Beyond the Threshold: Time Is All You Need

Stefan Dendorfer[1] **Andreas M. Kist**[1]

[1]Friedrich-Alexander-Universität Erlangen-Nürnberg

Abstract This study examines common misconceptions and suboptimal use of tabular benchmarks for neural architecture search (NAS). We address statistical limitations in performance evaluation, emphasizing adequate sample sizes and proper statistical tests, such as the two-sample t-test, to ensure reliable results. We propose a new guideline of averaging at least 1000 runs for reliably benchmarking NAS algorithms. Additionally, we explore the impact of time constraints on algorithm performance, showing that final algorithm performance highly depends on the time budget. Our findings highlight the need for robust experimental designs and extended time budgets in NAS research.

1 Introduction

Neural architecture search (NAS) is a technique to optimize neural architectures towards an objective, such as classification accuracy or model size, and can be performed using a variety of optimization procedures, among others reinforcement learning (Zoph and Le, 2016) and evolutionary optimization (Groh and Kist, 2023). Recently, NAS benchmarks were introduced to provide a standardized framework enhancing reproducibility, accessibility, and comparability of different NAS methods (Ying et al., 2019). Tabular benchmarks (Klein and Hutter, 2019) optimize evaluation time by leveraging precomputed data, allowing researchers to focus on algorithm development and enabling statistically significant performance comparisons. However, misconceptions and suboptimal usage, particularly in statistical analysis and experiment termination, hinder research progress. Proper statistical testing methods, like the two-sample t-test, are vital for reliable performance evaluations, while premature experiment termination can obscure valuable algorithm insights. Our study addresses these challenges and proposes methods to enhance NAS research reproducibility and evaluation robustness. The code required to reproduce our results and findings is publicly available at `https://github.com/sdnfr/time-bench`.

2 Underlying Foundations

We use the terminology of *runs* and *experiments*. A given experiment consists of n runs performing the neural architecture search in the given search space within a given time budget t. While samples in NAS commonly refer to sampled architectures from the search space, we refer to one entire NAS run as a sample for statistical analysis of our experiments. Each experiment is associated with its corresponding sample mean $\bar{s}$ and uncontrolled (biased) sample standard deviation σ_n. The latter is used to show the variance of each experiment without further interpretations or control of estimating the underlying population standard deviation s_n. The notation here distinguishes the corrected sample standard deviation s_n from the uncorrected sample standard deviation σ_n. The former assumes n as a sample of a population by having the $(n-1)$ term in the denominator, while the latter assumes n to be the whole population thus dividing by n. The objective of evaluating the performance of multiple runs is to gain insights into the "true" performance of the algorithm, such as the population mean and population standard deviation. Practically, we can only perform a finite number of runs n of the unknown population, thus the evaluation of an experiment becomes a sampling problem.

The central limit theorem, for large enough n, states that even if the original distribution of an experiment is not normally distributed, the sampling distribution of the sample means is normally distributed (Krzywinski and Altman, 2013). This distribution measures the spread of sample means $\sigma_{\bar{s}}$, indicating where the sample mean of succeeding experiments will lie.

While there is a fair bit of chance in NAS runs (random unfavorable initial populations (Maaranen et al., 2007), stuck in local optima (Barbulescu et al., 2000), random mutation), statistical analysis for the results becomes crucial to avoid potentially reporting erroneous results. While we do not assume any intentional falsification of results, we aim to raise awareness for more robust scientific practices.

3 Mitigating Statistical Misuse

Table 1 shows the p-value of two-sample t-tests[1] as stated in the CIFAR-10 results from the topology search on the NATS-Bench using 5 runs (Fan et al., 2023). We corrected the standard deviation and adjusted for equal variances accordingly (Welch, 1947). The data and formula can be found in the supplied code notebook `n1`. The depicted top 5 algorithms are ordered by performance and the table shows that algorithms with similar performance do not show statistically significant results due to the low number of runs (n=5).

Table 1: P-values of two-sample t-test with top 5 algorithms from Fan et al. (2023). Algorithms ordered by performance. Significance is indicated with an asterisk (*), significance level is set to 0.05/10 (Bonferroni correction).

