NAS-Bench-360: Benchmarking Neural Architecture Search on Diverse Tasks

Renbo Tu
University of Toronto
renbo.tu@mail.utoronto.ca

Nicholas Roberts
University of Wisconsin
nick11roberts@cs.wisc.edu

Mikhail Khodak
Carnegie Mellon University
khodak@cmu.edu

Junhong Shen
Carnegie Mellon University
junhongs@andrew.cmu.edu

Frederic Sala
University of Wisconsin
fsala@wisc.edu

Ameet Talwalkar
Carnegie Mellon University
talwalkar@cmu.edu

Abstract

Most existing neural architecture search (NAS) benchmarks and algorithms prioritize well-studied tasks, e.g. image classification on CIFAR or ImageNet. This makes the performance of NAS approaches in more diverse areas poorly understood. In this paper, we present NAS-Bench-360, a benchmark suite to evaluate methods on domains beyond those traditionally studied in architecture search, and use it to address the following question: do state-of-the-art NAS methods perform well on diverse tasks? To construct the benchmark, we curate ten tasks spanning a diverse array of application domains, dataset sizes, problem dimensionalities, and learning objectives. Each new task is carefully chosen to interoperate with modern convolutional neural network (CNN) search methods while being far-afield from their original development domain. To speed up and reduce the cost of NAS research, for two of the tasks we release the precomputed performance of 15,625 architectures comprising a standard CNN search space. Experimentally, we show the need for more robust NAS evaluation of the kind NAS-Bench-360 enables by showing that several modern NAS procedures perform inconsistently across the ten tasks, with many catastrophically poor results. We also demonstrate how our benchmark and its associated precomputed results will enable future scientific discoveries by testing whether several recent hypotheses promoted in the NAS literature hold on diverse tasks. NAS-Bench-360 is hosted at https://nb360.ml.cmu.edu/.

1 Introduction

Neural architecture search (NAS) aims to automate the design of deep neural networks, ensuring performance on par with hand-crafted architectures while reducing human labor devoted to tedious architecture tuning [20]. With the growing number of application areas of ML, and thus of use-cases for automating it, NAS has experienced an intense amount of study in well-established machine learning domains, with significant progress in search space design [63, 42, 6], search efficiency [46], and search algorithms [55, 37, 53]. Notably, the field has largely been dominated by methods designed for and evaluated on benchmarks in computer vision [42, 56, 17], yet the use of NAS techniques may

Equal contribution

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

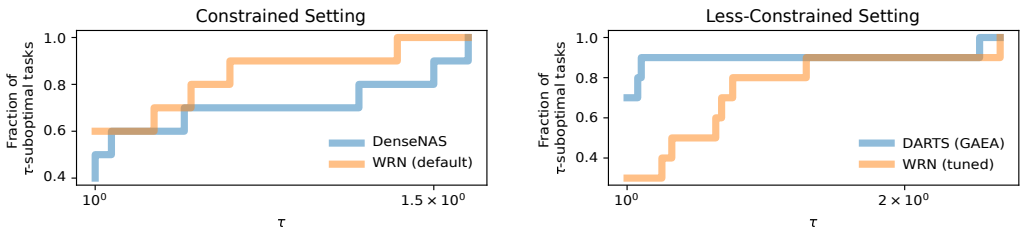

Figure 1: Performance profiles on NAS-Bench-360 comparing NAS methods (blue) to a fixed CNN (orange), specifically a Wide ResNet (WRN) [57]. Resource-constrained practitioners might be better off not using NAS (left), while less constrained practitioners can still benefit (right). The y-axis is the fraction of tasks on which error is within a factor of the optimal method, i.e. higher is better.

be especially impactful in under-explored or under-resourced domains where less is known about useful architecture design patterns. There have been a few recent efforts to diversify these benchmarks to settings such as vision-based transfer learning [18] and speech and language processing [43, 33]; however, evaluating NAS methods on such well-studied tasks using traditional CNN search spaces does not give a good indication of their utility on more far-afield applications, which have often necessitated the design of custom neural operations [10, 40].

We make progress towards studying NAS on more diverse tasks by introducing a suite of benchmark datasets drawn from various data domains that we collectively call NAS-Bench-360. This benchmark consists of an organized setup of ten suitable datasets that represent diverse application domains, dataset sizes, problem dimensionalities, and learning objectives. We also include a standard image classification task as a baseline point of comparison, as many new methods continue to be designed for that setting. Note that the core component of NAS-Bench-360 is not a typical NAS benchmark, which often involves precomputing all architectures in some fixed search space. In contrast, our contribution is explicitly intended to be agnostic of the search space being used, as different search spaces may work well for different tasks. Thus NAS-Bench-360 is a task-oriented NAS benchmark with the intended use-case of evaluating NAS method and search space pairs on a wide variety of domains. However, to aid research, three of our tasks—for two of which we contribute the precompute—do come with trained architectures from the NAS-Bench-201 search space [17].

Experimentally, we demonstrate the usefulness of NAS-Bench-360 by performing a set of analyses evaluating whether the success of NAS in computer vision is indicative of strong performance on the much broader set of problems to which NAS can be applied. Specifically, we report performance comparisons between NAS methods, investigate the validity of existing NAS hypotheses made solely on computer vision tasks, and extend an existing analysis of zero-cost proxies already-enabled by our benchmark [52]. From these analyses, we arrive at the following conclusions:

- Resource-constrained practitioners may be better of using a fixed CNN rather than NAS (Figure 1).
- NAS-Bench-201 analyses on computer vision tasks do not generalize to diverse tasks.
- Zero-cost proxies perform inconsistently on diverse tasks, corroborating earlier findings [52].

