Proxy-Driven Estimation of Distribution Algorithm for Efficient Neural Architecture Search

André Silva, Mário Haddad-Neto, Rayol Mendonca-Neto,
Vanessa Câmara, Moysés Lima, Luis Uebel and Luiz Cordovil-Jr
Sidia Science and Technology Institute, Manaus, Brazil
E-mail: {andre.fernandes, mario.neto-e, rayol.neto, vanessa.camara, moyses.mendes, luis.uebel, luiz.cordovil}@sidia.com

***Abstract*—Automating the design of Deep Neural Networks for specific applications using Neural Architecture Search (NAS) has become increasingly attractive. However, traditional NAS methods are often hindered by the high cost of evaluating numerous candidate networks, rendering them impractical for large-scale searches. In this work, we assessed a proxy-driven Estimation of Distribution Algorithm (EDA) for optimization of topological and quantitative hyperparameters. By leveraging the ideas of designing network design spaces and iterated sampling algorithms with zero-cost proxies, our approach reduces the search cost significantly. Furthermore, the proxy-driven EDA achieves near-optimal results in the NATS-Bench topology and size search spaces on CIFAR-10, CIFAR-100, and ImageNet16-120 datasets for image classification. The results show that a proxy-driven EDA can reliably explore large, mixed-type design spaces with negligible computational overhead, opening the door to rapid, resource-friendly architecture discovery.**

***Index Terms*—Neural Architecture Search, Zero-Cost Proxies, Estimation of Distribution Algorithm, Image Classification**

I. Introduction

Deep Neural Networks (DNN) are the state-of-the-art machine learning models for processing unstructured data such as images, audio, video, and text [1]. They are crucial components of cutting-edge technology for autonomous vehicles, medical image analysis, surveillance, safety visual inspection, and generative artificial intelligence. However, it is not a trivial task to develop neural networks tailored for a specific application. It requires many trial-and-error attempts in the iterative process of designing a general architecture, fine-tuning the hyperparameters, and validating the performance of candidate models. This comprises configuration of quantitative hyperparameters (e.g., number of neurons, number of channels in convolutional layers, number of attention layers, etc.) and non-quantitative hyperparameters (e.g., activation functions, attention functions, residual connections, and topology). Besides time-consuming, it is also an expensive task due to hardware and energy requirements [2].

Neural Architecture Search (NAS) algorithms have been proposed to automate this process of designing DNNs and optimizing their hyperparameters [3], [4]. Evolutionary NAS algorithms are the most popular [5]–[12]. These algorithms are inspired by the convergence of neuroscience and evolutionary theory, which found its expression in computer science very early [13]. Estimation of Distribution Algorithms (EDA) have also been proposed for NAS [14]–[18]. EDAs have many advantages over other evolutionary algorithms and conventional metaheuristics (e.g., adaptive operators, problem structure, prior knowledge exploitation, and reduced memory requirements) [19]. However, their applications on NAS is still underexplored. Besides evolutionary algorithms, reinforcement Learning NAS has also enjoyed great success [20]–[23]. Gradient-based NAS [24]–[26] and other search strategies have also been proposed [27], [28].

The success of NAS depends on a well-designed search space. A block (or cell) is a design motif that is replicated along the whole network architecture. This is a successful design strategy that was employed in many well-known architectures such as ResNet [29], ResNeXt [30], Inception models [31]–[33], and DenseNet [34]. Many NAS algorithms define a search space with these block designs [7], [9]–[11], [23]. Employing that strategy simplifies the search space significantly and prevents getting lost in the vast space of all possible networks. On the other hand, these algorithms miss the opportunity to discover new patterns when they rely on blocks designed by humans. Some NAS algorithms are capable of automatically designing novel block types [35], whereas others do not rely on block designs and employ other strategies to constrain the search space [8], [15], [27].

However, NAS is an expensive process because it has to validate the performance of many candidate DNNs. Many cost-saving strategies have been proposed for NAS such as weight inheritance, early stopping policy, reduced training set, reduced population, and population memory [4]. Zhou et al. [36] made a comprehensive study about common proxy tasks such as reducing the network size, the resolution of input images, the number of training epochs, and the training dataset. Mellor et al. [37] proposed a zero-cost proxy to estimate DNNs' performance without training. Their proxy requires a single forward pass on a mini-batch of data, which reduces the NAS cost significantly. Their study laid the groundwork for subsequent investigations into zero-cost proxies for NAS. Abdelfattah et al. [38] proposed zero-cost proxies for NAS based on several pruning criteria originally developed for pruning-at-initialization algorithms. Keserwani et al. [39] introduce a reformulation of zero-cost proxies based

This article is the result of the R&D&I project *ASTRO*, carried out by Sidia Science and Technology Institute in partnership with Samsung Eletrônica da Amazônia Ltd., using resources from Brazilian Federal Law 8,387/1991, and its disclosure and publicity are in accordance with the provisions of article 39 of Brazilian Decree 10,521/2020.

on pruning-at-initialization criteria. Their method combines kernel strength with a receptive field weighting mechanism. Zhou et al. [40] proposed a zero-cost proxy for Vision Transformers based on synaptic diversity and synaptic salience. Qiao et al. [41] present a NAS framework specifically designed for microcontroller unit deployment. Their method optimizes architectures using a hybrid objective that combines a custom neural tangent kernel spectrum, the number of linear region counts, and hardware proxies. Ning et al. [42] and White at al. [43] made comprehensive assessment of proxies for NAS.