	LayerNAS	RE	PPO	TE-NAS	NASI
LayerNAS	-	0.12	0.012	0.17	1.8e-05*
RE	0.12	-	0.4	0.45	0.001*
PPO	0.012	0.4	-	0.68	0.0009*
TE-NAS	0.17	0.45	0.68	-	0.26
NASI	1.8e-05*	0.001*	0.0009*	0.26	-

Publications can go as low as experiments with 5 runs (Fan et al., 2023) or 100 runs (Ying et al., 2019), even though about 500 runs are recommended in the literature (Ying et al., 2019). Besides the implied weak statistical significance, this can cause the problem of unintentional cherry-picking which leads to the common phenomenon of p-hacking (Head et al., 2015), even though we would like to stress that this might happen fully unintentionally. While rerunning an experiment takes little time, experimental results are often not carefully saved as compared to when dealing with computational heavy loads. Hyperparameters can be quickly evaluated, resulting in multiple experiment repetitions which increases the potential for statistical error.

Assuming the results of a first experiment with n samples approximate the true population mean and standard deviation, the central limit theorem can be applied when running another independent experiment a second time. Consequently, there is a 50% chance that the new sample mean will be higher than the previous sample mean, and vice versa, as the new sample mean will be drawn from the underlying Gaussian distribution with the properties mentioned above. Thus, a superior result is fairly probable when rerunning an experiment after reverting a hyperparameter. The reason why this problem does not scale for larger n lies in the nature of the central limit theorem: once the sample size n in an experiment is increased, the sampling distribution of sample

[1]Two-sample t-tests (with correction) are used to provide a commonly known significance testing method for a comprehensive comparison. However, some other comparison methods can be considered such Friedman-Nemenyi tests for comparing multiple algorithms (Gijsbers et al., 2024; Demšar, 2006; Herbold, 2020).

means from multiple experiments stays normally distributed. However, the standard deviation of this distribution $\sigma_{\bar{s}}$ is scaled down by the factor of $1/\sqrt{n}$, making the deviation much smaller.

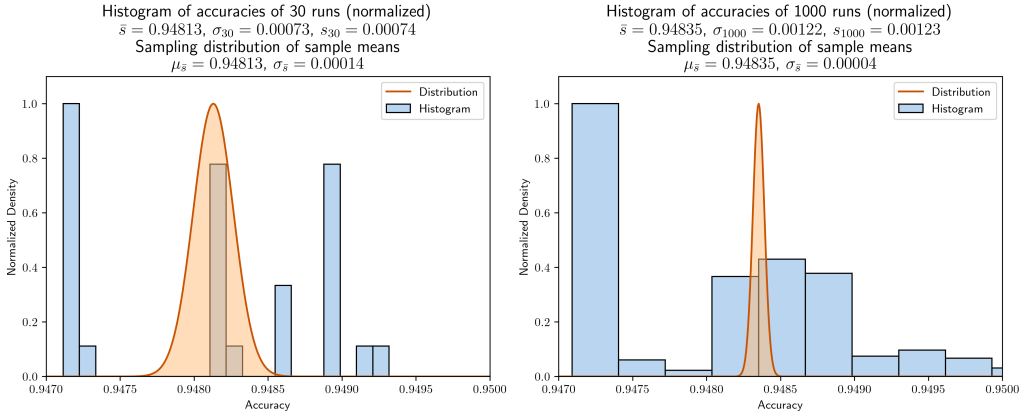

Figure 1: Example comparison of two experiments with 30 runs and 1000 runs each.

In the panels of Figure 1, the histograms of one experiment with 30 runs (left) using the regularized evolution as well as one experiment with 1000 runs (right) are shown as a tentative example of the impact of sample sizes. The Gaussian curves show the application of the central limit theorem of sampling another sample mean - thus performing another experiment - given the first experiment. The mean $\mu_{\bar{s}}$ of this normal distribution equals the sample mean $\bar{s}$ from the first experiment, while its standard deviation $\sigma_{\bar{s}}$ equals the population standard deviation s_n divided by $\sqrt{n}$. The probability distribution of expected sample means given an experiment with 1000 runs (right) is much sharper and defined. This can be interpreted with much more certainty to achieve the sample mean from the first experiment again, or only deviating little. The distribution given 30 runs (left) shows a higher variance. The resulting higher likelihood of deviating from the results signifies more uncertainty.

