Accel-NASBench: A Surrogate Benchmark for Accelerator-Aware NAS

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

The exorbitant compute costs of Neural Architecture Search (NAS) research have highlighted the need for benchmarks and reproducibility. Emerging benchmarks have offered a zero-cost evaluation of NAS techniques, drastically simplifying the comparison of NAS methods. However, most existing NAS benchmarks are constructed using synthetic proxy datasets or model proxies that make simplified assumptions about the characteristics of these datasets and models. When using these benchmarks, NAS evaluation can only be performed within the scope of these assumptions, with no guarantee of transferability of performance to real-world scenarios. In addition, the rise of high-performance hardware accelerators has led to an emerging trend in accelerator-aware architecture search. However, the existing hardware-aware benchmarks are limited to dated search spaces and proxy datasets that do not allow realistic evaluation. To address these challenges, we propose a surrogate NAS benchmark for accelerator-aware NAS on the ImageNet2012 dataset. Our approach uses controlled *training proxies* that substantially reduce the cost of benchmark construction while ensuring that architecture rankings relative to true evaluation are largely maintained. We showcase the generality of our techniques in reducing benchmark construction costs on different datasets and search spaces. We also offer performance benchmarks for high-performance accelerators such as cloud TPUs, GPUs, and FPGAs.

1 Introduction

The proliferation of research in Neural Architecture Search (NAS), combined with the prohibitive costs of NAS evaluation have highlighted the need for benchmarks and reproducibility (Li and Talwalkar, 2020). Zero-cost NAS benchmarks using surrogate predictors have paved the way for cheap NAS evaluation while ensuring complete coverage of large search spaces at a minimal cost of benchmark construction (Siems et al., 2020; Yan et al., 2021). These benchmarks are constructed by evaluating a small but representative portion of the search space exhaustively, and using these evaluations to train black-box surrogates that can be utilized to obtain robust estimates of performance of unseen regions of the search space. However, currently available zero-cost NAS benchmarks, whether tabular or surrogate, only apply to small datasets such as CIFAR10/100 (Krishna et al., 2009), or cheap variants of large datasets such as ImageNet16-120 (Chrabaszcz et al., 2017).

Numerous works have pointed out the parameter-inefficiency and low quality of results obtained using *proxy* datasets for search (Cai et al., 2018; Chen et al., 2021). Specifically, the process of searching on proxy datasets completely disregards the target dataset during the optimization process.

Submitted to the 37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks. Do not distribute.

Since the proxy and target datasets often have a massive disparity in complexity, the searched models exhibit a low parameter-efficiency. On the other hand, a few high-quality works that perform direct search on large datasets have shown that a direct search yields models that exhibit higher parameter-efficiency (Tan and Le, 2019; Cai et al., 2018). However, the high search cost of these methods, often in tens of thousands of GPU/TPU-hours (Wan et al., 2020), prohibits reproducibility.

On another front, the proliferation of high-performance hardware accelerators, combined with a drive for fast model serving, has given rise to accelerator-aware neural network design, intending to optimize multiple performance metrics, including accuracy and on-device throughput/latency. However, numerous research works have pointed out the inability of conventional model-specific complexity metrics, such as FLOPs and model size, to serve as accurate proxies for the model's on-device performance (Gupta and Akin, 2020; Li et al., 2021; Bakhtiarifard et al., 2022). This is because a model's on-device performance is influenced by a range of factors beyond just FLOPs and model size, such as memory-bandwidth requirements, data reuse, and the frequency of off-chip memory accesses. These factors are device-specific, meaning that the optimal choice of model varies depending on the characteristics of the hardware being used.

This work introduces a surrogate benchmark allowing direct evaluation of searched models on the ImageNet2012 (Deng et al., 2009) dataset on the MnasNet search space (Tan et al., 2019). We also offer surrogates for hardware accelerator performance metrics for GPUs, TPUs, and FPGAs. Accel-NASBench is geared towards discrete NAS optimizers such as RL-based (Zoph and Le, 2017) and Evolutionary Search (Real et al., 2019), preferably in bi-objective optimization settings (e.g., Tan and Le (2019); Howard et al. (2019); Gupta and Akin (2020); Tan et al. (2019)). This work makes the following contributions:

- We offer a surrogate NAS benchmark for the ImageNet2012 dataset on the MnasNet search space. Accel-NASBench is the first NAS benchmark for a popular large-scale dataset. The benchmark is constructed by applying a host of aggressive training proxies that substantially reduce the benchmark construction costs.
- We offer surrogates for accelerator performance metrics for Cloud TPUs, GPUs, and FPGAs. Specifically, we offer inference throughput surrogates for Cloud TPUv2 and TPUv3, A100 and RTX-3090 GPUs, and Xilinx Zynq Ultrascale+ ZCU102 and Versal AI Core VCK190 FPGAs. Latency surrogates are also offered for the FPGA platforms.
- Through extensive ablations, we show that despite the use of aggressive training proxies for benchmark construction, the surrogates' predictive architecture rank distribution strongly resembles the true rank distribution of architectures in the search space. We explore the applicability of training proxies to two popular datasets and search spaces utilized in NAS. We also evaluate the performance of NAS optimizers on Accel-NASBench in both uni- and bi-objective search settings.

