NAS-Bench-301 and the Case for Surrogate Benchmarks for Neural Architecture Search

Julien Siems[1*], **Lucas Zimmer**[1*], **Arber Zela**[1], **Jovita Lukasik**[2],
Margret Keuper[2] **& Frank Hutter**[1,3]
[1]University of Freiburg, [2]University of Mannheim, [3]Bosch Center for Artificial Intelligence
{siemsj, zimmerl, zelaa, fh}@cs.uni-freiburg.de
{jovita, keuper}@uni-mannheim.de

Abstract

Several tabular NAS benchmarks have been proposed to simulate runs of NAS methods in seconds in order to allow scientifically sound empirical evaluations. However, all existing tabular NAS benchmarks are limited to extremely small architectural spaces since they rely on exhaustive evaluations of the space. This leads to unrealistic results that do not transfer to larger search spaces. Motivated by the fact that similar architectures tend to yield comparable results, we propose NAS-Bench-301 which covers a search space many orders of magnitude larger than any previous NAS benchmark. We achieve this by meta-learning a performance predictor that predicts the capability of different neural architectures to facilitate base-level learning, and using it to define a surrogate benchmark. We fit various regression models on our dataset, which consists of $\sim$60k architecture evaluations, and build surrogates via deep ensembles to also model uncertainty. We benchmark a wide range of NAS algorithms using NAS-Bench-301 and obtain comparable results to the true benchmark at a fraction of the real cost.

1 Introduction

Despite many advancements in terms of both efficiency and performance, empirical evaluations in Neural Architecture Search (NAS) are still problematic. Different NAS papers often use different training pipelines, different search spaces and different hyperparameters, do not evaluate other methods under comparable settings, and cannot afford enough runs for testing significance. This practice impedes assertions about the statistical significance of reported results, recently brought into focus by several authors (Yang et al., 2019; Lindauer & Hutter, 2019; Shu et al., 2020; Yu et al., 2020).

To circumvent these issues and enable scientifically sound evaluations in NAS, several tabular benchmarks (Ying et al., 2019; Zela et al., 2020b; Dong & Yang, 2020; Klyuchnikov et al., 2020) have been proposed recently (see also Appendix A.1 for more details). However, all these benchmarks rely on an exhaustive evaluation of *all* architectures in a search space, which limits them to unrealistically small search spaces (containing between 6k and 423k architectures). This is a far shot from standard spaces used in the NAS literature, which contain more than 10^{18} architectures (Zoph & Le, 2017; Liu et al., 2019). This discrepancy can cause results gained on existing tabular NAS benchmarks to not generalize to realistic search spaces; e.g., promising anytime results of local search on existing tabular NAS benchmarks were shown to not transfer to realistic search spaces (White et al., 2020).

To address these problems, we present *NAS-Bench-301*, a surrogate NAS benchmark that is first to cover a realistically-sized search space (namely the cell-based search space of DARTS (Liu et al., 2019)), containing more than 10^{18} possible architectures. This is made possible by meta-learning the

[*] Equal contribution

4th Workshop on Meta-Learning at NeurIPS 2020, Vancouver, Canada.

performance of neural architectures. That is, we meta-learn a surrogate model across neural architectures that captures the capability of these architectures to facilitate base-level learning. We then exploit this meta-model to define a benchmark. Specifically, we make the following contributions:

1. We empirically demonstrate that a surrogate fitted on a subset of architectures can in fact model the true performance of architectures *better* than a tabular benchmark (Section 2).
2. We analyze and release the NAS-Bench-301 training dataset consisting of ∼60k fully trained and evaluated architectures, which will also be publicly available in the Open Graph Benchmark (Hu et al., 2020) (Section 3).
3. Using this dataset, we thoroughly evaluate a variety of regression models as surrogate candidates, showing that strong generalization performance is possible even in large spaces (Section 4).
4. We utilize NAS-Bench-301 as a benchmark for running various NAS optimizers and show that the resulting search trajectories closely resemble the ground truth trajectories. This enables sound simulations of thousands of GPU hours in a few seconds on a single CPU machine (Section 5).
5. We demonstrate that NAS-Bench-301 can help in generating new scientific insights (Section 6).