Radosavovic et al. [44] took a different approach to NAS by proposing a methodology to find a good design space rather than optimizing individual neural networks. By observing the general trends in populations of DNNs, it aims at understanding the design principles of the top-performing networks. It is an iterative process that works as follows. First, it starts with a general design space called AnyNet, which contains block-based ResNeXt-like models. Then, statistics are computed in a sample of random networks. After that, a new design principle is discovered (or hypothesized) by analyzing the best networks in the sample. Essentially, a design principle is a constraint in the distribution of hyperparameters that is likely to maximize the models' performance. Finally, the process continues by sampling random networks from the new design space. The process results in a refined design space called RegNet. This methodology can improve NAS significantly. Although it could potentially be applied to other design spaces beyond AnyNet, the reported results are limited to that space and comprising only optimization of quantitative hyperparameters. Furthermore, this is a trial-and-error process that relies heavily on the insights of human designer, not to mention it requires training many more DNNs than a typical NAS algorithm would.

Building on the RegNet design-space methodology [44], Silva et al. [17] and Haddad-Neto et al. [18] introduced a proxy-driven EDA to automate the exploration of ResNeXt-like architectures and hybrid Vision Transformers, respectively. In their work, the proxy-driven EDA served primarily as an analysis tool for evaluating how zero-cost proxies influence search-space traversal, but its viability as a stand-alone NAS algorithm was not demonstrated. Furthermore, the method was restricted to tuning only quantitative hyperparameters.

Conversely, other EDA-based NAS frameworks—such as Architecture Search by Estimation of network-structure Distributions (ASED) proposed by Muravev et al. [15]—can optimize non-quantitative hyperparameters. Zhao et al. [14] presented an EDA-based framework named EDNAS. Regarding only topological hyperparameters, the method derives candidate solutions from a directed acyclic graph representing the possible search space. The authors uses a parameter sharing approach and by modeling the distribution of the top-performing candidate solutions from an iterative sampling approach, their probabilities are computed and employed to generate the solutions of the next generation until convergence. Nonetheless, these approaches still rely on the costly training of deep neural networks during the search, which limits their efficiency.

Consequently, there remains a need for a proxy-driven EDA method that (i) treats the estimation of distribution as a genuine search algorithm, (ii) accommodates non-quantitative hyperparameters, and (iii) reduces or eliminates the dependence on full model training. Our work addresses these gaps by the following contributions:

- We assess a proxy-driven NAS algorithm to optimize topological and quantitative hyperparameters. The algorithm builds on the systematic design-space methodology and integrates ideas from EDA-inspired NAS for both quantitative and non-quantitative hyperparameter optimization. By employing inexpensive zero-cost proxies, the algorithm can evaluate candidate architectures without full training, significantly reducing search cost.
- The proxy-driven EDA was tested on NATS-Bench [35], which comprises topology and size search spaces across three standard image-classification benchmarks (CIFAR-10, CIFAR-100, and ImageNet-16-120). The proxy-driven EDA attains performance close to the global optimum while completely avoiding training during the search, thereby establishing an excellent trade-off between search efficiency and final model quality.

This paper is organized as follows. Section II provides a concise overview of the NATS-Bench architectures, covering both topology and size search spaces. Section III formally defines the proposed algorithm. Section IV describes the methodology employed in this research, including experimental settings. Section V presents the experimental results. Finally, Section VI summarizes our findings and highlights potential future research directions.

II. NATS-Bench Architectures

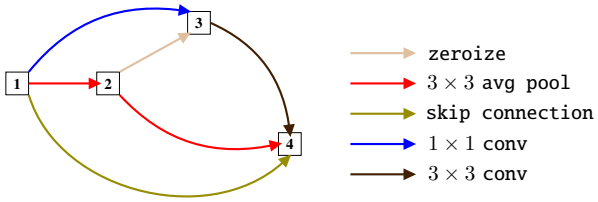

Fig. 1. Cell structure in NATS-Bench architectures.

NATS-Bench contains data from thousands of pre-trained DNNs on CIFAR-10, CIFAR-100, and ImageNet16-120 [35]. The architectures in the search spaces have a general structure that starts with a 3×3 convolution with 16 output channels followed by: a batch normalization, a 1st stack of cells, a 1st residual block, a 2nd stack of cells, a 2nd residual block, a 3rd stack of cells, and a global average pooling layer, whose output is flattened and fed to a fully connected layer followed by softmax. The cell is a structure that will be replicated along the network backbone (see Figure 1). The residual blocks down-sample the input tensor with stride $= 2$ and,

for dimension compatibility, the residual connection consists of a 2×2 average pooling layer with stride $= 2$ followed by a 1×1 convolution. The NATS-Bench provides a topology search space for optimization of the cell structure and a size search space for optimization of the number of output channels in cell stacks and residual blocks.