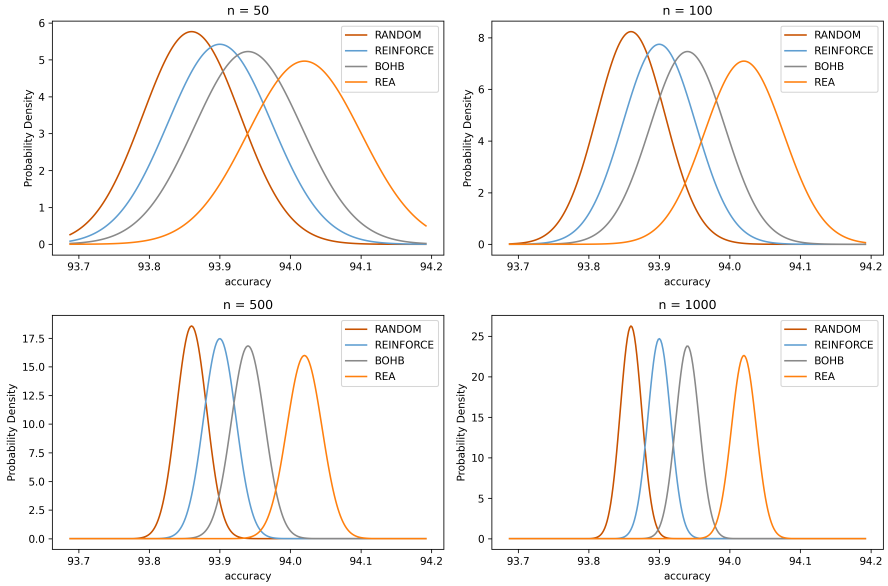

Figure 2: Comparison of experiments from Dong et al. (2021) with different n using mean and standard deviaton from 500 runs on the CIFAR-10 test dataset.

The application of the central limit theorem provides insights into the validity and statistical significance of results. Using data from a topology search with 500 runs (Dong et al., 2021) as the ground truth first experiment, we can simulate probabilities of rerunning experiments with different run sizes to visually compare how likely the results can be reproduced. Figure 2 shows the normal distribution of sampling means for experiments with 50, 100, 500, or 1000 runs, assuming the provided data from 500 runs as the ground truth. Each graph represents one experiment with n given runs. Although using standard deviations from one sample size to simulate others is technically incorrect, it illustrates the fuzzy borders of significance and provides a visual understanding of statistical significance. This also gives some intuition on when to apply the Student's t-test for algorithm performance. When comparing two algorithms, a two-sample t-test assesses their statistical significance. For example, real data (bottom left, Figure 2) show that REA and BOHB barely overlap, with a p-value of 0.02, indicating significance. In contrast, REINFORCE and BOHB significantly overlap, with a p-value of 0.22, indicating potential for error, even with 500 runs.

In NAS optimization, we know that multiple algorithms will reach a reasonable neural architecture. Therefore, research tries to uncover strategies to improve certain strategies, hypothesizing only incremental advantages to cutting-edge applications. In statistics, we would assume a low Cohen's d value, ranging between 0.1 and 0.2 indicating only a small effect. Using the concept of power analysis (Cohen, 1992), we would gain suggested sample sizes between 526 and 1570 (assuming a power of 0.8-0.9 for a significance level of 0.05). The formula for calculating the suggested sample size (n) is:

$$n = \frac{2 \cdot (z_{\alpha/2} + z_{\text{power}})^2}{\text{d}^2}$$

Where $z_{\alpha/2}$ is the z-score corresponding to the significance level (α) divided by 2 for a two-tailed test, and z_{power} is the z-score corresponding to the desired power level. Assuming that the truth is again somewhere in between, we propose a suggested sample size of 1000 to allow for existing uncertainty. While more runs are generally better, studies with fewer than 500 samples should be taken with a grain of salt. Additionally, presenting or visualizing the confidence interval, alongside the mean and variance, offers valuable insights. While the standard deviation interval indicates the likelihood of samples falling within the interval, the confidence interval essentially offers a level of certainty stating that the true mean value lies within the interval.

4 Addressing Time Constraints

A common constraint addressed by various NAS benchmarks was the selection of a maximal time budget. While there is typically no strict time constraint on executing maximum generations or cycles during NAS runs, algorithms are sometimes terminated before reaching convergence (Dong et al., 2021). While this premature termination may be common in real-world scenarios due to computational limitations, it may not be applicable when using tabular data. The risk of prematurely halting these algorithms is the potential oversight of insights into their behavior. It favors especially algorithms with fast initial progress - algorithms with slow initial progress that eventually converge to superior solutions are put at a disadvantage.