We have released all datasets, experiment code, precomputed models, seeds, and environments used in our experiments.[1] Releasing our code, random seeds, and environments in the form of Docker containers assures reproducibility of all experimental results presented in this work and encourages the same level of reproducibility for future research performed using NAS-Bench-360.

2 Related Work

Benchmarks have been critical to the development of NAS in recent years. This includes standard evaluation datasets and protocols, of which the most popular are the CIFAR-10 and ImageNet routines used by DARTS [42]. Another important type of benchmark has been tabular benchmarks such as NAS-Bench-101 [56], NAS-Bench-201 [17], NAS-Bench-1Shot1 [58], and TransNAS-Bench-101 [19]; these benchmarks exhaustively evaluate all architectures in their search spaces, which is made computationally feasible by defining simple searched cells. Consequently, they are less

[1] https://github.com/rtu715/NAS-Bench-360

expressive than the DARTS cell [42], often regarded as the most powerful search space in the cell-based regime. Notably, the full NAS-Bench-360 benchmark is not intended to be a tabular benchmark, i.e. we do not evaluate every architecture from a fixed search space on all ten of our tasks; instead, the focus is on the organization of a suite of tasks for assessing both NAS algorithms and search spaces, which would necessarily be restricted by fixing a search space for a tabular benchmark. Pre-computing on an expansive search space such as DARTS, with 10^{18} possible architectures, is computationally intractable. Architectures found on lesser search spaces are most likely suboptimal: a vanilla Wide ResNet (WRN) outperforms all networks in the NAS-Bench-201 search space on CIFAR-100. Nonetheless, we find that including precompute results for all of NAS-Bench-201 on two of our tasks is useful in evaluating various claims in the NAS literature centered on computer vision tasks.

While NAS methods and benchmarks have generally been focused on computer vision, recent work such as AutoML-Zero [48] and XD-operations [49] has started moving towards a more generically applicable set of tools for AutoML. However, even more recent benchmarks that do go beyond the most popular vision datasets have continued to focus on well-studied tasks, including vision-based transfer learning [18], speech recognition [43], and natural language processing [33]. We aim to go beyond such areas to evaluate the potential of NAS to automate the application of ML in truly under-explored domains. One analogous work to ours in the field of meta-learning is the Meta-Dataset benchmark of few-shot tasks [50], which similarly aimed to establish a wide-ranging set of evaluations for that field. For our inclusion of diverse tasks, we title our benchmark NAS-Bench-360 to resemble the idea of a 360-degree camera that covers all possible directions.

3 NAS-Bench-360: A Suite of Diverse and Practical Tasks

In this section, we introduce the NAS setting targeted by our benchmark, our motivation for organizing a new set of diverse tasks as a NAS evaluation suite, and our task-selection methodology. We report evaluations of specific algorithms on this new benchmark in the next section.

3.1 Neural Architecture Search: Problem Formulation and Baselines

For completeness and clarity, we first formally discuss the architecture search problem itself, starting with the extended hypothesis class formulation [37]. Here the goal is to use a dataset of points $x \in X$ to find parameters $w \in W$ and $a \in A$ of a parameterized function $f_{w;a} : X \to R_0$ that minimize the expectation $E_{x \sim D} f_{w;a}(x)$ for some test distribution D over X; here X is the input space, W is the space of model weights, and A is the set of architectures. For generality, we do not require the training points to be drawn from D to allow for domain adaptation, as is the case for one of our tasks, and we do not require the loss to be supervised. Note also that the goal here does not depend on computational or memory efficiency, which we do not focus on in our evaluations; our restriction is only that the entire pipeline can be run on an NVIDIA V100 GPU.

Notably, this formulation makes no distinction between the model weights w and architectures a, treating both as parameters of a larger model. Indeed, the goal of NAS may be seen as similar to model design, except now we include the design of an (often discrete) architecture space A such that it is easy to find an architecture $a \in A$ and model weights $w \in W$ whose test loss $E_{D} f_{w;a}$ is low using a search algorithm. This can be done in a one-shot manner—simultaneously optimizing a and w—or using the standard approach of first finding an architecture a and then keeping it fixed while training model weights w using a pre-specified algorithm such as stochastic gradient descent (SGD). This formulation divides NAS algorithms into two camps: one-shot, weight-sharing methods and non-weight-sharing ones such as random search, which operate by repeatedly sampling architectures and evaluating them. The formulation also includes non-NAS methods by allowing the architecture search space to be a singleton. When the sole architecture is a standard and common network such as WRN [57], this yields a natural baseline with an algorithm searching for training hyperparameters, not architectures. For our empirical investigation, we compare the performance of state-of-the-art NAS approaches against that of the three baselines: WRN, PerceiverIO [30], and XGBoost [7].

3.2 Task Selection: Motivation and Methodology

Curating a diverse, practical set of tasks for the study of NAS is our primary motivation behind this work. We observe that past NAS benchmarks focused on creating larger search spaces and more

Table 1: Task metadata for NAS-Bench-360. Metrics are standardized such that lower is better.