To foster reproducibility, we open-source all our code and data in a public repo: `https://github.com/automl/nasbench301`.

2 Motivation – Can we do Better Than a Tabular Benchmark?

We start by motivating the use of surrogate benchmarks by exposing an issue of tabular benchmarks that has largely gone unnoticed. Tabular benchmarks are built around a costly, exhaustive evaluation of *all* possible architectures in a search space, and when an architecture's performance is queried, the tabular benchmark simply returns the respective table entry. The issue with this process is that the stochasticity of mini-batch training is also reflected in the performance of an architecture i, hence making it a random variable Y_i. Therefore, the table only contains results of a few draws $y_i \sim Y_i$ (existing NAS benchmarks feature up to 3 runs per architecture). Given the variance in these evaluations, a tabular benchmark acts as a simple estimator that assumes *independent* random variables, and thus estimates the performance of an architecture based only on previous evaluations of the same architecture. From a machine learning perspective, knowing that similar architectures tend to yield similar performance, and that the variance of individual evaluations can be high (both shown to be the case by Ying et al. (2019)), it is natural to assume that better estimators may exist. In the remainder of this section, we empirically verify this hypothesis and show that surrogate benchmarks can provide *better* performance estimates than tabular benchmarks based on *less* data.

Setup We choose NAS-Bench-101 (Ying et al., 2019) as a tabular benchmark for our analysis and a Graph Isomorphism Network (GIN, Xu et al. (2019a)) as our surrogate model.[2] Each architecture x_i in NAS-Bench-101 contains 3 validation accuracies y_i^1, y_i^2, y_i^3 from training x_i with 3 different seeds. We excluded all diverged models with less than 50% validation accuracy on any of the three evaluations in NAS-Bench-101. We split this dataset to train the GIN surrogate model on one of the seeds, e.g., $\mathcal{D}^{train} = \{(x_i, y_i^1)\}_i$ and evaluate on the other two, e.g., $\mathcal{D}^{test} = \{(x_i, \bar{y}_i^{23})\}_i$, where $\bar{y}_i^{23} = (y_i^2 + y_i^3)/2$.

Model	Mean Absolute Error (MAE)		
	1, [2, 3]	2, [1, 3]	3, [1, 2]
Tab.	4.534e-3	4.546e-3	4.539e-3
Surr.	**3.446e-3**	**3.455e-3**	**3.441e-3**

Table 1: MAE between performance predicted by a tab./surr. benchmark fitted with one seed each, and the true performance of evaluations with the two other seeds. Test seeds in brackets.

We emphasize that training a surrogate to model a search space is not a typical inductive regression task but rather a transductive one. By definition of the search space, the set of possible architectures is known ahead of time (although it may be very large), hence a surrogate model does not have to generalize to out-of-distribution data if the training data covers the space well.

Results We compute the mean absolute error MAE $= \frac{\sum_i |\hat{y}_i - \bar{y}_i^{23}|}{n}$ of the surrogate model trained on $\mathcal{D}^{train} = \{(x_i, y_i^1)\}_i$, where $\hat{y}_i$ is predicted validation accuracy and $n = |\mathcal{D}^{test}|$. Table 1 shows that the surrogate model yields a lower MAE than the tabular benchmark, i.e. MAE $= \frac{\sum_i |y_i^1 - \bar{y}_i^{23}|}{n}$.

[2] We used a GIN implementation by Errica et al. (2020); see Appendix B for details on training the GIN.