We open-source the surrogate benchmark, collected datasets, and searched models, and encourage further research and reproducibility[†].

2 Motivation

Constructing NAS benchmarks is a computationally demanding process that requires thousands of model evaluations. We highlight a few limitations of current NAS benchmarks, discuss the environmental impacts of benchmark construction, and future outlook of NAS research.

Datasets and Proxies. Most NAS works utilize small proxy datasets for search owing to the substantially lower search cost. A small number of research works that perform direct search on ImageNet2012 have yielded models that beat state-of-the-art in parameter inefficiency (e.g., Tan and Le (2019); Howard et al. (2019); Cai et al. (2018)). This is because of the huge disparity in

[†]https://anonymous.4open.science/r/Accel-NASBench

complexity of proxy datasets such as CIFAR10, and the target datasets such as ImageNet. Recent NAS benchmarks have employed ImageNet16-120 (Chrabaszcz et al., 2017) as a proxy for ImageNet2012, however, given the huge disparity in input size (16x16 vs 469x387 average resolution), number of classes (120 vs 1000), and the number of samples in the dataset (0.15M vs 12.4M images), ImageNet16-120 does not serve as a representative proxy for ImageNet2012.

In addition, most NAS benchmarks are constructed using not just *dataset proxies* but also *model proxies*, whereby shallow (reduced depth), narrow (reduced width) variants of models in the search spaces are utilized in benchmark construction to lower the compute costs. For example, CIFAR-10 evaluation results of differentiable architecture search works (e.g., Liu et al. (2018); Wang et al. (2021); Chen et al. (2019)) are reported on a network of 20 cells with 36 initial channels while NASBench-301 utilizes 8 cells and 32 initial channels. Existing works have shown that the disparity between search and evaluation networks, especially depth gaps between the networks, leads to sub-optimal results (Chen et al., 2019; Xie et al., 2018).

Environmental Footprint. While high-quality NAS benchmarks exist for proxy datasets, constructing benchmarks for large-scale datasets such as ImageNet without using model proxies has remained elusive owing to the high cost of model evaluation. While NAS benchmarks, such as NASBench-101 (Ying et al., 2019), are an indispensable tool towards democratizing NAS research using zero-cost evaluations, the tremendous amount of CO_2 emissions from 100 TPU-years of compute for benchmark construction, the limited coverage of the search space (423k models), and the use of CIFAR-10 dataset calls into question the environmental impacts and sustainability of NAS benchmarks. Hence, there is a strong need to fundamentally rethink the approach to benchmark construction from the perspective of sustainability.

Search Spaces. There is a growing interest in exploring search spaces beyond the heavily investigated DARTS search space to better understand the performance of NAS methods on diverse search spaces and in real-world applications (Tu et al., 2022). Towards this end, existing benchmarks call for construction of benchmarks for diverse and rich search spaces (Siems et al., 2020).

The MnasNet search space is a hierarchical block-based search space that is known to yield state-of-the-art parameter-efficient models, in both platform-agnostic (Tan and Le, 2019) and platform-aware (Gupta and Akin, 2020; Tan et al., 2019) settings. We utilize a modified version of the search-space that caters to the goal of high-quality model search across a variety of hardware platforms. For a thorough list of changes relative to the MnasNet search space, please see *Appendix. B*. The search space holds roughly 10^{11} unique architectures.

On-Accelerator Performance. The rise of high-performance AI accelerators has substantiated calls for accelerator-aware model search, aiming to optimize both model-specific metrics such as accuracy, and device-specific metrics such as throughput. Numerous recent works have shown that model-specific complexity metrics such as FLOPs/model size serve as weak proxies for on-device performance. This is due to the wide variety of potential performance bottlenecks that could result from device-agnostic search, owing to architectural differences between hardware platforms.

Outlook of NAS. With the ongoing trend of increasing model complexity in ML, hardware accelerators will become unavoidable and hardware-aware benchmarks would be indispensable in furthering the domain of NAS towards more practical solutions. In addition, as models continue to scale up, constructing realistic NAS benchmarks will continue to be prohibitively expensive, especially on large models and datasets. Within this prospect, our work in using only *training proxies* to alleviate this cost is worth exploring.

We compare Accel-NASBench with existing HW-aware NAS benchmarks in *Appendix. A*.

3 Dataset collection and surrogate fitting

Emergent NAS benchmarks have offered a blueprint for cheap benchmark construction using surrogates (e.g., NASBench-301 (Siems et al., 2020)). Given both FBNet and MnasNet search spaces are hierarchical block-based search spaces, the findings of NASBench-301 on FBNet largely apply to the MnasNet search space.

Architecture Sampling. NASBench-301 utilized architectures sampled using various NAS optimizers to obtain good coverage of strong and weak regions of the search space. However, unbiased surrogates (i.e., those fitted using only randomly sampled data) are found to already serve as strong predictors. Furthermore, the notion of 'strong' and 'weak' regions of a search space is hard to define when evaluation is based on multiple performance metrics and various hardware platforms and not just based on accuracy. Hence, we construct our surrogates using unbiased, randomly sampled data from the search space.