The *topology search space* contains networks with all possible cell designs. A cell is represented as a densely connected directed acyclic graph with four nodes and six edges as illustrated in Figure 1. An edge represents a transformation and a node represents the sum of all transformations pointing to it. This search space contains five representative transformations: zeroize, skip connection, 1×1 conv, 3×3 conv, and 3×3 avg pool. The zeroize transforms the signal into a zero vector, which effectively disables that connection. The skip connection is a common residual connection. The 1×1 conv and 3×3 conv are abbreviations for the sequence: ReLU, convolution, and batch normalization. For the architectures in the topology search space, each cell is stacked 5 times, with the number of output channels set to 16, 32 and 64 for the first, second and third stages, respectively. Since there are 5 possibilities for edge transformations and there are 6 edges, the topology search space contains $5^6 = 15,625$ architectures.

The *size search space* contains networks with different number of output channels for the cells and residual blocks. The number of channels in each layer is chosen independently from $\{8, 16, 24, 32, 40, 48, 56, 64\}$. For the architectures in the size search space, each cell is stacked once, and all the architectures have the same cell configuration, defined as the configuration of the best architecture on CIFAR-100 from the topology search space. Since there are 8 possible values of output channels and 5 hyperparameters to configure, the size search space contains $8^5 = 32,768$ architectures.

III. Proxy-driven EDA for NAS

The proxy-driven EDA assessed in this work is defined in Algorithm 1, which is implemented as an incremental univariate EDA. The algorithm is based on the ideas about design of network design spaces [44] and on the iterated sampling algorithm to discover design principles automatically [17], [18]. In this work, we demonstrate that the proxy-driven EDA can optimize both topological and quantitative hyperparameters. Furthermore, this can be done without training by leveraging zero-cost proxies.

The model M_g represents a probabilistic description of the most promising hyperparameter configurations at the gth generation. At every iteration the top-scoring DNNs are used to estimate the distribution of each hyperparameter, which becomes the next generation's distribution M_{g+1}. Algorithm 1 starts from a completely uninformative model M_1 in which every hyperparameter is assigned a uniform distribution over its admissible domain. For instance, in NATS-Bench topology search space each edge $e_{i,j}$ is initially draw uniformly from the set $e_{i,j} \in \{$ zeroize, skip connection, 1×1 conv, 3×3 conv, 3×3 avg pool $\}$, whereas in the size search space the number of channels c_i is draw uniformly from

Algorithm 1 Proxy-Driven EDA for NAS

Input: population size N, sample size R, selection size D,
Output: model M_g

```
g ← 0
M_1 ← uniform distribution for each hyperparameter
repeat
    g ← g + 1
    population ← generate N random DNNs from M_g
    calculate a proxy score for all DNNs in population
    best ← ∅
    for i = 1 to D do
        sample ← resample R DNNs from population
        best.add( top-scoring network from sample )
    end for
    M_{g+1} ← estimate new model from best
until M_{g+1} ≈ M_g
return M_g
```

$c_i \in \{8, 16, 24, 32, 40, 48, 56, 64\}$. Once initialized, the algorithm proceeds iteratively performing four steps: sampling, evaluation, selection and model update.

In the sampling step, a population of N candidate DNNs is generated by drawing hyperparameter values from the current distributions M_g. Notice that the population is entirely replaced at each iteration, which makes the algorithm an incremental EDA.

In the evaluation step, each candidate DNN is scored by a zero-cost proxy (e.g., the proxies proposed by Mellor et al. [37] or by Abdelfattah [38]). This step is crucial to guarantee the efficiency of the algorithm since zero-cost proxies do not require training. Without proxies, this would require fully training of N models at each iteration of the algorithm, which is expensive in terms of time and energy consumption.

In the selection step, the algorithm creates a pool of the best DNNs by employing the resampling strategy of empirical bootstrap. First, it resamples with replacement R DNNs from the population. Then, it selects the top-scoring DNN according to its proxy score. The selected top-scoring DNN is added to the pool of best DNNs. This is repeated D times.

In the model update step, for each hyperparameter, the algorithm estimates a new distribution from the D best DNNs in the pool and then updates the next generation model M_{g+1}. It is important to notice that throughout the process we assume independence among all hyperparameters, which makes the algorithm a univariate EDA.

For quantitative hyperparameters, the update step uses the method of RegNet design space [44], i.e., it estimates the 95% confidence intervals (CI) with the 2.5-percentile and the 97.5-percentile as lower and upper limits of the CI, respectively. Then, the previous distribution is replaced by a new uniform distribution in which the upper and lower limits are given by the estimated percentiles.

For non-quantitative hyperparameters, the update step estimates new probabilities based on the relative frequency of