We repeat the experiment in a cross-validation fashion w.r.t to the seeds and conclude: *In contrast to a single tabular entry, the surrogate model learns to smooth out the noise.*[3]

Next, we fit the GIN surrogate on subsets of $\mathcal{D}^{train}$ and plot how its performance scales with the amount of training data used in Figure 1. The surrogate model performs better than the tabular benchmark when the training set has more than ∼21,500 architectures. Note that $\mathcal{D}^{test}$ remains the same as in the previous experiment, i.e., it includes all architectures in NAS-Bench-101. As a result, we conclude that: *A surrogate model can yield strong predictive performance when only a subset of the search space is available as training data.*

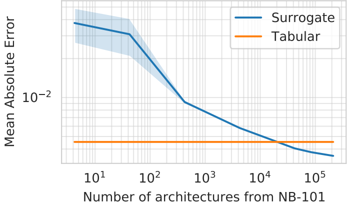


Figure 1: Number of architectures used for training the GIN surrogate model vs MAE on the NAS-Bench-101 dataset.

These empirical findings suggest that we can create reliable surrogate benchmarks for much larger and more realistic NAS spaces, which are infeasible to be exhaustively evaluated as done by tabular benchmarks. In the remainder of the paper, we focus on creating such a benchmark.

3 The NAS-Bench-301 Dataset

We now describe the *NAS-Bench-301 dataset* which consists of ∼60k architectures and their performances on CIFAR-10 (Krizhevsky, 2009) sampled from the most popular NAS cell search space: the one from DARTS (Liu et al., 2019). We use this dataset not only to fit surrogate models but also to gain new insights, such as which regions of the architecture space are being explored by different NAS methods, or what the characteristics of architectures are that work well.

	NAS methods	**# eval**
	RS (Bergstra & Bengio, 2012)	23746
Evolution	DE (Awad et al., 2020)	7275
	RE (Real et al., 2019)	4639
BO	TPE (Bergstra et al., 2011)	6741
	BANANAS (White et al., 2019)	2243
	COMBO (Oh et al., 2019)	745
One-Shot	DARTS (Liu et al., 2019)	2053
	PC-DARTS (Xu et al., 2020)	1588
	DrNAS (Chen et al., 2020)	947
	GDAS (Dong & Yang, 2019)	234

Table 2: NAS methods used to cover the search space.

3.1 Data Collection

Since the DARTS search space (detailed description in Appendix C.1) is far too large to be exhaustively evaluated, care has to be taken when sampling the architectures which will be used to train the surrogate models. Sampling should yield a good overall coverage of the architecture space while also providing a special focus on the well-performing regions that optimizers tend to exploit.

Our principal methodology is inspired by Eggensperger et al. (2015), who collected unbiased data about hyperparameter spaces by random search, as well as biased and dense samples in high-performance regions by running hyperparameter optimizers. This is desirable for a surrogate benchmark since we are interested in evaluating NAS methods that exploit such good regions of the space. Table 2 lists the NAS methods we used to collect such samples and the respective number of samples. Additionally, we evaluated ∼1k architectures in poorly-performing regions for better coverage and another ∼10k for the analysis conducted on the dataset and surrogates. We refer to Appendix C.2 for details on the data collection and the optimizers.

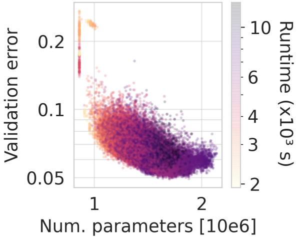


Figure 2: Number of parameters against val. error with model training time as colorbar.

3.2 Performance statistics

Figure 2 shows the validation error on CIFAR-10 (Krizhevsky, 2009) of all sampled architectures in relation to the model parameters and training runtime. Generally, as expected, models with more parameters are more costly to train but achieve lower validation errors. We also find that different NAS methods yield quite different performance distributions (see Appendix C.3 for their individual performances). Validation and test errors are highly correlated with a Kendall tau rank correlation of $\tau = 0.852$ (Spearman rank corr. 0.969), minimizing the risk of overfitting on the validation error.