Collecting the accuracy data for the selected architectures on ImageNet2012 is computationally challenging, as we do not use dataset and model proxies. However, given the smaller search space size relative to FBNet, fewer evaluations are required to obtain strong surrogates. We evaluated the surrogate performance at various stages during the dataset collection process to find the stopping point for dataset collection. At roughly only 5.2k model evaluations, we found that the surrogates constructed using such few evaluations already exhibit strong performance. Our analysis in *Appendix. E* shows that surrogates trained with only 5.2k evaluations outperform tabular benchmarks. For these 5.2k architectures, we also collected device-specific performance metrics such as throughput on all six accelerators and latency on the FPGAs.

3.1 Architecture-accuracy pairs

With the environmental impact of constructing NAS benchmarks and their practical utility in mind, we aim to reduce the evaluation cost of architectures for dataset collection while ensuring that the constructed benchmark retains architecture rankings relative to a true benchmark. Towards this end, we utilize a host of *training proxies* to alleviate the cost of model evaluation. To motivate the use of training proxies, we first show the difference in time and dollar cost of training a single model, EfficientNet-B0, with and without proxies. Table. 1 shows the approximated training cost without proxies, both in time and dollar value, is an order of magnitude higher than proxified training, however, proxification yields a significant degradation in the validation accuracy of the model owing to *underfitting* due to constrained training budget. In the subsequent sections, we will show that despite this degradation of accuracy, the rankings of architectures in the search space relative to those obtained using more realistic training schemes are largely maintained. Please note that the proxies utilized in our work only pertain to the training schemes, and not to the models or datasets. NASBench-301 utilized both training and model proxies (e.g., narrow, shallow models with fewer channels and layers, combined with shortened training budgets). Hence, when using NASBench-301, search methods can only be evaluated within the scope of these proxies, without guarantee of transferability of performance when models are scaled up. Our work differs from NASBench-301, in that, we offer heuristics that show with sufficient confidence that evaluation results under training proxies are similar to those under true evaluation.

Table 1: Impact of training proxies.

	No-Proxy[†]	Proxified[‡]
Top-1 Acc (%)	77.30	67.06
Train Time (TPU/GPU-hours)	53	2.3
TPU/GPU $ Cost	466	20

Trained on [†] a TPUv3-8 and [‡] 4x RTX 3090 GPUs

Next, we briefly describe the prominent training proxies that we utilized to drastically decrease the training cost. For a full list of proxies, please see *Appendix. C*.

Constrained Training Budget. Existing works in both NAS and HPO have shown that early stage training performance of a model is often indicative of its performance during the later stages. Methods such as Hyperband (Li et al., 2017) utilize this observation as the basis for algorithm configuration selection. Given existing benchmarks also utilize short training budgets for model training, this is a natural choice to reduce model evaluation cost.

Progressive Resizing. Progressive resizing has been shown to accelerate the training speed of convolutional models while maintaining a high level of accuracy (Karras et al., 2017). By initially training the model on small image sizes and progressively increasing the input size, the model is able to rapidly learn low-level features, which leads to faster convergence during training.

Fast Dataloading and Input Compression. We utilize FFCV (Leclerc et al., 2022), a fast dataloader that offers input prefetching, caching, fast input transformations, and pipelining. We pair FFCV with JPEG compressed input, with the dataset compressed to 90% JPEG to allow caching the entire dataset in memory for faster loading speeds.

The accuracy dataset is collected on a cluster of 6 nodes, each with $4\times$ RTX3090 GPUs, and incurs a cost of 17k GPU-hours. The collected accuracy dataset is named `ANB-Acc` for future reference.

3.2 Accelerator-specific performance metrics

We measured architecture throughputs for the 5.2k architectures on six accelerator devices: Google Cloud TPUv2 and TPUv3, NVIDIA RTX 3090 and A100 GPUs, and Xilinx Ultrascale+ ZCU102 and Versal AI Core VCK190 FPGAs. We also performed latency measurements on the FPGA platforms. These on-device performance datasets would allow the benchmark to be utilized to evaluate search algorithms geared towards bi-objective optimization NAS.

On **cloud TPUs**, the main challenge in measuring throughput is the TPU warmup, which involves XLA graph compilations and caching. We discard the TPU warmup phase measurements before taking throughput measurements four times and using the average as the measured values. On **GPU** platforms, we discard warmup throughput measurements and use the mean of 2 inference runs as the throughput value.

On **FPGA** platforms, inference throughput, and latency measurements are performed using Xilinx Deep-Learning Processing Unit (DPU) blocks. We performed 8-bit post-training quantization of weights for the 5.2k architectures, followed by cross-compilation for the target DPUs and platforms. FPGAs have negligible variability in performance; hence there is no warm up period to exclude from the measurements and little measurement noise to smooth by averaging.

Please see *Appendix. D* for details on collection pipelines. The collected accelerator performance datasets on the 5.2k architectures are termed `ANB-{device}-{metric}`, where throughput (`Thr`) metric is supported by all devices while latency (`Lat`) is supported only by FPGAs.

3.3 Surrogate fitting

Following NASBench-301 (Siems et al., 2020), we compare the predictive performance of a variety of candidate surrogates, including XGBoost, LGBoost, Random Forests, ϵ-SVR, and ν-SVR.