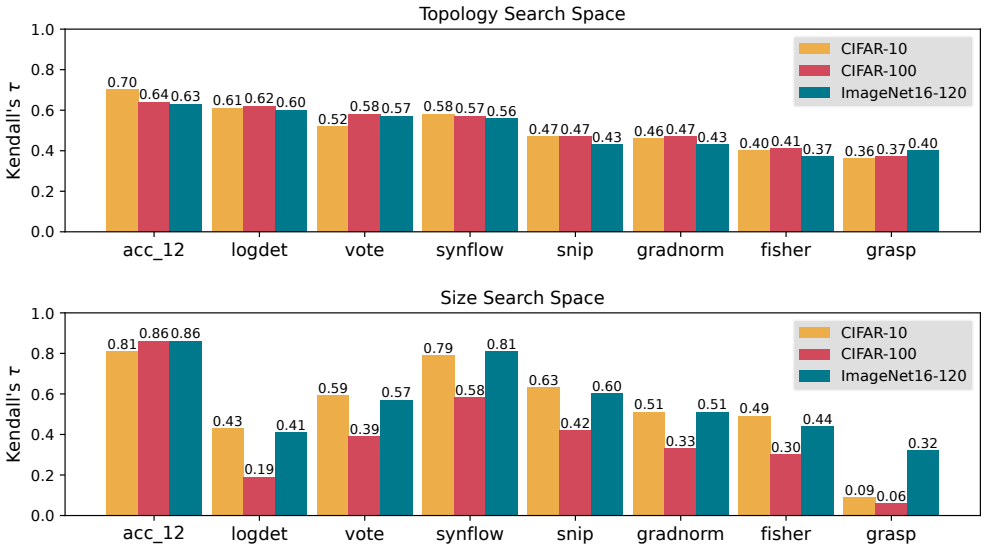

Fig. 2. Kendall rank correlation coefficient between the proxies and the validation accuracy on NATS-Bench topology and size search spaces.

the values occurring in the pool of best DNNs. For instance, if 94% of the best DNNs have $e_{1,2} = 3 \times 3$ conv, 6% have $e_{1,2} = 1 \times 1$ conv, and none has zeroize, skip connection, nor 3×3 avg pool, then the probabilities of the values of $e_{1,2}$ are updated to these frequencies. Usually, EDA algorithms update the probabilities according to a formula such as $p_{g+1} = (1 - \alpha) \times p_g + \alpha \times p_m$, where p_g is the current probability distribution, p_m is a new probability distribution estimated from the selected best DNNs in the current generation, and α is a learning rate. The EDA implementation in this work is equivalent to setting $\alpha = 1$, which greedily focus on current best distribution for fast convergence.

Finally, if the update step causes no significant change in the previous probability model, that is, if $M_{g+1} \approx M_g$ given a numerical tolerance threshold (e.g., $\epsilon = 10^{-5}$), then the algorithm stops and returns the current model. Otherwise, it repeats the process until convergence.

IV. Methodology

We conducted experiments with the proxy-driven EDA using eight zero-cost proxies: the `logdet` proxy proposed by Mellor et al. [37], the proxies adapted from pruning-at-initialization salience criteria (namely `grasp`, `synflow`, `snip`, and `fisher`) the `gradnorm` proxy, and the majority voting proxy (`vote`) that selects the best architecture according to the rankings produced by `logdet`, `synflow`, and `snip` as defined by Abdelfattah et al. [38]. Since the accuracy obtained after a few training epochs is also a common performance proxy [36], we compare the zero-cost proxies with the validation accuracy at the 12th epoch (`acc_12`). The proxies `logdet`, `synflow`, `fisher`, `grasp`, `snip`, and `gradnorm` are computed with their publicly available implementations, and `acc_12` is queried from the NATS-Bench API. The zero-cost proxies are computed on a mini-batch of 128 random images, with the exception of `synflow`, which requires no data.

In the topology search space, the proxy-driven EDA is configured to optimize the edges $e_{1,2}$, $e_{1,3}$, $e_{2,3}$, $e_{1,4}$, $e_{2,4}$, and $e_{3,4}$ from the initial search space $e_{i,j} \in \{$zeroize, skip connection, 1×1 conv, 3×3 conv, 3×3 avg pool$\}$. In the size search space, the algorithm is configured to optimize the number of output channels of the three cell stacks (c_1, c_2, and c_3) and the number of output channels of the two residual blocks (r_1 and r_2) from the initial search space $c_1, c_2, c_3, r_1, r_2 \in \{8, 16, 24, 32, 40, 48, 56, 64\}$. In both search spaces, the algorithm is executed with population size $N = 500$, sample size $R = 400$, selection size $D = 10,000$, and numerical tolerance threshold $\epsilon = 10^{-5}$. The zero-cost proxies are computed on a mini-batch of 128 random images from the training set. When the search is complete, we compare the hyperparameters and the validation accuracy of the fully trained models with the global optimum from each search space. We further assess the accuracy distribution of DNNs at each iteration of the algorithm and compare it with the proxies' Kendall rank correlation coefficient (Kendall's τ).

V. Results

Figure 2 shows the Kendall's τ for each proxy on NATS-Bench search spaces. The `acc_12` proxy obtained the best rank correlation. However, the zero-cost proxies showed competitive performance. Among the zero-cost proxies, `logdet` achieved better Kendall's τ in the topology search space, whereas `synflow` is better in the size search space. The proxies tend to have a better performance on the size search space. The exceptions are `grasp` and `logdet`, which actually perform significantly better on topology search space. Nonetheless, `grasp` does not show a robust correlation across the different datasets.