[3] We do note that the average estimation error of tabular benchmarks could be reduced by a factor of $\sqrt{k}$ by performing k runs for each architecture. The error of a surrogate model would also shrink when the model is based on more data, but as k grows large tabular benchmarks would become competitive with surrogate models.

Furthermore, we find that cells of all depths can reach a good performance, but shallow topologies are slightly favored in our setting (see Figure 9 in the Appendix). Also, a small number of parameter-free operations (e.g., skip connections) can benefit the performance but featuring many of these significantly deteriorates performance. For the full analysis, see Appendix C.4. Following standard practice in modern NAS papers (e.g., Liu et al. (2019)), we employ various data augmentation techniques during training for more reliable estimates of an architecture's performance. For a description of our full training pipeline, please see Appendix C.6. Finally, we found that the variance of multiple evaluations of the same architecture in our setting is lower than for NAS-Bench-101 (Details in Section C.5 in the Appendix).

4 Fitting Surrogate Models on the NAS-Bench-301 Dataset

We now focus on creating a surrogate model. To that end, we evaluated a wide range of regression models on the NAS-Bench-301 dataset. In principle, any such model can give rise to a surrogate NAS benchmark, but models that fit the true performance better yield surrogate NAS benchmarks whose characteristics are more similar to the ones of the true benchmark. Therefore, we naturally strive for the best-fitting model. We emphasize that in this work we do not attempt to introduce a new regression model but rather build on the shoulders of the architecture performance prediction community.

4.1 Surrogate Model Candidates

Deep Graph Convolutional Neural Networks are frequently used as NAS predictors (Friede et al., 2019; Wen et al., 2019; Ning et al., 2020). In particular, we choose the GIN since several works have found it to perform well on many benchmark datasets (Errica et al., 2020; Hu et al., 2020; Dwivedi et al., 2020). We use the publicly available implementation from the Open Graph Benchmark (Hu et al., 2020) and refer to Appendix D.2 for further details. We compare the GIN to a variety of common regression models. We evaluate Random Forests (RF) and Support Vector Regression (SVR) using implementations from scikit-learn (Pedregosa et al., 2011). We also compare to the tree-based gradient boosting methods XGBoost (Chen & Guestrin, 2016). LGBoost (Ke et al., 2017) and NGBoost (Duan et al., 2020), recently used for predictor-based NAS (Luo et al., 2020). We comprehensively review architecture performance prediction in Appendix A.2.

Model	Test	
	R^2	sKT
LGBoost	**0.892**	0.816
XGBoost	0.832	**0.817**
GIN	0.832	0.778
NGBoost	0.810	0.759
μ-SVR	0.709	0.677
RF	0.679	0.683
ϵ-SVR	0.675	0.660

Table 3: Performance of different regression models fitted on the NB-301 dataset.

4.2 Evaluation

We assess the quality of the data fit via the coefficient of determination (R^2) and the sparse Kendall τ (sKT) rank correlation, a variant proposed by Yu et al. (2020) which ignores rank changes at 0.1% accuracy precision by rounding the predicted validation accuracy prior to computing τ. We provide additional details on the preprocessing of the architectures for the surrogate models and the Hyperparameter Optimization in Appendices D.1 and D.3 respectively.

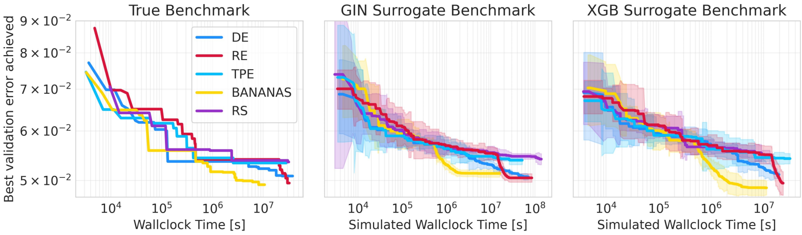
As Table 3 shows, the three best-performing models are LGBoost, XGBoost and GIN; we therefore focus our analysis on these in the following. In addition to evaluating the data fit on our data splits, we investigate the impact of parameter-free operations Appendix D.6. We find that all of LGBoost, XGBoost and GIN accurately predict the drop in performance when increasingly replacing operations with parameter-free operations in a normal cell. Following Eggensperger et al. (2015) we perform a cross validation on the optimizers in Section D.5 in the Appendix and find that the XGB and GIN extrapolate to 'unseen' optimizers.