In Figure 3, the run data averaged over 1000 runs for two algorithms is plotted using a time budget of respectively $2e4$ and $2e5$ seconds. It demonstrates that the REINFORCE algorithm performs better after an initial slow warm-up period of 125,000 seconds, a timeframe that exceeds the time budget of 20,000 seconds utilized in Dong et al. (2021). These findings can be crucial for researchers when deciding which algorithm to select, making the choice of the REINFORCE algorithm more appealing when faced with fewer computational limitations. We therefore recommend a best practice to show the behavior of all algorithms until convergence, especially when evaluating these on precomputed, tabular benchmarks.

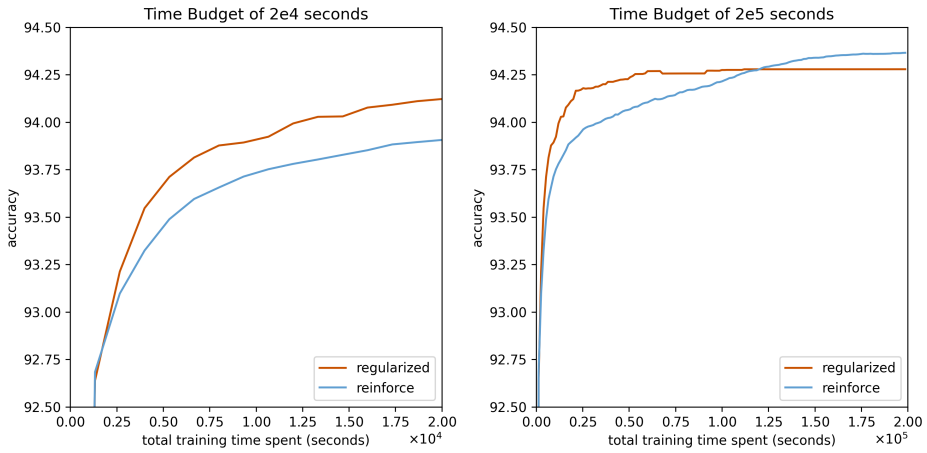

Figure 3: Comparison between regularized evolution and REINFORCE from Dong et al. (2021) using a time budget of 2e4 (left) and 2e5 (right) seconds. 1000 runs averaged.

5 Broader Impact Statement

Improving the reliability and robustness of algorithm comparisons in NAS research is crucial for advancing machine learning effectively. This requires a comprehensive approach that balances statistical significance testing with practical considerations. While statistical testing is important, it is essential to avoid overemphasizing p-values, as this can lead to misleading conclusions. In addition or instead, researchers should incorporate scientific judgment and broader contextual evidence into their analyses (Nuzzo, 2014).

In NAS evaluations, reaching convergence is important, but it might not always reflect real-world usability because of computational constraints. Rather than just extending the time budget, researchers can run more experiments within the original timeframe and choose the best-performing architecture. This method often leads to better results, especially when dealing with large search spaces.

Another concern is the reliance on mean performance as the primary metric for evaluation. In real-world applications where computational resources are limited, researchers may only be able to run a single or a few experiments. Relying solely on mean performance may not capture the full picture of algorithm robustness. Therefore, exploring alternative metrics to assess robustness becomes crucial for informed decision-making in algorithm selection.

However, regardless of the metric used, ensuring statistical reliability remains paramount. Adhering to proper statistical practices facilitates meaningful algorithm comparisons, providing genuine insights into performance. By addressing issues like p-hacking and cherry-picking, NAS research can produce more trustworthy and reproducible outcomes. This not only enhances individual studies but also elevates the overall standard of NAS research, fostering a more rigorous scientific approach.

Moreover, educating researchers on the importance of proper statistical measures contributes to a more robust scientific process. This, in turn, advances machine learning and promotes responsible AI innovation. Additionally, by promoting efficient use of precomputed data, NAS research can have positive environmental impacts, aligning with sustainability efforts.

In conclusion, prioritizing reliability and reproducibility in NAS research is essential for developing more efficient neural architectures. This approach not only drives progress in various AI applications but also contributes to broader advancements in the field. By adhering to rigorous methodologies and standards, researchers can ensure that their work contributes meaningfully to the advancement of machine learning and the broader goals of scientific inquiry.