Proxy-driven EDA results on the topology and size search spaces are summarized in Tables I and II, respectively. In most cases, the proxies converged to subspaces with only

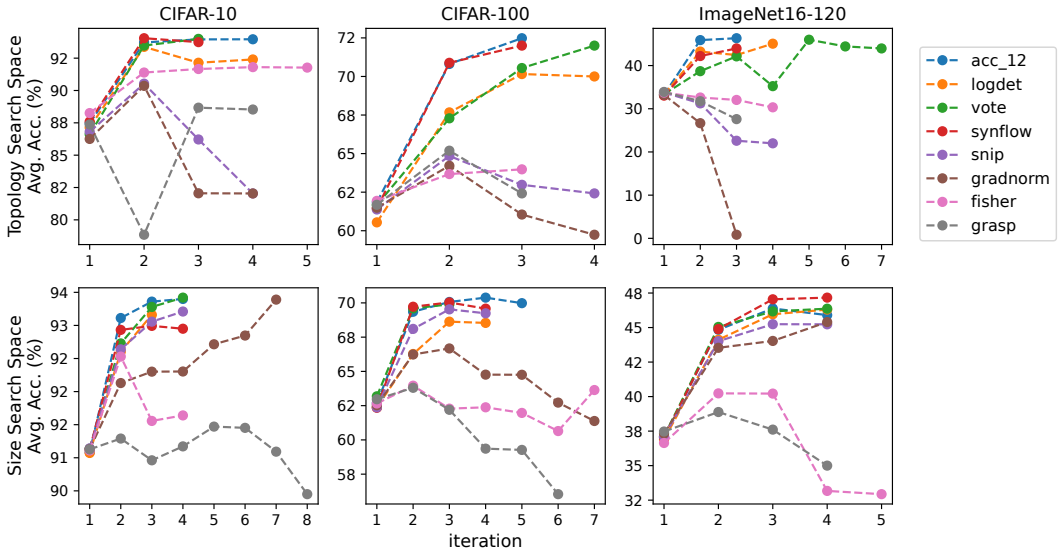

Fig. 3. Evolution of average population accuracy on NATS-Bench topology and size search spaces.

one architecture. The exception is `vote`, which converged to subspaces with multiple options of output channels in the size search space on CIFAR-100 and ImageNet16-120. The proxies are sorted by validation accuracy and the global optimum is provided for comparison at the bottom, which, in this case, it is the best architecture in the NATS-Bench for each dataset. These results show that zero-cost proxies are capable of achieving near-optimum performance in topological and quantitative hyperparameter optimization. This is a remarkable achievement, specially considering that they do not require training.

In the topology search space, `acc_12` obtained excellent performance close to the global optimum on all datasets. The zero-cost proxies `vote`, `synflow`, and `logdet` achieved competitive results, whereas `fisher`, `grasp`, and `gradnorm` obtained the worst results. In the size search space, the proxies' performance vary significantly when compared to the topology search space. For instance, `gradnorm` achieved inconsistent results getting near optimum performance on CIFAR-10 and near worst on CIFAR-100. In addition, zero-cost proxies outperformed `acc_12` on all datasets in the size search space.

Figures 3 and 4 provide a straightforward visual assessment of the proxies' performance by showing the evolution of population accuracy at each iteration of the search algorithm. These plots demonstrate that `acc_12`, `logdet`, `vote`, and `synflow` effectively lead the search process to better subspaces by assimilating the design principles of top-performing architectures, whereas other proxies are either inconsistent or consistently poor. From these results, it is evident that the proxy-driven EDA with `vote` achieved the best overall performance among the zero-cost proxies yielding excellent trade-off between search cost and model accuracy.

In addition, our experiments revealed discrepancies between the Kendall's τ (Figure 2) and the actual performance obtained by the search algorithm (Tables I and II). Specifically, we found that `vote` consistently outperformed other proxies and achieved near-optimal results on all datasets in the size search space. In contrast, `logdet` was outperformed by `vote` and `synflow` on CIFAR datasets in the topology search space. Also according to Kendall's τ, `gradnorm` was not expected to be the worst proxy on topology search space, which highlights the limitations of Kendall's τ as a metric to assess proxies for NAS. The issue with rank correlation metrics like Kendall's τ and Spearman's ρ is that they assign equal weight to all architectures. In practice, achieving perfect rank correlation is not necessary. What matters more is the proxy's ability to differentiate between good and poor architectures. For a detailed discussion of this problem, please refer to Ning et al. [42]. To this end, Figures 3 and 4 provide a better straightforward visual assessment of the proxies capacity to select good architectures at each iteration of the search algorithm.

Finally, we observe that the search time when using zero-cost proxies is orders of magnitude lower than when using the `acc_12` metric. Computing `acc_12` requires a full 12-epoch training run—i.e., forward and backward propagation over the entire training set—whereas each zero-cost proxy can be evaluated with a single forward pass on a mini-batch of only 128 images. This dramatic reduction in search time underscores the practical advantage of zero-cost proxies for rapid architecture evaluation.