We split the collected datasets into train/val/test splits of ratio 0.8/0.1/0.1 and utilize the train/val splits for surrogate hyperparameter tuning. We represent the hyperparameters in ConfigSpace (Lindauer et al., 2019) and utilize SMAC3 (Lindauer et al., 2022) for finding the configurations for the surrogate candidates (tuned hyperparameters in *Appendix. F*). We utilize vectorized encoding of ConfigSpace configurations as input to the surrogates. We fit the surrogates on the datasets using the train split and evaluate on the test split.

Table 2: Surrogate test performance on `ANB-Acc`.

Model	R^2	KT τ	MAE
XGB	**0.984**	**0.922**	**3.06e-3**
LGB	0.984	0.922	3.08e-3
RF	0.869	0.782	8.88e-3
ϵ-SVR	0.943	0.886	5.32e-3
ν-SVR	0.942	0.881	5.45e-3

We evaluate the surrogates' fit quality using the coefficient of determination (R^2), Kendall's Tau rank correlation τ, and mean absolute error (MAE). As shown in Table. 2, gradient-boosting techniques outperform all other surrogates in all three evaluation metrics. We also notice a similar trend in the on-device performance surrogates. Table. 3 shows the performance of XGBoost on the test datasets of `ANB-{device}-{metric}`. Please note that MAE is measured in images/sec for `Thr` datasets and milliseconds (ms) for `Lat` datasets. Given the strongly predictable and deterministic

Table 3: XGB test performance on `ANB-{device}-{metric}`.

Dataset	R^2	KT τ	MAE
`ANB-ZCU-Thr`	0.990	0.955	13.2
`ANB-ZCU-Lat`	1.000	0.987	5.2e-2
`ANB-VCK-Thr`	0.991	0.949	69.5
`ANB-VCK-Lat`	0.999	0.980	4.0e-2
`ANB-TPUv3-Thr`	0.975	0.905	29.1
`ANB-TPUv2-Thr`	0.994	0.962	14.4
`ANB-A100-Thr`	0.995	0.975	159.7
`ANB-RTX-Thr`	0.996	0.968	116.1

nature of performance on most of the evaluated accelerators, most of the performance surrogates perform better than the accuracy surrogates.

For noise modeling, we utilize ensembles of 6 base learners using the XGBoost and LGBoost surrogates using 6-fold cross-validation of train/val datasets. The quality of fit of the XGBoost surrogate ensembles and the relevant surrogate performance metrics are shown in Appendix I.

4 Ablation studies

4.1 Underfitting performance as indicator of true performance

The use of aggressive training proxies leads to underfitting. Since we utilized the underfitting performance as an indicator of true performance, in this section, we justify the use of aggressive proxies by comparing architecture rankings obtained under the proxified training scheme to those obtained by training using a few proxies. First, we introduce a few-proxy training scheme that achieves a balance between the aggressiveness of proxies and the cost of model training (See *Appendix. C* for configurations of different training recipes). Few-proxy scheme is utilized as a replacement for the no-proxy scheme, given the immense cost of model evaluation without any training proxies. This is under the assumption that architecture rankings under few-proxy and no-proxy training schemes are strongly correlated. Existing benchmarks are also firmly based on this assumption and utilize proxies in model evaluation. For example, NASBench-301 utilized 100 epochs for evaluation on CIFAR-10, while the standard DARTS-like evaluation uses 400-450 epochs.

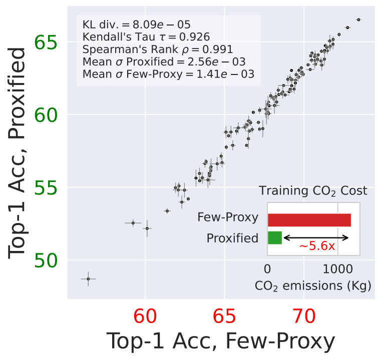

Figure 1: Underfitting vs true accuracy

Comparing architecture rank correlations. We trained 120 randomly sampled architectures from the search space using three seeds using both proxified and few-proxy training schemes. For each architecture, we take the average accuracy of the three runs under the few-proxy training scheme as its true accuracy. The average accuracy under the proxified training scheme is the underfit accuracy of the model. We plot the average and standard deviation of the true and underfit architecture accuracies in Fig. 1. The key findings in Fig. 1 are listed as follows

- The high Kendall's Tau correlation in evaluation between the two training schemes of $\tau = 0.926$, combined with a low Kullback–Leibler (KL) divergence highlights the strong ability of underfitting distribution to model the true distribution of architecture rankings.
- Proxified training scheme results in $5.6\times$ lower CO_2 cost than the few-proxy training scheme, highlighting the reduced carbon footprint for benchmark construction.
- Underfit models exhibit a high variability in evaluation noise as shown by the higher mean σ. This points to a weak predictive performance of a tabular benchmark constructed using a proxified scheme relative to a true benchmark constructed using a few-proxy scheme.

These findings suggest high-quality benchmarks for large-scale datasets can be constructed in an efficient manner using controlled proxies. The few-proxy and proxified training schemes incur 4488 and 800 GPU-hours for training the models, respectively, highlighting the $5.6\times$ cost and emissions savings for benchmark construction. In the next subsection, we showcase the generalizability of training proxies in reducing the benchmark construction costs across search spaces and datasets.

