraction	[0.1, 1]
mtry.power	[0, 1]
respect.unordered.factors	ignore, order, partition
min.node.size	[1, 100]
splitrule	gini, extratrees
num.random.splits	[1, 100]
num.impute.selected.cpo	impute.mean, impute.median, impute.hist

Table 5: Sample bounds of the Ranger (random Forest) learning algorithm.

Hyperparameter	Range
kernel	linear, polynomial, radial
cost	$[2^{-12}, 2^{12}](log)$
gamma	$[2^{-12}, 2^{12}](log)$
degree	[2, 5]
tolerance	$[2^{-12}, 2^{-3}](log)$
shrinking	TRUE, FALSE
num.impute.selected.cpo	impute.mean, impute.median, impute.hist

Table 6: Sample bounds of the SVM learning algorithm.

Hyperparameter	Range
cp	$[2^{-10}, 2^{0}](log)$
maxdepth	[1, 30]
minbucket	[1, 100]
minsplit	[1, 100]
num.impute.selected.cpo	impute.mean, impute.median, impute.hist

Table 7: Sample bounds of the RPART (decision tree) learning algorithm.

Hyperparameter	Range
alpha	[0, 1]
s	$[2^{-10}, 2^{10}](log)$
num.impute.selected.cpo	impute.mean, impute.median, impute.hist

Table 8: Sample bounds of the glmnet (elastic net) learning algorithm.

Appendix B. Preprocessing

We preprocess data using `mlrCPO` using the following procedure:

1. fixfactors: Set all categorical features missing not present during training to MISSING in the prediction phase
2. imputation: Impute using a new level for categorical features and mean, median or histogram for numerics. The latter is a hyperparameter exposed for all learners and explored during the sampling procedure.
3. We add indicator columns for missing numeric values.
4. We limit the number of factor levels for a given categorical variable to 32. All other columns with lower cardinality are collapsed to a "other" category.
5. We drop constant features.
6. If learners can not handle categorical features natively (xgboost, keras, rcpphnsw), we encode those using dummy encoding.

Appendix C. Subsampling

We use subsampling with the following factions of the training data:
$0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$

References

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

Martin Binder, Lars Kotthoff, Michel Lang, and Bernd Bischl. *mlrCPO: Composable Pre-processing Operators and Pipelines for Machine Learning*, 2020. R package version 0.3.6.

B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. mlr: Machine learning in R. *JMLR*, 17(170):1–5, 2016.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. Openml benchmarking suites and the openml100. *arXiv preprint arXiv:1708.03731*, 2017.

Pavel Brazdil, Christophe Giraud Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to data mining.* Springer Science & Business Media, 2008.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2.

François Chollet et al. Keras. `https://keras.io`, 2015.

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.

Katharina Eggensperger, Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:1–21, 2019.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, and Joaquin Vanschoren. An open source automl benchmark. *arXiv preprint arXiv:1907.00909*, 2019.

Frank Hutter, Lin Xu, Holger H Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

Aaron Klein, Zhenwen Dai, Frank Hutter, Neil Lawrence, and Javier Gonzalez. Meta-surrogate benchmarking for hyperparameter optimization. In *Advances in Neural Information Processing Systems*, pages 6270–6280, 2019.

Daniel Kühn, Philipp Probst, Janek Thomas, and Bernd Bischl. Automatic Exploration of Machine Learning Experiments on OpenML. *arXiv preprint arXiv:1806.10961*, 2018.

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.

Yury A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020.

Luke Metz, Niru Maheswaranathan, Ruoxi Sun, C Daniel Freeman, Ben Poole, and Jascha Sohl-Dickstein. Using a thousand optimization tasks to learn hyperparameter search strategies. *arXiv preprint arXiv:2002.11887*, 2020.

Florian Pfisterer, Jan N. van Rijn, Philipp Probst, Andreas M. Müller, and Bernd Bischl. Learning Multiple Defaults for Machine Learning Algorithms. *arXiv preprint arXiv:1811.09409*, 2018.

P. Probst, B. Bischl, and A. Boulesteix. Tunability: Importance of hyperparameters of machine learning algorithms. *arXiv preprint arXiv:1802.09596*, 2018.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2019.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, USA, 2012. Curran Associates Inc.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

Terry Therneau and Beth Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2018. R package version 4.1-13.

Jan N. van Rijn and Frank Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2367–2376. ACM, 2018.

Jan N van Rijn, Florian Pfisterer, Janek Thomas, Andreas Muller, Bernd Bischl, and Joaquin Vanschoren. Meta learning for defaults–symbolic defaults. In *Neural Information Processing Workshop on Meta-Learning*, 2018.

Joaquin Vanschoren. Meta-learning. In *Automated Machine Learning*, pages 35–61. Springer, Cham, 2019.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization. In *Proc. of ECML/PKDD 2015*, pages 104–119. Springer, 2015a.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE, 2015b.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

Marvin N Wright and Andreas Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017.

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, Long Beach, California, USA, 2019. PMLR.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.