VI. Conclusion

In this work we assessed a proxy-driven EDA that optimizes the topology (choice of operations and skip connections) and size (channel dimensions) of convolutional neural networks. By treating a zero-cost proxy as the fitness function, the algorithm iteratively refines a probabilistic model of promising hyperparameters, eliminating the need for costly full-network

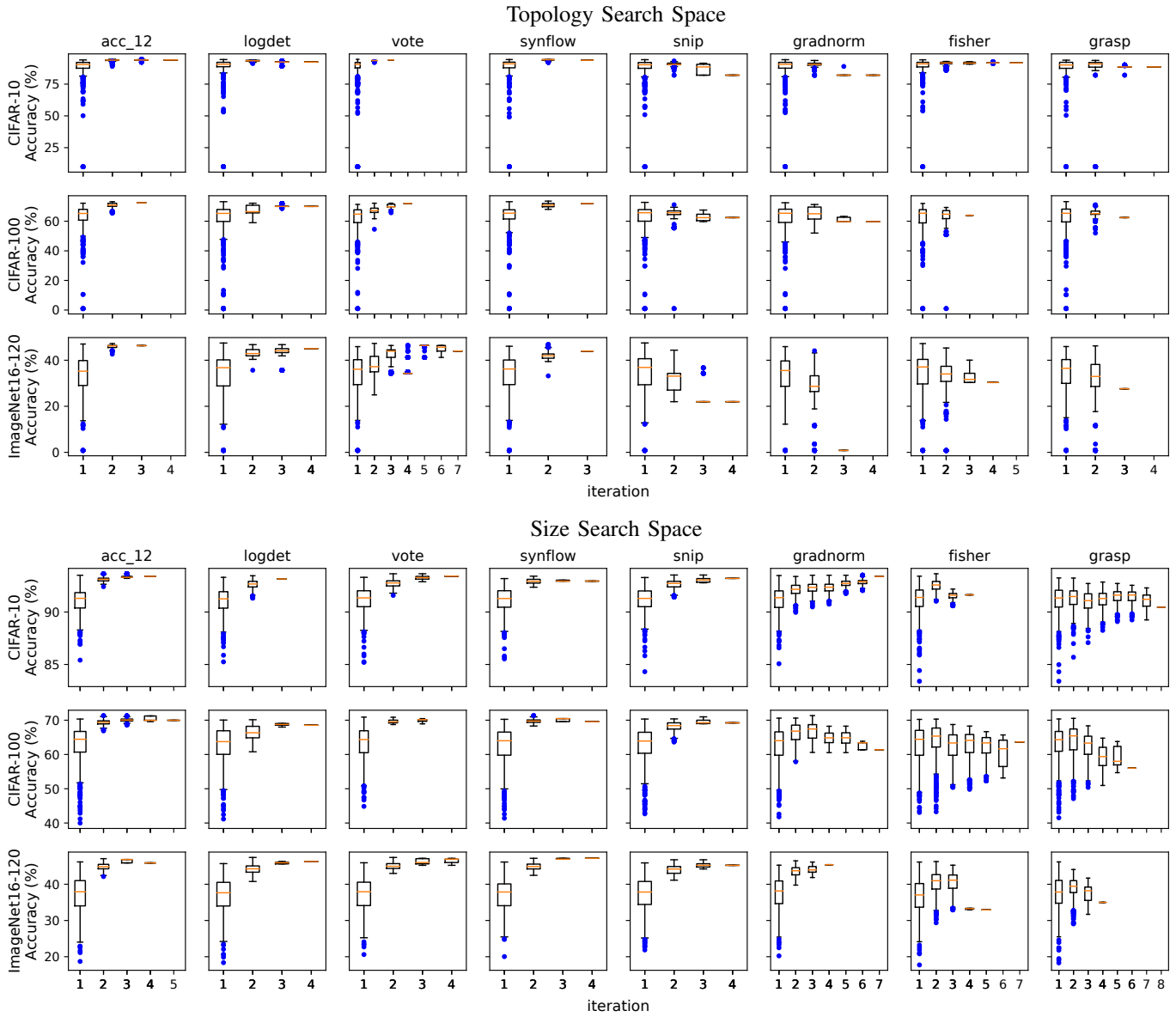

Fig. 4. Evolution of accuracy distribution on NATS-Bench topology and size search spaces.

training. Empirical evaluation on image-classification benchmarks shows that the algorithm consistently discovers architectures whose validation accuracy is close to the true optimum while consuming orders of magnitude less compute. These results demonstrate that a proxy-driven EDA can reliably search large, mixed-type design spaces with negligible computational overhead, paving the way for rapid, energy-efficient NAS.

Future work will explore several promising directions. First, we plan to extend the current framework to a multi-objective setting, jointly minimizing latency, memory footprint, and energy consumption together with predictive performance so that the resulting architectures are not only accurate but also ready for deployment on constrained devices. Second, we will investigate multivariate EDA algorithms that employ dependency-aware models such as Bayesian networks or Markov structures, thereby capturing interactions among hyperparameters that the present univariate assumption may miss. Third, we aim to broaden the applicability of the proxy-driven NAS pipeline by adapting it to generative tasks (including text and image synthesis) and to other vision problems such as object detection and semantic segmentation, where architectural choices critically affect efficiency. Addressing these challenges will further lower the computational barrier to neural architecture discovery and enable the design of high-performing, resource-conscious models across a wide spectrum of AI applications.

REFERENCES

[1] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021.

TABLE I
PROXY-DRIVEN EDA RESULTS ON NATS-BENCH TOPOLOGY SEARCH SPACE.