Task name	Size	Dim.	Type	Learning objective	Metric	New to NAS?
CIFAR-100 [34]	60K	2D	Point	Classify natural images into 100 classes	0-1 error	no, widely used
Spherical [10]	60K	2D	Point	Classify spherically projected images into 100 classes	0-1 error	X
NinaPro [4]	3956	2D	Point	Classify sEMG signals into 18 classes of hand gestures	0-1 error	X
FSD50K [24]	51K	2D	Point (multi-label)	Classify sound events in log-mel spectrograms with 200 labels	1 mAP	X
Darcy Flow [40]	1100	2D	Dense	Predict the final state of a fluid from its initial conditions	relative ℓ_2	no, used in [49]
PSICOV [3]	3606	2D	Dense	Predict pairwise distances between residuals from pairwise sequence features	MAE_8	no, used in [49]
Cosmic [59]	5250	2D	Dense	Predict probabilistic maps to identify cosmic rays in telescope images	1 - AUROC	X
ECG [9]	330K	1D	Point	Detecting atrial cardiac disease from ECG recordings	1 F1	X
Satellite [45]	1M	1D	Point	Classify satellite image pixel time series into 24 land cover types	0-1 error	X
DeepSEA [11]	250K	1D	Point (multi-label)	Predicting chromatin and binding states of RNA sequences	1 AUROC	no, used in [60, 61]

sophisticated search methods for neural networks. However, the utility of these search spaces and methods are only evaluated on canonical computer vision datasets. On a broader range of problems, whether these new methods can improve upon simple baselines remains an open question. This calls for the introduction of new datasets lest NAS research overfits to the biases of CIFAR-10 and ImageNet. By identifying these possible biases, future directions in NAS research can be better primed to suit the needs of practitioners and to increase the deployment of NAS.

Summarized in Table 1, NAS-Bench-360 consists of problems that are conducive to processing by convolutional neural networks, which includes a trove of applications associated with spatial and temporal data, spanning single and multiple dimensions. Most current NAS methods are not implemented to search for other types of architectures to process tabular data and graph data. Therefore, we have set this scope for our investigation. During the selection of tasks, diversity is our primary consideration. We define the following axes of diversity to govern our task-filtering process: the first is problem dimensionality, including both 2D with matrix inputs and 1D with sequence inputs; the second is dataset size, for which our selection spans the scale from 1,000 to 1,000,000; the third is problem type , divisible into tasks requiring a singular prediction (point prediction) and multiple predictions (dense prediction); fourth and finally, diversity is achieved through selecting tasks from various learning objectives from applications of deep learning, where introducing NAS could improve upon the performance of handcrafted neural networks.

In lieu of providing raw data, we perform data pre-processing locally and store the processed data on a public Amazon Web Services S3 data bucket with download links available on our website. Our data treatment largely follows the procedure defined by the researchers who provided them. This enhances reproducibility by ensuring the uniformity of input data for different pipelines. Additional information about the datasets, pre-processing, and augmentation steps are described in the Appendix.

Table 2: Performance of NAS and baselines across NAS-Bench-360. Methods are divided into efficient methods (e.g. DenseNAS and fixed WRN) that take 1-10 GPU-hours, more expensive methods (e.g. DARTS and tuned WRN) that take 10-100+ GPU-hours, and specialized methods (Auto-DL and AMBER). All results are averages of three random seeds, and lower is better for all metrics. The best performing method is shown in bold and the best non-expert-designed method is underlined.

Search space	Search algorithm	CIFAR-100	Spherical	Darcy Flow	PSICOV	Cosmic
WRN	default	23.350.05	85.770.71	0.0730.001	3.840.05	0.2450.02
DenseNAS	random	25.490.41	71.231.65	0.0710.006	3.700.06	0.3090.04
DenseNAS	original	25.980.38	72.990.95	0.1000.010	3.840.15	0.3830.04
Perceiver IO	default	70.040.44	82.570.19	0.2400.010	8.060.06	0.4850.01
XGBoost	default	84.834.15	96.920.02	0.0850.000	n/a	0.2320.00
WRN	ASHA	23.390.01	75.460.40	0.0660.000	3.840.05	0.2510.02
DARTS	GAEA	24.021.92	48.232.87	0.0260.001	2.940.13	0.2290.04
Auto-DL	DARTS	n/a	n/a	0.0490.005	6.730.73	0.4950.00
Expert	default	19.390.20	67.410.76	0.0080.001	3.350.14	0.1270.01

Search space	Search algorithm	NinaPro	FSD50K	ECG	Satellite	DeepSEA
WRN	default	6.780.26	0.920.001	0.430.01	15.490.03	0.400.001
DenseNAS	random	8.450.56	0.600.001	0.420.01	13.910.13	0.400.001
DenseNAS	original	10.171.31	0.640.002	0.400.01	13.810.69	0.400.001
Perceiver IO	default	22.221.80	0.720.002	0.660.01	15.930.08	0.380.004
XGBoost	default	21.900.70	0.980.002	0.560.00	36.360.02	0.500.000
WRN	ASHA	7.340.76	0.910.030	0.430.01	15.840.52	0.410.002
DARTS	GAEA	17.671.39	0.940.020	0.340.01	12.510.24	0.360.020
AMBER	ENAS	n/a	n/a	0.330.02	12.970.07	0.320.010
Expert	default	8.730.90	0.620.004	0.280.00	19.800.00	0.300.024

did not fit on a single V100 GPU.

4 Experimental design

Having detailed our construction of NAS-Bench-360, in this section we will establish the experimental setup for our analyses in the following section, which demonstrates the usefulness of NAS-Bench-360 for evaluating NAS methods on diverse tasks. We first specify the NAS methods and baselines we compare, followed by the details of the experimental setup and intended use of the benchmark. Finally, we provide details of the precomputed NAS-Bench-201 search space for two representative diverse tasks from NAS-Bench-360: NinaPro and Darcy Flow.