4.2 Generalizability of training proxies

We study the generalizability of training proxies in reducing the benchmark construction costs on CIFAR100 and ImageNet2012 datasets on the FBNet search space. This serves to show that the findings of Sec. 4.1 are not tied to ImageNet2012 dataset combined with the MnasNet space.

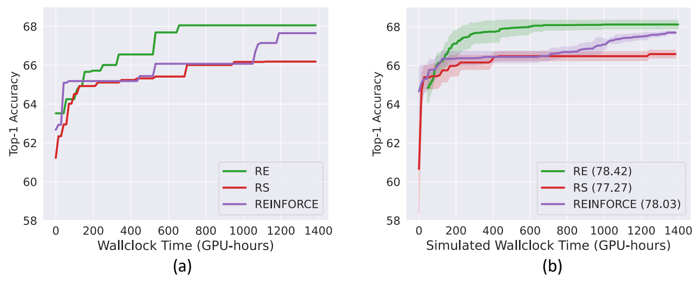

Figure 3: Comparison of trajectory of uni-objective search between (a) real and (b) simulated runs. Each simulated run is an average of 5 optimizer runs with different random seeds. Also shown in the legend of (b) are the true top-1 ImageNet accuracy of best models found from each of the three search methods.

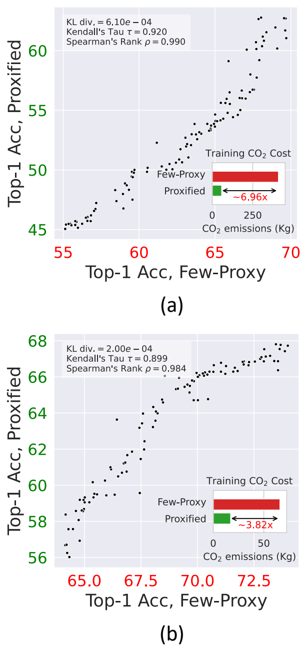

Figure 2: Generalization to FBNet on (a) ImageNet, (b) CIFAR100.

We trained 100 randomly selected models in the FBNet search space on both CIFAR100 and ImageNet2012 datasets using the proxified and few-proxy training schemes. Similar to Section. 4.1, we study the architecture rank correlations of these randomly selected architectures between the two training schemes. Please note that for this study, we did not tune the training schemes for the new dataset and search space combinations relative to those in Section. 4.1. Figure. 2 (a) and (b) plot the top-1 accuracies of the selected FBNet models under proxified vs. few-proxy training schemes on ImageNet2012 and CIFAR100 datasets, respectively. On ImageNet, we obtain a Kendall's Tau of $\tau = 0.920$, similar to that obtained on the MNasNet search space in Sec. 1, combined with training cost reduction of $6.96\times$. This demonstrates the generalizability of training proxies across search spaces. On CIFAR100, we obtain a relatively lower Kendall's Tau of $\tau = 0.899$. This is a consequence of the significantly lower size of CIFAR100 combined with fewer classes relative to ImageNet2012 and the use of a training scheme better suited for ImageNet. Nevertheless, the high Kendall's Tau achieved for both ImageNet and CIFAR100 datasets, combined with the $6.96\times$ and $3.82\times$ lower training costs of the proxified training scheme, offer fair heuristic justification for our claim that training proxies offer a sustainable way of constructing NAS benchmarks. On CIFAR100, rank correlation can be improved by tuning the proxified training scheme to account for the disparity between dataset complexities. The few-proxy and proxified training schemes incur 1536 and 220 GPU-hours on ImageNet, and 248 and 64 GPU-hours on CIFAR100, respectively, for training the models in this study.

To further show the integrity of benchmarks constructed using training proxies relative to true evaluation, in *Appendix E*, we show that the predictive distributions of surrogate and tabular benchmarks constructed using single seed using proxified scheme model the true benchmarks accurately. *Please see Appendix. E for additional ablation studies.*

5 Evaluating Accel-NASBench

5.1 Evaluating the accuracy surrogate

We compare the trajectory of search using only the accuracy surrogate against true optimizer runs on three popular discrete NAS optimizers: Regularized Evolution (RE) (Real et al., 2019), Random Search (RS) (Li and Talwalkar, 2020), and REINFORCE (Zoph and Le, 2017). The trajectories using simulated runs are averaged over five runs; however, the true runs are only performed once owing to the high evaluation cost of each run (as also performed by Siems et al. (2020)).