Dataset	Proxy	Edge $e_{1,2}$	Edge $e_{1,3}$	Edge $e_{2,3}$	Edge $e_{1,4}$	Edge $e_{2,4}$	Edge $e_{3,4}$	Accuracy (%)
CIFAR-10	acc_12	conv 3x3	conv 1x1	conv 3x3	skip conn.	zeroize	conv 3x3	94.0
	vote	conv 3x3	conv 3x3	conv 3x3	conv 1x1	conv 3x3	conv 3x3	94.0
	synflow	conv 3x3	conv 3x3	conv 3x3	conv 3x3	conv 3x3	conv 3x3	93.7
	logdet	conv 1x1	conv 1x1	conv 3x3	conv 1x1	conv 1x1	conv 1x1	92.4
	fisher	conv 1x1	zeroize	conv 3x3	zeroize	conv 1x1	conv 3x3	91.8
	grasp	conv 1x1	zeroize	conv 3x3	zeroize	zeroize	conv 1x1	88.5
	snip	conv 1x1	zeroize	conv 1x1	zeroize	zeroize	conv 1x1	82.0
	gradnorm	conv 1x1	zeroize	conv 1x1	zeroize	zeroize	conv 1x1	82.0
	global optimum	conv 3x3	conv 3x3	conv 3x3	skip conn.	conv 1x1	conv 3x3	94.6
CIFAR-100	acc_12	conv 3x3	conv 3x3	zeroize	skip conn.	conv 3x3	conv 1x1	72.5
	vote	conv 3x3	conv 3x3	conv 3x3	conv 3x3	conv 3x3	conv 3x3	72.0
	synflow	conv 3x3	conv 3x3	conv 3x3	conv 3x3	conv 3x3	conv 3x3	72.0
	logdet	conv 1x1	conv 3x3	conv 3x3	conv 1x1	conv 1x1	conv 1x1	70.0
	fisher	conv 1x1	conv 3x3	conv 1x1	zeroize	zeroize	conv 1x1	64.0
	grasp	conv 1x1	zeroize	conv 3x3	zeroize	zeroize	conv 3x3	62.4
	snip	conv 1x1	zeroize	conv 3x3	zeroize	zeroize	conv 3x3	62.4
	gradnorm	conv 3x3	zeroize	conv 3x3	zeroize	zeroize	conv 3x3	59.7
	global optimum	conv 1x1	conv 3x3	conv 3x3	skip conn.	conv 3x3	conv 1x1	73.7
ImageNet16-120	acc_12	conv 3x3	conv 1x1	conv 3x3	skip conn.	conv 3x3	conv 3x3	46.3
	logdet	conv 1x1	conv 1x1	conv 3x3	conv 1x1	conv 1x1	conv 1x1	45.1
	vote	conv 3x3	conv 3x3	conv 3x3	conv 3x3	conv 1x1	conv 3x3	44.0
	synflow	conv 3x3	conv 3x3	conv 3x3	conv 3x3	conv 1x1	conv 3x3	44.0
	fisher	conv 3x3	zeroize	conv 1x1	zeroize	conv 3x3	conv 3x3	30.3
	grasp	conv 3x3	zeroize	conv 1x1	zeroize	zeroize	conv 1x1	27.6
	snip	conv 1x1	conv 1x1	conv 1x1	zeroize	zeroize	conv 1x1	22.0
	gradnorm	conv 1x1	zeroize	conv 1x1	zeroize	zeroize	conv 1x1	0.8
	global optimum	conv 1x1	conv 3x3	conv 3x3	skip conn.	conv 3x3	conv 1x1	47.7

[2] A. Boumendil, W. Bechkit, P.-E. Portier, F. L. Mouël, and M. Egan, "Grow, prune or select data: which technique allows the most energy-efficient neural network training?" in *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2023, pp. 303–308.

[3] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: a survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[4] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[5] Y. Xue, Y. Wang, J. Liang, and A. Slowik, "A self-adaptive mutation neural architecture search algorithm based on blocks," *IEEE Computational Intelligence Magazine*, vol. 16, no. 3, pp. 67–78, 2021.

[6] M. Javaheripi, M. Samragh, T. Javidi, and F. Koushanfar, "GeneCAI: genetic evolution for acquiring compact AI," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 350–358.

[7] M. Loni, S. Sinaei, A. Zoljodi, M. Daneshtalab, and M. Sjödin, "DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems," *Microprocessors and Microsystems*, vol. 73, p. 102989, 2020.

[8] J. Chen, J. Gao, Y. Chen, B. M. Oloulade, T. Lyu, and Z. Li, "Auto-GNAS: A parallel graph neural architecture search framework," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 3117–3128, 2022.

[9] S. Li, Y. Sun, G. G. Yen, and M. Zhang, "Automatic design of convolutional neural network architectures under resource constraints," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[10] Y. Xue, P. Jiang, F. Neri, and J. Liang, "A multi-objective evolutionary approach based on graph-in-graph for neural architecture search of convolutional neural networks," *International Journal of Neural Systems*, vol. 31, no. 09, p. 2150035, 2021.

[11] A. R. F. da Silva, L. M. Pavelski, L. A. Q. C. Júnior, P. H. D. O. Gomes, L. M. Azevedo, and F. E. F. Junior, "An evolutionary search algorithm for efficient ResNet-based architectures: a case study on gender recognition," in *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2022, pp. 1–10.

[12] T. Zhang, N. Li, H. Kang, J. Liu, H. Wang, and L. Ma, "Neural architecture search based on brain storm optimization algorithm for face detection," in *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024, pp. 1–8.