4.1 Baselines and Search Procedures

Our initial experiments follow two practitioners with different resource settings: one with enough compute to tune a WRN (less-constrained) and another who can only train it once with the default hyperparameters (constrained). Given these two scenarios, we compare against NAS methods that each practitioner would be able to run. In both cases, we focus on two well-known search paradigms: cell-based NAS (using DARTS [42]) and macro NAS (using DenseNAS [22]). We further compare these approaches to two customized NAS methods: Auto-DeepLab [41] for 2D dense prediction and AMBER [61] for 1D prediction, as well as general-purpose baselines: Perceiver IO [30] and XGBoost [7]. Additional details are provided in the Appendix.

4.2 Experimental Setup

Below we discuss the main reporting details of our empirical evaluation.

Table 3: Median rank and performance improvement over WRN across NAS-Bench-360.

Search space Search algorithm	WRN default	DenseNAS original	DenseNAS random	WRN ASHA	DARTS GAEA	Auto-DL DARTS	AMBER ENAS
Median rank	4.0	4.0	4.0	3.5	1.5	6.0[y]	1.0[y]
% better than WRN	0.0%	2.53%	0.0%	0.1%	14.6%	-75.3%[y]	20.0%[y]

relative improvement over the default (untuned) WRN baseline

[y] metric computed only on the subset of three tasks on which the method was evaluated

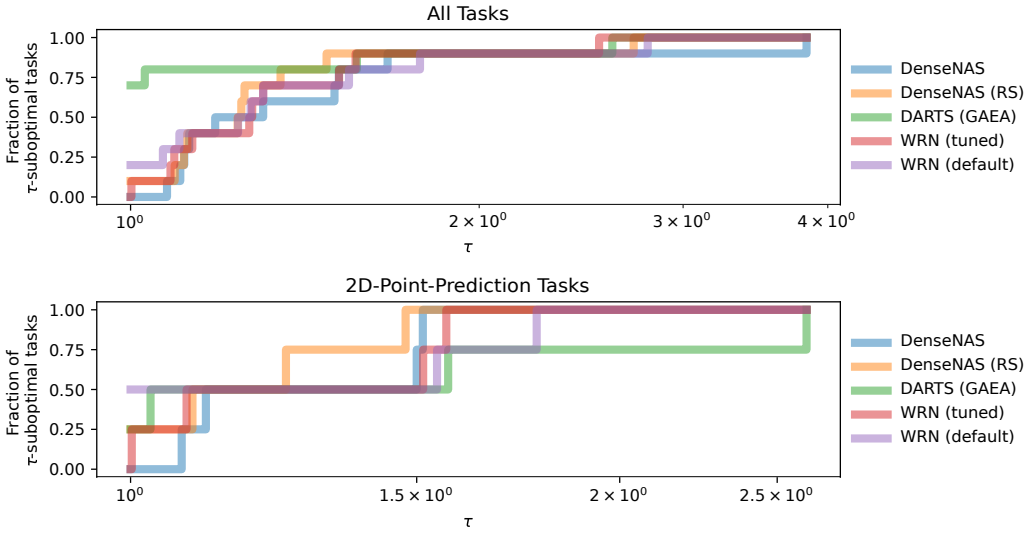

Figure 2: In our investigation of modern methods on NAS-Bench-360, we find that methods like GAEA DARTS can be strong in aggregate, as shown in the performance profiles on all tasks (top), but worse on salient subsets such as 2D point tasks (bottom). The y-axis is the fraction of tasks on which error is within a factor of the optimal method, i.e. higher is better.

- Hyperparameter tuning: As detailed in the Appendix, we use the same hyperparameter ranges across all tasks to tune WRN. We use ASHA [36] to search over these hyperparameters and give it a budget on each task that matches the total search and retraining budget of DARTS (GAEA).
- Aggregation metrics: To aggregate results across tasks, we use the median rank of each method and its performance improvement over WRN for direct comparison via a singe number, as demonstrated in Table 3. However, since these metrics can be sensitive to small differences in performance, we also employ performance profiles [14] to mitigate that effect while still accounting for outliers. As described in Figure 1, these curves denote for each the fraction of tasks on which a method is no worse than a -factor from the optimal. Concretely, we plot $\rho_s(\tau) = \frac{1}{|P|} |\{p \in P : \min_{s' \in S} \frac{error_{p;s}}{error_{p;s'}} \leq \tau\}|$ given some method $s \in S$ on tasks P.
- Software and hardware: We adopt the free, open-source software Determined[2] for experiment management, hyperparameter tuning, and cloud deployment. All experiments are performed on a single p3.2xlarge instance with a 16GB NVIDIA V100 GPU. While evaluation on NAS-Bench-360 indeed assumes access to at least a single V100 GPU, we reiterate that we provide the precomputed NAS-Bench-201 search space for two of our tasks in cases where GPU access is limited. Costs in GPU-hours are in the appendix.

[2] https://github.com/determined-ai/determined

4.3 Precomputing NAS-Bench-201 on NinaPro and Darcy Flow

The intended goal of NAS-Bench-360 is to evaluate the performance of NAS search method and search space pairs on diverse tasks, which precludes the precomputation of all architectures in general due to the lack of a single fixed search space. A complete lack of precomputed architectures would be perhaps limiting for many NAS researchers, who rely on precomputed NAS benchmarks when developing new search methods. In an effort to address this potential limitation, we precompute all architectures in the NAS-Bench-201 [17] search space on two representative tasks in NAS-Bench-360: NinaPro and Darcy Flow. We follow the same experimental procedure as in the original NAS-Bench-201 benchmark [17] to generate the precompute results, except where they vary the number of models trained for each architecture between one and three, we fix the number of trials per architecture to one. Note that NAS-Bench-201 already includes precompute for CIFAR-100, a dataset we include in NAS-Bench-360.