4.3 Noise Modelling

Model	MAE 1, [2,3,4,5]	Mean σ	KL div.
Tabular	1.38e−3	undef.	undef.
GIN	**1.13e-3**	0.6e−3	**16.4**
LGB	1.33e−3	0.3e−3	68.9
XGB	1.51e−3	0.3e−3	134.4

Table 4: Metrics for the selected surrogate models on 500 architectures that were evaluated 5 times.

Ensemble methods are commonly used to improve predictive performance (Dietterich, 2000). Moreover, ensembles of deep neural networks, so-called deep ensembles, have been proposed as a simple way to obtain predictive uncertainty (Lakshminarayanan et al., 2017). We therefore create an ensemble of 10 base learners for each of our three best performing models (GIN, XGB, LGB) using a 10-fold cross-validation for our train and validation split, as well as different


True Benchmark

GIN Surrogate Benchmark

XGB Surrogate Benchmark

Figure 3: Anytime performance of different optimizers on the real benchmark (left) and the surrogate benchmark (GIN (middle) and XGB (right)) when training ensembles on data collected from all optimizers. Trajectories on the surrogate benchmark are averaged over 5 optimizer runs.

initializations. We use the architectures with multiple evaluations (see Section C.5 in the Appendix) to mirror the analysis in the motivation in Section 2. We train using only one evaluation per architecture (i.e., seed 1) and take the mean accuracy of the remaining ones as groundtruth (i.e., seeds 2-5). We then compare against a tabular model with just one evaluation (seed 1).

Table 4 shows that the GIN and LGB surrogate models yield estimates closer to groundtruth than the table lookup based on one evaluation. This confirms our main finding from Section 2, but this time on a much larger search space. We also compare the predictive distribution of our ensembles to the groundtruth. To that end, we assume the noise in the architecture performance to be normally distributed and compute the Kullback–Leibler (KL) divergence between the groundtruth accuracy distribution and predicted distribution. We find the GIN ensemble to quite clearly provide the best estimate. To allow evaluations of multi-objective NAS methods, and to allow using "simulated wallclock time" on the x axis of plots, we also predict the runtime of architecture evaluations. For this, we train an LGB model with the runtime as targets (see Appendix D.4 for details). Runtime prediction is less challenging than performance prediction, resulting in an excellent fit of our LGB runtime model on the test set (sKT: 0.936, R^2: 0.987). Other metrics of architectures, such as the number of parameters and multiply-adds, do not require a surrogate model but can be queried exactly.

5 NAS-Bench-301 as a Surrogate NAS Benchmark

Having assessed the ability of the surrogate models to model the search space, we now use NAS-Bench-301 to benchmark various NAS algorithms.

5.1 Blackbox Optimizers

We first compare the trajectories on the true benchmark and on the surrogate benchmark for blackbox optimizers when training the surrogate on all data. For the true benchmark, we show the trajectories contained in our dataset (based on a single run, since we could not afford repetitions due to the extreme compute requirements of 115 GPU days for a single run). For the evaluations on the surrogate, on the other hand, we can trivially afford to perform multiple runs. For the surrogate trajectories, we use an identical initialization for the optimizers (e.g., initial population for RE) but evaluations of the surrogate benchmark are done by sampling from the surrogate model's predictive distribution for the architecture at hand, leading to different trajectories.