References

Barbulescu, L., Watson, J.-P., and Whitley, L. D. (2000). Dynamic representations and escaping local optima: Improving genetic algorithms and local search. *AAAI/IAAI*, 2000:879–884.

Cohen, J. (1992). Statistical power analysis. *Current directions in psychological science*, 1(3):98–101.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

Dong, X., Liu, L., Musial, K., and Gabrys, B. (2021). NATS-Bench: Benchmarking nas algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Fan, Y., Alon, D., Shen, J., Peng, D., Kumar, K., Long, Y., Wang, X., Iliopoulos, F., Juan, D.-C., and Vee, E. (2023). Layernas: Neural architecture search in polynomial complexity. *arXiv preprint arXiv:2304.11517*.

Gijsbers, P., Bueno, M. L., Coors, S., LeDell, E., Poirier, S., Thomas, J., Bischl, B., and Vanschoren, J. (2024). Amlb: an automl benchmark. *Journal of Machine Learning Research*, 25(101):1–65.

Groh, R. and Kist, A. M. (2023). End-to-end evolutionary neural architecture search for microcontroller units. In *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–7. IEEE.

Head, M. L., Holman, L., Lanfear, R., Kahn, A. T., and Jennions, M. D. (2015). The extent and consequences of p-hacking in science. *PLOS Biology*, 13(3):1–15.

Herbold, S. (2020). Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48):2173.

Klein, A. and Hutter, F. (2019). Tabular benchmarks for joint architecture and hyperparameter optimization.

Krzywinski, M. and Altman, N. (2013). Points of significance: Importance of being uncertain. *Nature methods*, 10:809–10.

Maaranen, H., Miettinen, K., and Penttinen, A. (2007). On initial populations of a genetic algorithm for continuous optimization problems. *Journal of Global Optimization*, 37:405–436.

Nuzzo, R. (2014). Scientific method: Statistical errors. *Nature*, 506(7487):150–152.

Welch, B. L. (1947). The generalization of student's problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35.

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

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [N/A] No claims were made.

   (b) Did you describe the limitations of your work? [Yes] The limitations are presented in the impact section.

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] The impacts are discussed in the impacts section.

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` [Yes]

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [Yes] The NATS-Bench experiments were performed with the same benchmark, as well as NAS-Bench-101 experiments.

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [N/A] No evaluation details were needed.

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [No] Sufficient runs were performed to account for randomness. The tests with low run counts were intentionally designed to show the impact of randomness.

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [Yes] The standard deviation of the tests was reported.

   (e) Did you report the statistical significance of your results? [Yes] This was the main idea of the paper.

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes] NAS-Bench-101, as well as NATS-Bench, were used.

   (g) Did you compare performance over time and describe how you selected the maximum duration? [Yes] The comparison used the maximum time until convergence of both presented algorithms.

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] The experiments where not computationally expensive.

   (i) Did you run ablation studies to assess the impact of different components of your approach? [N/A] No approach was presented.

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] Provided under https://github.com/sdnfr/time-bench.

(b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes] The minimal example is provided as provided data under https://github.com/sdnfr/time-bench.

(c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes] Code was supplied under https://github.com/sdnfr/time-bench.

(d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes] Raw results are included and saved in provided code directory under https://github.com/sdnfr/time-bench.

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes] All material and code is openly available under https://github.com/sdnfr/time-bench.

4. If you used existing assets (e.g., code, data, models)…

(a) Did you cite the creators of used assets? [Yes] The authors of the presented results were cited.

(b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [Yes] Data was used in accordance with the provided licenses.

(c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Data does not contain such information.

5. If you created/released new assets (e.g., code, data, models)…

(a) Did you mention the license of the new assets (e.g., as part of your code submission)? [N/A] No assets submitted.

(b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [N/A] No assets submitted.

6. If you used crowdsourcing or conducted research with human subjects…

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No human subjects.

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

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] No human subjects.

7. If you included theoretical results…

(a) Did you state the full set of assumptions of all theoretical results? [Yes] The underlying equations and foundations were presented, otherwise cited or considered text book knowledge.

(b) Did you include complete proofs of all theoretical results? [N/A] No proofs were derived.