5 Analysis

We conclude our presentation of NAS-Bench-360 with three sets of analyses. The first, a performance analysis of NAS methods and fixed baselines across diverse tasks, reveals new insights about the capabilities and robustness of current NAS methods and demonstrates how our benchmark can enable critical next steps in NAS research. In our second analysis, we evaluate claims from the NAS literature originally made using computer vision tasks, and show that they do not generalize to diverse tasks; this demonstrates how NAS research can benefit from our contribution in the future. Finally, we extend an existing analysis of zero-cost proxy methods on diverse tasks that already uses NAS-Bench-360 [52].

5.1 Performance across diverse tasks using NAS-Bench-360

As discussed in Section 4.2, we start by considering two practitioners faced with a choice of spending their limited compute on a (possibly tuned) fixed-architecture CNN or trying to find a better architecture using NAS. With this study, we investigate whether modern NAS methods perform well beyond the tasks for which they were designed.

1. A surface-level analysis suggests that under light resource constraints, modern NAS in the form of DARTS (GAEA) is quite robust to a wide variety of tasks: Table 3 shows it is the highest-ranked domain-independent method and attains the most significant improvement over the fixed WRN baseline. The performance profile in Figure 2 (left) also seems favorable, although it is overtaken by tuned WRN at a higher -suboptimality. However, a closer look at 2D point tasks in Figure 2 (right) reveals that DARTS is quite poor there, despite its design domain being image classification; in particular, it performs very poorly on NinaPro and FSD50K. Furthermore, on tasks where it performs well, it can still lag behind expert architectures; for example, on Darcy Flow, networks that use FNO [40] or XD-operations [49] do much better. Overall, our results suggest that this practitioner can apply NAS and expect to see some improvement, but also risks catastrophically poor performance (e.g. FSD50K) or not getting truly state-of-the-art results (e.g. Darcy Flow).
2. Under stronger budget constraints, our experiments strongly suggest that a practitioner should simply apply the default WRN to their problem rather than undergo the additional complexity of using DenseNAS, as the latter attains little-to-no improvement over the former in Table 3 and has a usually-worse performance profiles Figure 2. On the other hand, DenseNAS performs well on FSD50K—it outperforms all methods even while DARTS (GAEA) fails.

These first experiments suggest that the modern NAS methods are not always robust to diverse tasks, especially under resource-constrained settings. We believe that NAS-Bench-360's main roles as a future benchmark include developing an understanding of the multi-domain performance of existing approaches and guiding research into better NAS methods. While the latter is beyond the scope of this paper, our additional experiments demonstrate how NAS-Bench-360 facilitates the former.

Notably, several of our results address the question of the relative importance of search space vs. search algorithm. For example, Table 3 shows that on DenseNAS, random search is nearly identical to the more sophisticated weight-sharing scheme of the original paper; the two algorithms' performance profiles are also difficult to distinguish in Figure 2. Furthermore, AMBER—a 1D NAS method whose search space includes larger-kernel convolutions for handling such tasks—does better than GAEA even though it uses an older search algorithm (ENAS). These both suggest that search space design, including the use of a wider variety of operations, may be at least as crucial for success as the search

algorithm. This point is reinforced by example tasks such as Darcy Flow, where architectures with more exotic operations substantially outperform our best results, as discussed earlier.

NAS-Bench-360 also reveals failure points of several methods, not just of general ones that usually perform quite well such as DARTS (GAEA) but also the objective-specific approach Auto-DL, which despite being designed for dense prediction tasks does poorly on all those considered here. Understanding when and why these performance drops happen is critical to developing a more robust NAS that is useful not just on average but in more challenging settings.

5.2 Do past NAS-Bench-201 analyses generalize to NAS-Bench-360?

Existing NAS-benches have been widely used for analyses such as (1) comparing performances of different architectures across tasks, (2) quickly evaluating search methods, and (3) investigating design choices that impact performance. In this section we show via the NAS-Bench-201 search space that the conclusions of past analyses cannot be assumed to hold on tasks beyond computer vision.

5.2.1 Architecture transferability

We start by using the precomputed results outlined in Section 4.3 to show in Figure 3 the rank of each architecture across different datasets, indexed on the x-axis by its rank on CIFAR-100. This reveals that while architecture rankings are highly correlated on image classification datasets—as pointed out by the authors of the original benchmark [17]—the rankings become uncorrelated when evaluated on a more diversified set of tasks. Therefore, NAS evaluations should be done across domains to verify true generalizability, and NAS-Bench-360 is especially useful for this purpose.

5.2.2 Search algorithm performance

Using the precomptued evaluations on two new datasets, we evaluate all ten typical NAS algorithms originally studied on NAS-Bench-201 [17]. The results are shown in the Appendix. With similar wall clock time, the non-weight-sharing NAS algorithms that we evaluate: REINFORCE [54], random search (RS) [5], regularized evolution (REA) [47], BOHB [21], and Hyperband [35] consistently perform well. Our results corroborate the strong performance of non-weight-sharing methods on this search space.

On the other hand, our experiments reveal some important differences for weight-sharing methods. In particular, unlike in past experiments on the NAS-Bench-201 search space, DARTS does not always yield a network of all skip-connection on Darcy Flow, despite this behavior on image classification and NinaPro. Instead, both first-order and second-order DARTS often pick convolution operations and sometimes achieve good performance, although still worse overall than the best non-weight-sharing methods. These results together with the ranking demonstrate that evaluating methods and search spaces on vision tasks alone does not give a full picture of their capabilities and limitations, a problem alleviated by NAS-Bench-360.