Results (all data) As Figure 3 shows, both the XGB and the GIN surrogate capture behaviors present on the true benchmark. For instance, the strong improvements of BANANAS and RE are also present on the surrogate benchmark at the correct time. In general, the ranking of the optimizers towards convergence is accurately reflected on the surrogate benchmark. Also, the initial random exploration of algorithms like TPE, RE and DE is captured as the large initial variation in performance indicates. Notably, the XGB surrogate ensemble exhibits a high variation in well-performing regions as well and seems to slightly underestimate the error of the best architectures. The GIN surrogate, on the other hand, shows less variance in these regions but slightly overpredicts for the best architectures.

Note, that due to the size of the search space, random search stagnates and cannot identify one of the best architectures even after tens of thousands of evaluations, with BANANAS finding better architectures orders of magnitude faster. This stands in contrast to previous NAS benchmarks. For instance, NAS-Bench-201 (Dong & Yang, 2020) only contains 6466 unique architectures in total, causing the median of random search runs to find the best architecture after only 3233 evaluations.

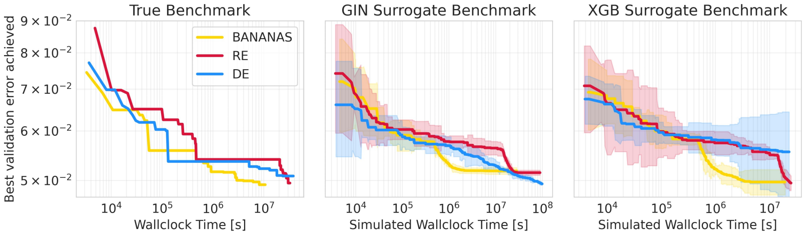


Figure 4: Anytime performance of blackbox optimizers, comparing performance achieved on the real benchmark and on surrogate benchmarks built with GIN and XGB in an LOOO fashion.

To simulate benchmarking of novel NAS methods, we expand on the leave-one-optimizer-out analysis (LOOO) from Section D.5 in the Appendix and assess each optimizer with surrogate benchmarks based on data excluding that gathered by said optimizer. We again compare the trajectories obtained from 5 runs on the surrogate benchmark to the groundtruth.

Results (LOOO) Figure 4 shows the trajectories in the leave-one-optimizer-out setting. The XGB and GIN surrogates again capture the general behavior of different optimizers well, illustrating that characteristics of new optimization algorithms can be captured with the surrogate benchmark. Leaving out DE appears to be a bigger problem for XGB than GIN, pointing to advantages of the smooth embedding learned by the GIN compared to gradient-boosting.

Besides discrete optimizers, we also evaluated One-Shot optimizers and failure cases of them in setting in Section E.1 and E.2 in the Appendix.

6 Using NAS-Bench-301 to Drive NAS Research

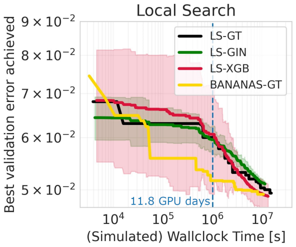


Figure 5: Case study results for Local Search. GT is the ground truth, GIN and XGB are results on NAS-Bench-301.

We finally use our new benchmark to perform a case study that demonstrates how NAS-Bench-301 can drive NAS research. Coming up with research hypotheses and drawing conclusions when prototyping or evaluating NAS algorithms on less realistic benchmarks is difficult, particularly when these evaluations require high computational budgets. NAS-Bench-301 alleviates this dilemma via its cheap and reliable estimates. To showcase such a scenario, we evaluate Local Search[4] (LS) on our surrogate benchmark and the actual DARTS benchmark. White et al. (2020) concluded that LS does not perform well on such a large space by running it for 11.8 GPU days ($\approx 10^6$ seconds), and we are able to reproduce the same results via NAS-Bench-301 in a few seconds (see Fig 5). While White et al. (2020) could not afford longer runs (nor repeats), on NAS-Bench-301 this is trivial. Doing so suggests that LS shows qualitatively different behavior when run for an order of magnitude longer, transitioning from being the worst method to being one of the best. We verified this suggestion by running LS for longer on the actual DARTS benchmark (also see Fig 5). This allows us to revise the initial conclusion of White et al. (2020) to: *LS is also state-of-the-art for the DARTS search space, but only when given enough time.*