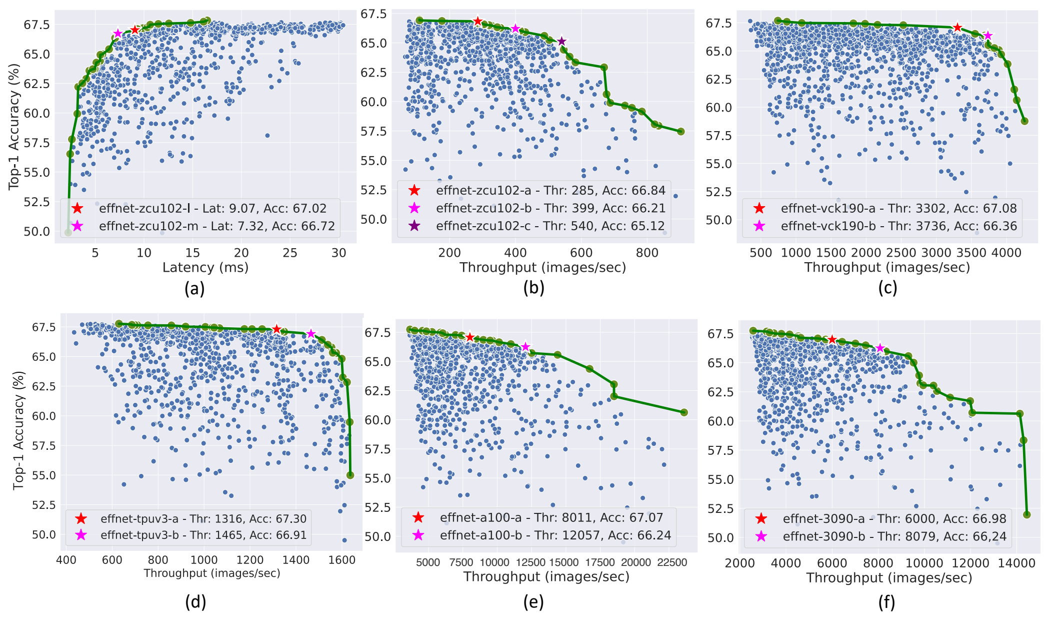

Figure 4: Search using RL-based bi-objective optimization. Fig (a) shows the pareto-optimal front using simulated search on accuracy-latency objectives. Fig (b)-(f) show the results of accuracy-throughput search on (b) ZCU102 and (c) VCK190 FPGAs, (d) TPUv3, (e) A100 and (f) RTX 3090 GPUs. Also shown in red, magenta, and purple star markers are pareto-optimal solutions hand-picked for evaluation. Legends show their performances predicted using surrogates.

Results. Fig. 3 shows the trajectory of (a) real and (b) simulated search. The accuracy surrogate is able to mimic the behaviour of true search, with RS predictably underperforming compared to REINFORCE and RE owing to the high variability of model performance in the search space. This is in contrast to the dense and highly predictable DARTS search space in which RS is able to achieve near state-of-the-art performance (Li and Talwalkar, 2020). On MnasNet space, RS stagnates fairly early while RE and REINFORCE achieve significantly better results. This is in line with the findings of existing works on other search spaces (such as Siems et al. (2020)).

5.2 Evaluating Accel-NASBench for bi-objective search

We evaluate the performance of Accel-NASBench for bi-objective RL-based search. Towards this end, instead of comparing the search trajectory between surrogates-based search and true search, we show that the search results using surrogates yield models that are comparable in accuracy and performance to those found by existing works that perform true search (Tan and Le, 2019; Gupta and Akin, 2020). We do this to avoid the exorbitant compute cost and complex instrumentation of true bi-objective discrete RL-based search, which would require both accuracy and on-device throughput measurements in a single, continuous pipeline. Having not utilized any *model* or *dataset* proxies in the construction of Accel-NASBench, the search results can be compared directly against known high-quality models (such as Tan and Le (2019); Gupta and Akin (2020)). We perform searches using the throughput surrogates of five devices and the latency surrogate of ZCU102 FPGA. Fig. 4 (a) shows the pareto-optimal solutions resulting from accuracy-latency bi-objective search using the ZCU102 FPGA latency surrogate. Fig. 4 (b)-(f) shows the accuracy-throughput search results targeting the five accelerators. *Please see Appendix. G for details of search.*

Evaluation. Since the accuracy surrogate predicts the accuracy under the proxified training scheme rather than the true accuracy, we evaluate a few hand-picked pareto-optimal solutions from the searches by training using the no-proxy training scheme, and by performing on-device throughput/latency measurements. The evaluation true accuracy and throughput/latency results are plotted in Fig. 5 for the 5 hardware platforms. We compare the obtained accuracy-throughput against existing state-of-the-art searched/handcrafted models. Results show performance improvements