5.2.3 Operation redundancy

Our final analysis using the NAS-Bench-201 search space is to investigate the conclusions of a more recent study on the redundancy of operations [51]. We find that the operation redundancy phenomenon they outline is task-dependent and does not generalize to tasks beyond the three vision tasks—CIFAR-10, CIFAR-100, and ImageNet16-120—that they study. To conduct our study we follow their procedure to obtain "operation importance" distributions for each operation in the NAS-Bench-201 search space for NinaPro and Darcy Flow; additionally, we reproduce their results on CIFAR-10, CIFAR-100, and ImageNet16-120. Operation importance measures the incremental effect of each operation choice in the NAS-Bench-201 search space—1x1 convolutions (c1), 3x3 convolutions (c3), skip connections (skip), and 3x3 average pooling (ap3)—on performance [51]. The original analysis found that the operation importance distributions are roughly similar across the original NAS-Bench-201 computer vision datasets, which we confirm and show in Figure 4. However, we found that the operation importance distributions were drastically different for NinaPro and Darcy Flow, which we also show in Figure 4. Not only are their distributions different from those of the computer vision tasks in the original analysis, but the operation importance distribution for NinaPro differs significantly from that of Darcy Flow. This tells us that different operations are more useful for different tasks, and using NAS-Bench-360, we find that we cannot conclude that any of these

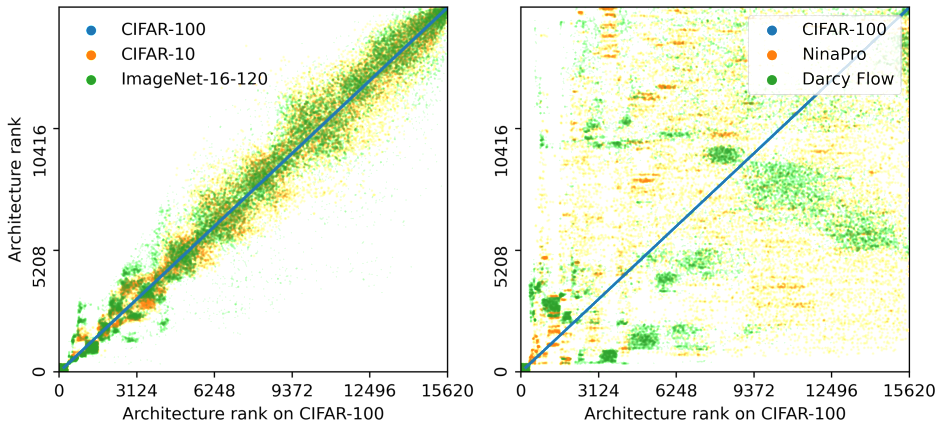

Figure 3: Architecture rankings between computer vision tasks correlate on NAS-Bench-201 [17] (left, sorted by performance on CIFAR-100) but are uncorrelated between CIFAR-100 and two NAS-Bench-360 tasks, NinaPro and Darcy Flow (right).

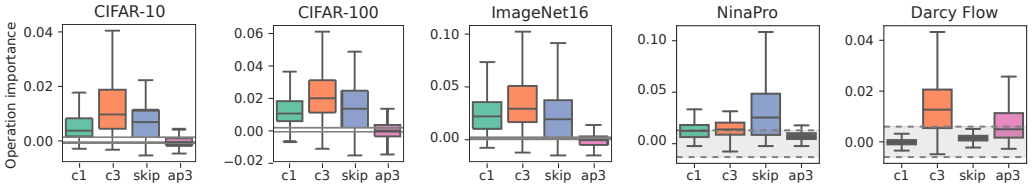

Figure 4: Different operations are important for different tasks. While prior work [51] shows that the operation importance distributions are stable across computer vision tasks—as shown by the high similarity of the three plots on the left—we find that they differ significantly for NinaPro and Darcy Flow.

Table 4: Performance comparison of TE-NAS and GAEA using the DARTS search space on CIFAR-100, Spherical, NinaPro, and Darcy Flow. Lower is better for all metrics.

	CIFAR-100	Spherical	NinaPro	Darcy Flow
TE-NAS	24.32	56.87	9.71	0.012
GAEA	24.02	48.23	17.67	0.026

operations are universally redundant or useful in a given search space across tasks. In other words, using NAS-Bench-360, we find that the original claim that "existing search spaces contain a high degree of redundancy" [51] does not hold when considering diverse tasks beyond computer vision.

5.3 Zero-cost proxies on diverse tasks

We conclude with an analysis of TE-NAS [8], a zero-cost proxy inspired by neural tangent kernel (NTK) analysis, on four NAS-Bench-360 tasks. Zero-cost proxies [44, 1] are the subject of a recent direction in NAS research that aims to construct quick-to-evaluate measures of architecture performance without doing any training. Recently, [52] evaluated several zero-cost proxies on tasks from NAS-Bench-360 (Spherical, NinaPro, and Darcy Flow), as well as on TransNAS-Bench-101 [18]. One major weakness of zero-cost proxies that they point out is that zero-cost proxies are not much more computationally efficient than weight-sharing methods, as the total compute cost is still dominated by the evaluation of the searched architecture [52]. For example, this renders TE-NAS in the DARTS search space comparable to GAEA DARTS in terms of computational efficiency. The authors of [52] also point out that the performance of different zero-cost proxies vary considerably across diverse datasets, even subject to the same search space. Performance may be strong on some tasks, but weak on others.