This case study shows how NAS-Bench-301 was already used to cheaply obtain hints on a research hypothesis that lead to correcting a previous finding that only held for short runtimes. We look forward to additional uses along such lines.

7 Conclusions

We proposed NAS-Bench-301, the first surrogate NAS benchmark and first to cover a realistic search space which is orders of magnitude larger than all previous tabular NAS benchmarks. This is made possible by learning a meta-level surrogate model that predicts the base-level learning performance of neural architectures, and using this surrogate model to define a benchmark. After motivating the benefits of a surrogate benchmark over a tabular one, we described the strategy used to collect the data which we used to fit our selected surrogate models and benchmarked their predictive performance. Lastly, we demonstrated that our surrogate benchmark can accurately simulate real

[4] We use the implementation and settings of Local Search provided by White et al. (2020).

anytime performance trajectories of various NAS methods at a fraction of the true cost and can lead to new scientific findings. We hope that NAS-Bench-301 will equip the NAS practitioner with a cheap, yet realistic benchmark for prototyping novel NAS methods and allow fast benchmarking. Due to the flexibility of surrogate NAS benchmarks to cover arbitrary search spaces, we expect NAS-Bench-301 to be the first of many such benchmarks. We collect best practices for the usage and creation of new surrogate benchmarks in Appendices G and F.

References

N. Awad, N. Mallik, and F. Hutter. Differential evolution for neural architecture search (dehb). In *International Conference on Machine Learning (ICLR) Neural Architecture Search (NAS) Workshop*, 2020.

B. Baker, O. Gupta, R. Raskar, and N. Naik. Accelerating Neural Architecture Search using Performance Prediction. In *NeurIPS Workshop on Meta-Learning*, 2017.

R. Baptista and M. Poloczek. Bayesian optimization of combinatorial structures. In *International Conference on Machine Learning (ICLR)*, pp. 462–471, 2018.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger (eds.), *Proceedings of the 25th International Conference on Advances in Neural Information Processing Systems (NIPS'11)*, pp. 2546–2554, 2011.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

X. Chen, R. Wang, M. Cheng, X. Tang, and C. Hsieh. Drnas: Dirichlet neural architecture search. *arXiv preprint arXiv:2006.10355*, 2020.

P. Chrabaszcz, I. Loshchilov, and F. Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

B. Deng, J. Yan, and D. Lin. Peephole: Predicting network performance before training. *arXiv preprint arXiv:1712.03351*, 2017.

T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

T. Dietterich. *Ensemble Methods in Machine Learning*, volume 1857 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000.

T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In Q. Yang and M. Wooldridge (eds.), *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pp. 3460–3468, 2015.

X. Dong and Y. Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1761–1770, 2019.

X. Dong and Y. Yang. Nas-bench-102: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

T. Duan, A. Avati, D. Ding, Khanh K. Thai, S. Basu, A. Ng, and A. Schuler. Ngboost: Natural gradient boosting for probabilistic prediction. In *Proceedings of Machine Learning and Systems 2020*, pp. 6138–6148, 2020.

V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

K. Eggensperger, F. Hutter, H.H. Hoos, and K. Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In B. Bonet and S. Koenig (eds.), *Proceedings of the Twenty-nineth National Conference on Artificial Intelligence (AAAI'15)*, pp. 1114–1120. AAAI Press, 2015.

F. Errica, M. Podda, D. Bacciu, and A. Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020.

S Falkner, A Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1437–1446, 2018.