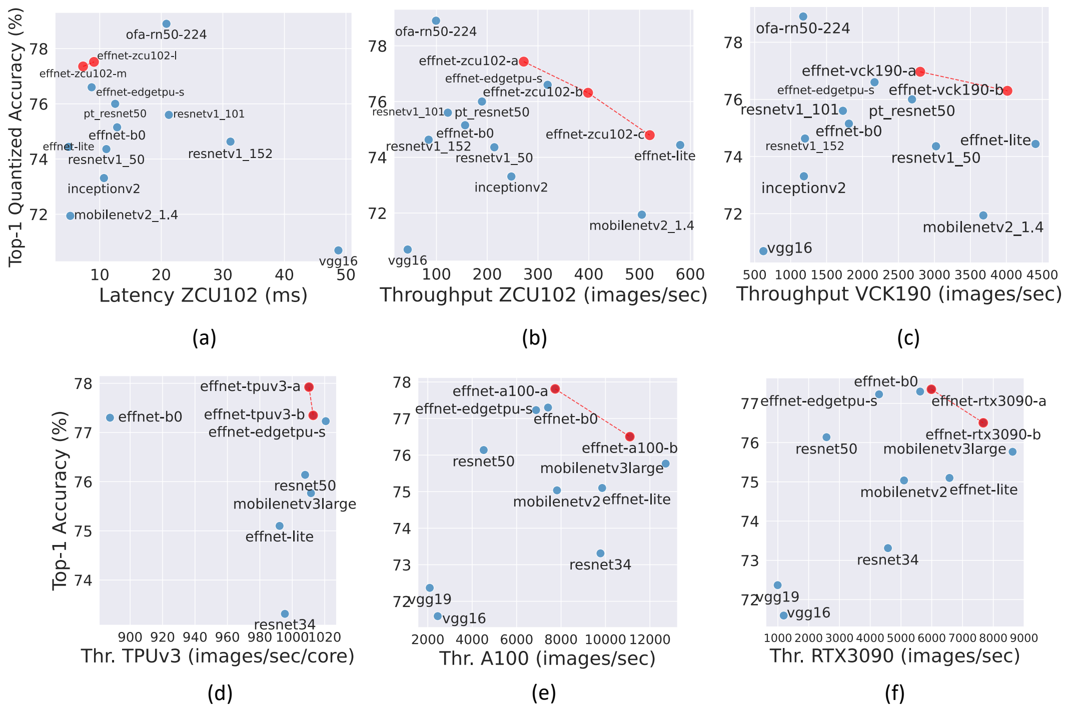

Figure 5: Evaluation results comparison against high-quality models on (a) accuracy-latency tradeoff on ZCU102, and accuracy-throughput tradeoff on (b) ZCU102, (c) VCK190 FPGAs, (d) TPUv3, (e) A100 and (f) RTX-3090 GPUs. In red markers are the evaluation results of our search using accuracy-throughput/latency bi-objective search using REINFORCE. The zero-cost search using Accel-NASBench yields models that compare to known high-quality solutions in the MnasNet search space (e.g., efficientnet-b0 and efficientnet-edgetpu-s).

compared to accuracy-FLOPs optimization works such as EfficientNet-B0 (Tan and Le, 2019) and MobileNetsV3 (Howard et al., 2019). For example, our `effnet-vck190-a` achieves 1.8% better accuracy and 55.0% better throughput than `effnet-b0` (Tan and Le, 2019), and 0.37% better accuracy and 29.4% better throughput than `effnet-edgetpu-s` (Gupta and Akin, 2020), on the VCK190 FPGA platform. Given the fact that surrogate-based zero-cost search is able to find high-quality models that offer comparable accuracy-throughput trade offs against existing high-quality true search results, we conjecture that the surrogates' predictive distributions mimic the true accuracy/throughput distributions fairly accurately since the NAS optimizer is able to effectively explore the well-performing regions of the search space.

6 Conclusion and future outlook

We proposed a sustainable approach to NAS benchmark construction for large-scale datasets using aggressive training proxies. The benchmark allows zero-cost evaluation of NAS works in real-world settings without using any dataset and model proxies. We justified the use of training proxies using extensive experimentation to show the integrity of the architecture rankings under these proxies relative to a true benchmark. We also offer surrogates for 6 hardware accelerators to allow evaluation of bi-objective NAS works, and showed that a bi-objective NAS optimizer can find models that compare to the state-of-the-art known models in the search space, at a zero cost using our surrogates. With growing model sizes, we expect our benchmark construction approach to be further explored by the NAS community in creating benchmarks for more exciting and practical large-scale datasets. In addition, with the growing prevalence of AI accelerators, we expect our benchmark to be an indispensable tool for evaluation of multi-objective NAS methods.

References

Bakhtiarifard, P., Igel, C., and Selvan, R. (2022). Energy consumption-aware tabular benchmarks for neural architecture search. *arXiv preprint arXiv:2210.06015*.

Cai, H., Zhu, L., and Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*.

Chen, W., Gong, X., and Wang, Z. (2021). Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*.

Chen, X., Xie, L., Wu, J., and Tian, Q. (2019). Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*.

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

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*. Ieee.

Dong, X. and Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*.

Dudziak, L., Chau, T., Abdelfattah, M., Lee, R., Kim, H., and Lane, N. (2020). Brp-nas: Prediction-based nas using gcns. *Advances in Neural Information Processing Systems*, 33:10480–10490.

Duplyakin, D., Ricci, R., Maricq, A., Wong, G., Duerig, J., Eide, E., Stoller, L., Hibler, M., Johnson, D., Webb, K., et al. (2019). The design and operation of cloudlab. In *USENIX Annual Technical Conference*, pages 1–14.

Gupta, S. and Akin, B. (2020). Accelerator-aware neural network design using automl. *arXiv preprint arXiv:2003.02838*.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *ICCV*.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Krishna, R., Hinton, G., et al. (2009). Cifar-10 dataset. `https://www.cs.toronto.edu/~kriz/cifar.html`.

Leclerc, G., Ilyas, A., Engstrom, L., Park, S. M., Salman, H., and Madry, A. (2022). FFCV: Accelerating training by removing data bottlenecks. `https://github.com/libffcv/ffcv/`. commit 19f4039b992561eb93542ca43dd841218d206b56.