To expand such study of zero-cost proxies, we look at one that [52] do not consider—TE-NAS—and evaluate its performance on the DARTS space using four NAS-Bench-360 tasks: CIFAR-100, Spherical, NinaPro, and Darcy Flow. The results of this evaluation are shown in Table 4. Unlike many other zero-cost-proxies [44], the fact that TE-NAS is constructed from a domain-agnostic NTK analysis rather than experiments makes it a potential candidate for good performance on diverse tasks. However, Table 4 shows that performance does vary considerably across tasks, as observed for other proxies by [52]. In-particular, TE-NAS performs okay on NinaPro and beats all methods in Table 2 on Darcy Flow—where its performance approaches that of the expert-designed FNO [40]—but does poorly on Spherical. This evaluation adds evidence to existing scientific findings already enabled by NAS-Bench-360 [52] and provides additional evidence for the need to evaluate all NAS methods, including zero-cost proxies, on diverse tasks.

6 Conclusion

NAS-Bench-360 is a new performance benchmark consisting of ten diverse tasks derived from various fields of research and practice. It is designed for reproducible research on an academic budget that will guide the development of NAS methods and other automated approaches towards more robust performance across different domains. In initial results, we have demonstrated both the need for such a benchmark and the utility of NAS-Bench-360 specifically for developing new search spaces and algorithms. We also provide precompute architectures from the NAS-Bench-201 search space on two of the ten tasks. While the precomputed architectures on these two tasks are useful for analysis on their own, adding more precomputed search spaces and tasks is an area of further improvement. We welcome researchers to use the NAS-Bench-360 tasks to develop new procedures for automating ML.

Acknowledgments

We thank Maria-Florina Balcan for providing useful feedback. We also thank Hewlett Packard Enterprise for compute resources and the Determined AI open-source community for its support. This work was supported in part by DARPA FA875017C0141, the National Science Foundation grants IIS1705121, IIS1838017, IIS2046613, IIS-2112471, CCF2106707, the American Family Funding Initiative, the Wisconsin Alumni Research Foundation (WARF), an Amazon Web Services Award, a Facebook Faculty Research Award, funding from Booz Allen Hamilton Inc., a Block Center Grant, a Two Sigma Fellowship Award, and a Facebook PhD Fellowship Award. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of these funding agencies.

References

[1] Mohamed S. Abdelfattah, Abhinav Mehrotra, Lukasz Dudziak, and Nicholas D. Lane. Zero-cost proxies for lightweight NAS. In Procedings of the 9th International Conference on Learning Representations, 2021.

[2] Badri Adhikari. Deepcon: protein contact prediction using dilated convolutional neural networks with dropout. Bioinformatics, 36(2):470–477, 2020.

[3] Badri Adhikari. A fully open-source framework for deep learning protein real-valued distances. Scientific reports, 10(1):1–10, 2020.

[4] Manfredo Atzori, Arjan Gijsberts, Simone Heynen, Anne-Gabrielle Mittaz Hager, Olivier Deriaz, Patrick Van Der Smagt, Claudio Castellini, Barbara Caputo, and Henning Müller. Building the ninapro database: A resource for the biorobotics community. In 2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), pages 1258–1265. IEEE, 2012.

[5] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13:281–305, 2012.

[6] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In Proceedings of the 7th International Conference on Learning Representations, 2019.

[7] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[8] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four {gpu} hours: A theoretically inspired perspective. In International Conference on Learning Representations, 2021.

[9] Gari D Clifford, Chengyu Liu, Benjamin Moody, H Lehman Li-wei, Ikaro Silva, Qiao Li, AE Johnson, and Roger G Mark. Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017. In 2017 Computing in Cardiology (CinC), pages 1–4. IEEE, 2017.

[10] Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical CNNs. In Proceedings of the 6th International Conference on Learning Representations, 2018.

[11] ENCODE Project Consortium et al. The encode (encyclopedia of dna elements) project. Science, 306(5696):636–640, 2004.

[12] Ulysse Côté-Allard, Cheikh Latyr Fall, Alexandre Drouin, Alexandre Campeau-Lecours, Clément Gosselin, Kyrre Glette, François Laviolette, and Benoit Gosselin. Deep learning for electromyographic hand gesture signal classification using transfer learning. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 27(4):760–771, 2019.

[13] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery, 34(5):1454–1495, 2020.

[14] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. Mathematical programming, 91(2):201–213, 2002.

[15] Xuanyi Dong and Yezhou Yang. One-shot neural architecture search via self-evaluated template network. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 3680–3689, 2019.

[16] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four GPU hours. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[17] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In Proceedings of the 8th International Conference on Learning Representations, 2020.

[18] Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. TransNAS-Bench-101: Improving transferability and generalizability of cross-task neural architecture search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2021.

[19] Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Li Xiaodan, Tong Zhang, and Zhenguo Li. Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5247–5256, 2021.

[20] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. Journal of Machine Learning Research, 20(55):1–21, 2019.

[21] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In International Conference on Machine Learning, pages 1437–1446. PMLR, 2018.

[22] Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020.

[23] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open dataset of human-labeled sound events. arXiv preprint arXiv:2010.00475, 2020.

[24] Eduardo Fonseca, Jordi Pons Puig, Xavier Favory, Frederic Font Corbera, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. Freesound datasets: a platform for the creation of open audio datasets. In Hu X, Cunningham SJ, Turnbull D, Duan Z, editors. Proceedings of the 18th ISMIR Conference; 2017 oct 23-27; Suzhou, China.[Canada]: International Society for Music Information Retrieval; 2017. p. 486-93. International Society for Music Information Retrieval (ISMIR), 2017.