M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

D. Friede, J. Lukasik, H. Stuckenschmidt, and M. Keuper. A variational-sequential graph autoencoder for neural architecture performance prediction. *ArXiv*, abs/1912.05317, 2019.

D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

J. Hwang, S. Lay, and A. Lippman. Nonparametric multivariate density estimation: a comparative study. *IEEE Transactions on Signal Processing*, 42(10):2795–2810, 1994.

R. Istrate, F. Scheidegger, G. Mariani, D. Nikolopoulos, C. Bekas, and A. C. I. Malossi. Tapas: Train-less accuracy predictor for architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3927–3934, 2019.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 3146–3154. Curran Associates, Inc., 2017.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter. Learning curve prediction with Bayesian neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*. CBLS, 2017.

N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, and E. Burnaev. Nas-bench-nlp: Neural architecture search benchmark for natural language processing. *arXiv preprint arXiv:2006.07116*, 2020.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pp. 6402–6413, 2017.

G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *International Conference on Learning Representations (ICLR)*, 2017.

M. Lindauer and F. Hutter. Best practices for scientific research on neural architecture search. *arXiv preprint arXiv:1909.02453*, 2019.

M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, J. Marben, P. Müller, and F. Hutter. Boah: A tool suite for multi-fidelity bayesian optimization & analysis of hyperparameters. *arXiv:1908.06756 [cs.LG]*, 2019.

H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR) 2017 Conference Track*, April 2017.

R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T. Liu. Neural architecture search with gbdt. *arXiv preprint arXiv:2007.04785*, 2020.

M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1): 55–61, 2000.

X. Ning, Y. Zheng, T. Zhao, Y. Wang, and H. Yang. A generic graph-based neural architecture encoding scheme for predictor-based nas. *arXiv preprint arXiv:2004.01899*, 2020.

C. Oh, J. Tomczak, E. Gavves, and M. Welling. Combinatorial bayesian optimization using the graph cartesian product. In *Advances in Neural Information Processing Systems*, pp. 2910–2920, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

K. Price, R. M. Storn, and J. A. Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

H. Shi, R. Pi, H. Xu, Z. Li, J. T. Kwok, and T. Zhang. Multi-objective neural architecture search via predictive network performance optimization. *arXiv preprint arXiv:1911.09336*, 2019.

Y. Shu, W. Wang, and S. Cai. Understanding architectures learnt by cell-based neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

I. M. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

Y. Tang, Y. Wang, Y. Xu, H. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu. A semi-supervised assessor of neural architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 9(Nov):2579–2605, 2008.

W. Wen, H. Liu, H. Li, Y. Chen, G. Bender, and P. Kindermans. Neural predictor for neural architecture search. *arXiv preprint arXiv:1912.00848*, 2019.

C. White, W. Neiswanger, and Y. Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. *arXiv preprint arXiv:1910.11858*, 2019.

C. White, S. Nolen, and Y. Savani. Local search is state of the art for nas benchmarks. *arXiv preprint arXiv:2005.02960*, 2020.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019a.

Y. Xu, Y. Wang, K. Han, H. Chen, Y. Tang, S. Jui, C. Xu, Q. Tian, and C. Xu. Rnas: Architecture ranking for powerful networks. *arXiv preprint arXiv:1910.01523*, 2019b.

Y. Xu, L. Xie, X. Zhang, X. Chen, G. Qi, Q. Tian, and H. Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2020.

A. Yang, P. M. Esperança, and F. M. Carlucci. Nas evaluation is frustratingly hard. *arXiv preprint arXiv:1912.12522*, 2019.

C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter. NAS-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7105–7114, 2019.

K. Yu, R. Ranftl, and M. Salzmann. How to train your super-net: An analysis of training heuristics in weight-sharing nas. *arXiv preprint arXiv:2003.04276*, 2020.

A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020a.

A. Zela, J. Siems, and F. Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020b.

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

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