Li, C., Yu, Z., Fu, Y., Zhang, Y., Zhao, Y., You, H., Yu, Q., Wang, Y., and Lin, Y. (2021). Hw-nas-bench: Hardware-aware neural architecture search benchmark. *arXiv preprint arXiv:2103.10584*.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816.

Li, L. and Talwalkar, A. (2020). Random search and reproducibility for neural architecture search. PMLR.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. (2022). Smac3: A versatile bayesian optimization package for hyperparameter optimization. *J. Mach. Learn. Res.*, 23(54):1–9.

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

Liu, H., Simonyan, K., and Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Siems, J., Zimmer, L., Zela, A., Lukasik, J., Keuper, M., and Hutter, F. (2020). Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*.

Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*. PMLR.

Tu, R., Roberts, N., Khodak, M., Shen, J., Sala, F., and Talwalkar, A. (2022). Nas-bench-360: Benchmarking neural architecture search on diverse tasks. *Advances in Neural Information Processing Systems*, 35:12380–12394.

Wan, A., Dai, X., Zhang, P., He, Z., Tian, Y., Xie, S., Wu, B., Yu, M., Xu, T., Chen, K., et al. (2020). Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12965–12974.

Wang, R., Cheng, M., Chen, X., Tang, X., and Hsieh, C.-J. (2021). Rethinking architecture selection in differentiable nas. *arXiv preprint arXiv:2108.04392*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. (2019). Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742.

Xie, S., Zheng, H., Liu, C., and Lin, L. (2018). Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*.

Yan, S., White, C., Savani, Y., and Hutter, F. (2021). Nas-bench-x11 and the power of learning curves. *Advances in Neural Information Processing Systems*, 34:22534–22549.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., and Hutter, F. (2019). Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

7 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? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see appendix. J
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see appendix. J
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] No theoretical results are presented.
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] The project repository is located at https://anonymous.4open.science/r/Accel-NASBench and will be de-anonimized after the peer-review process. However, some of the open-sourced code requires specialized hardware such as TPUs and FPGAs. Additionally, some of the code is based on in-house HPC routines that may need to be adapted to other HPC configurations. The main claims of the paper can be reproduced without need for specialized hardware as the dataset obtained using the specialized hardware has been provided. Nevertheless, we have included clear documentation on the hardware and software requirements needed to run the codes. Furthermore, we have provided instructive README, and requirements.txt files.
   (b) Did you include the raw results of running the given instructions on the given code and data? [Yes] Please see code repository.
   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [Yes] Yes, the code repository contains both the data and the scripts to generate the figures in the paper.
   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [Yes] To ensure code quality and style consistency, we followed the PEP8 style guides when writing our Python code. We used flake8, a popular linter, to enforce these style guidelines and ensure that our code met industry-standard best practices.
   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes] Standard practices are followed in training. Search space is detailed in Appendix. B and in code. Searched hyperparameters are in Appendix. F and their search process is detailed in README.md. Standard data splits as utilized by NASBench-301 have been used.
   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes] Evaluation techniques are compared in a way that ensures fairness: We utilize same sample budget for different NAS optimizers, and other hyperparameters are also chosen/searched in a way that ensures NAS optimizers are compared at an equal footing.
   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes] Please see Section. 4.1 and Appendix. E.

(h) Did you use the same evaluation protocol for the methods being compared? [Yes] Evaluation hyperparameters are selected in a way that ensures fairness in comparison between different NAS optimizers.

(i) Did you compare performance over time? [Yes] Please see Sec. 5.

(j) Did you perform multiple runs of your experiments and report random seeds? [Yes] Experiments using surrogates are performed using multiple runs with different random seeds (e.g., Sec. 5.1). Noise evaluation of surrogates is performed using different random seeds, where the seeds utilized are reported in the Tables in Sec. E.1 and Appendix. E. However, real optimizer runs and dataset collection is done using a single random seed owing to the high evaluation cost. Furthermore, impact of seeds on quality of dataset collection is also studied in Sec. 4.1.

(k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Where computationally feasible (e.g., Sec. 4.1 and 5.1). Surrogate noise evaluation is done in detail in Sec. E.1.

(l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes]

(m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see Sec. 3.1. We utilized a variety of compute sources: in-house and remote (e.g., CloudLab (Duplyakin et al., 2019)) clusters for accuracy data collection, GCP for TPUs utilized for throughput measurements and no-proxy evaluations, and in-house FPGAs for throughput/latency measurements.

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [Yes] Please see Sec. 3.3. The time required for hyperparameter tuning per surrogate is minor compared to the dataset collection time (roughly 35 mins per surrogate). Since we utilized SMAC3 for surrogate HPO, we will be releasing the HPO logs. For training recipes for the accuracy dataset collection utilized in this work, we obtained most hyperparameters from existing works that operate on the MnasNet/EfficientNetB0 search space such as Tan and Le (2019) and from FFCV (Leclerc et al., 2022).

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [Yes] Please see source code.

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The surrogate NAS benchmarks source code used in this research was based on NASBench-301 (Siems et al., 2020) which were distributed under the Apache 2.0 license. The licensing terms of the Apache 2.0 license do not require obtaining consent from individuals whose data is used or curated, and therefore consent was not needed for this study.

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

5. If you used crowdsourcing or conducted research with human subjects...

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

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]