[25] John S Garofolo. Timit acoustic phonetic continuous speech corpus. Linguistic Data Consortium, 1993, 1993.

[26] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 776–780. IEEE, 2017.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[28] Shenda Hong, Yanbo Xu, Alind Khare, Satria Priambada, Kevin Maher, Alaa Aljiffry, Jimeng Sun, and Alexey Tumanov. Holmes: health online model ensemble serving for deep learning models in intensive care units. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1614–1624, 2020.

[29] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708, 2017.

[30] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In International Conference on Learning Representations, 2022.

[31] David Josephs, Carson Drake, Andy Heroy, and John Santerre. semg gesture recognition with a simple model of attention. In Machine Learning for Health, pages 126–138. PMLR, 2020.

[32] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Kathryn Tunyasuvunakool, Olaf Ronneberger, Russ Bates, Augustin Žídek, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Anna Potapenko, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Martin Steinegger, Michalina Pacholska, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. High accuracy protein structure prediction using deep learning. In Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book), 2020.

[33] Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, and Evgeny Burnaev. NAS-Bench-NLP: Neural architecture search benchmark for natural language processing. arXiv, 2020.

[34] Alex Krizhevksy. Learning multiple layers of features from tiny images. Technical report, 2009.

[35] Liam Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research, 18(185):1–52, 2018.

[36] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. arXiv preprint arXiv:1810.05934, 2018.

[37] Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Geometry-aware gradient algorithms for neural architecture search. In Proceedings of the 9th International Conference on Learning Representations, 2021.

[38] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, 2019.

[39] Lisha Li, Kevin G Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In ICLR (Poster), 2017.

[40] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In Proceedings of the 9th International Conference on Learning Representations, 2021.

[41] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 82–92, 2019.

[42] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In Proceedings of the 7th International Conference on Learning Representations, 2019.

[43] Abhinav Mehrotra, Alberto Gil, C. P. Ramos, Sourav Bhattacharya, Łukasz Dudziak, Ravichander Vipperla, Thomas Chau, Samin Ishtiaq, Mohamed S. Abdelfattah, and Nicholas D. Lane. NAS-Bench-ASR: Reproducible neural architecture search for speech recognition. In Proceedings of the 8th International Conference on Learning Representations, 2021.

[44] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. Neural architecture search without training. In Proceedings of the 38th International Conference on Machine Learning, 2021.

[45] François Petitjean, Jordi Inglada, and Pierre Gançarski. Satellite image time series analysis under time warping. IEEE transactions on geoscience and remote sensing, 50(8):3081–3095, 2012.

[46] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Proceedings of the 35th International Conference on Machine Learning, 2018.

[47] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019.

[48] Esteban Real, Chen Liang, David R. So, and Quoc V. Le. AutoML-Zero: Evolving machine learning algorithms from scratch. In Proceedings of the 37th International Conference on Machine Learning, 2020.

[49] Nicholas Roberts, Mikhail Khodak, Tri Dao, Liam Li, Chris Ré, and Ameet Talwalkar. Rethinking neural operations for diverse tasks. arXiv, 2021.

[50] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In Proceedings of the 8th International Conference on Learning Representations, 2020.

[51] Xingchen Wan, Binxin Ru, Pedro M Esperança, and Zhenguo Li. On redundancy and diversity in cell-based neural architecture search. In International Conference on Learning Representations, 2022.

[52] Colin White, Mikhail Khodak, Renbo Tu, Shital Shah, Sébastien Bubeck, and Debadeepta Dey. A deeper look at zero-cost proxies for lightweight nas. In ICLR Blog Track, 2022. https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/.

[53] Colin White, Willie Neiswanger, and Yash Savani. BANANAS: Bayesian optimization with neural architectures for neural architecture search. In Proceedings of the 35th AAAI Conference on Artificial Intelligence, 2021.

[54] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3):229–256, 1992.

[55] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: Partial channel connections for memory-efficient architecture search. In Proceedings of the 8th International Conference on Learning Representations, 2020.

[56] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-Bench-101: Towards reproducible neural architecture search. In Proceedings of the 36th International Conference on Machine Learning, 2019.

[57] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Proceedings of the British Machine Vision Conference, 2016.

[58] Arber Zela, Julien Siems, and Frank Hutter. NAS-Bench-1Shot1: Benchmarking and dissecting one-shot neural architecture search. In Proceedings of the 8th International Conference on Learning Representations, 2020.

[59] Keming Zhang and Joshua S Bloom. deepcr: Cosmic ray rejection with deep learning. The Astrophysical Journal, 889(1):24, 2020.

[60] Zijun Zhang, Evan M Cofer, and Olga G Troyanskaya. Ambient: accelerated convolutional neural network architecture search for regulatory genomics. bioRxiv, 2021.

[61] Zijun Zhang, Christopher Y Park, Chandra L Theesfeld, and Olga G Troyanskaya. An automated framework for efficiently designing deep convolutional neural networks in genomics. Nature Machine Intelligence, 3(5):392–400, 2021.

[62] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. Nature methods, 12(10):931–934, 2015.

[63] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

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] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix.
   (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] We do not include any theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A] We do not include any theoretical results.
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Instructions are described in the paper; code and data are available on our dedicated website.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In both Section 3 and in the Appendix.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 2

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix.

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] See Appendix.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] New assets are on our website.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Appendix.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix.

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] We do not crowdsource any data or conduct research with human subjects.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We do not crowdsource any data or conduct research with human subjects.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We do not crowdsource any data or conduct research with human subjects.