[13] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[14] Z. Zhao, G.-e. Zhang, M. Jiang, L. Feng, and K. C. Tan, "EDNAS: An efficient neural architecture design based on distribution estimation," in *2020 2nd International Conference on Industrial Artificial Intelligence (IAI)*, 2020, pp. 1–6.

[15] A. Muravev, J. Raitoharju, and M. Gabbouj, "Neural architecture search by estimation of network structure distributions," *IEEE Access*, vol. 9, pp. 15 304–15 319, 2021.

[16] E. Franco-Gaona, M. S. Avila-Garcia, and I. Cruz-Aceves, "Automatic neural architecture search based on an estimation of distribution algorithm for binary classification of image databases." *Mathematics (2227-7390)*, vol. 13, no. 4, 2025.

[17] A. R. F. da Silva, L. M. Pavelski, and L. A. Q. C. Júnior, "Hidden design principles in zero-cost performance predictors for neural architecture search," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–10.

[18] M. Haddad-Neto, A. Silva, R. Mendonca-Neto, and L. Cordovil, "Exploring the impact of zero-cost proxies for hybrid vision transformers," in *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024, pp. 1–8.

[19] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and evolutionary computation*, vol. 1, no. 3, pp. 111–128, 2011.

[20] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[21] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2423–2432.

[22] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4095–4104.

[23] G. Zhong, W. Jiao, W. Gao, and K. Huang, "Automatic design of deep networks with neural blocks," *Cognitive Computation*, vol. 12, no. 1, pp. 1–12, 2020.

[24] N. Mitschke, M. Heizmann, K.-H. Noffz, and R. Wittmann, "Gradient based evolution to optimize the structure of convolutional neural

TABLE II
PROXY-DRIVEN EDA RESULTS ON NATS-BENCH SIZE SEARCH SPACE.

Dataset	Proxy	Channels c_1	Channels r_1	Channels c_2	Channels r_2	Channels c_3	Accuracy (%)
CIFAR-10	vote	64	64	64	56	64	93.4
	acc_12	64	64	64	64	64	93.4
	gradnorm	64	56	64	64	48	93.4
	snip	56	56	64	48	56	93.2
	logdet	64	64	64	56	40	93.2
	synflow	56	56	64	56	64	93.0
	fisher	64	64	48	8	8	91.6
	grasp	56	56	8	24	32	90.5
	global optimum	64	64	56	64	24	93.6
CIFAR-100	vote	64	[48, 64]	[56, 64]	48	64	70.5*
	acc_12	64	64	56	56	64	70.0
	synflow	64	64	64	56	64	69.6
	snip	64	56	64	48	56	69.2
	logdet	64	56	64	40	40	68.5
	fisher	64	40	32	24	24	63.6
	gradnorm	64	56	48	16	16	61.4
	grasp	56	64	48	24	8	56.0
	global optimum	64	64	48	48	64	71.3
ImageNet16-120	synflow	64	64	64	64	64	47.2
	vote	64	[56, 64]	64	[56, 64]	64	46.9*
	logdet	64	64	56	56	56	46.3
	acc_12	64	56	56	64	56	45.9
	gradnorm	64	56	64	40	48	45.4
	snip	64	56	64	56	64	45.2
	grasp	64	56	56	8	8	35.0
	fisher	64	56	32	16	8	32.9
	global optimum	64	64	56	64	56	47.4

* best architecture selected by vote in the final subspace

networks," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 3438–3442.

[25] B. P. Evans, H. Al-Sahaf, B. Xue, and M. Zhang, "Genetic programming and gradient descent: A memetic approach to binary image classification," *arXiv preprint arXiv:1909.13030*, 2019.

[26] S. Yang, Y. Tian, C. He, X. Zhang, K. C. Tan, and Y. Jin, "A gradient-guided evolutionary approach to training deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[27] M. Loni, A. Zoljodi, A. Majd, B. H. Ahn, M. Daneshtalab, M. Sjödin, and H. Esmaeilzadeh, "FastStereoNet: A fast neural architecture search for improving the inference of disparity estimation on resource-limited platforms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2021.

[28] Y. Chen, Z. Guo, Q. Yin, H. Chen, and K. Huang, "Layer-wisely supervised learning for one-shot neural architecture search," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[30] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[32] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI Conference on Artificial Intelligence*, 2017.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[34] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

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

[36] D. Zhou, X. Zhou, W. Zhang, C. C. Loy, S. Yi, X. Zhang, and W. Ouyang, "EcoNAS: Finding proxies for economical neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 396–11 404.

[37] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 7588–7598.

[38] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight NAS," in *International Conference on Learning Representations*, 2021.

[39] P. Keserwani, S. S. Miriyala, V. N. Rajendiran, and P. N. Shivamurthappa, "Receptive field reliant zero-cost proxies for neural architecture search," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[40] Q. Zhou, K. Sheng, X. Zheng, K. Li, X. Sun, Y. Tian, J. Chen, and R. Ji, "Training-free transformer architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 894–10 903.

[41] Y. Qiao, H. Xu, Y. Zhang, and S. Huang, "MONAS: Efficient zero-shot neural architecture search for MCUs," in *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–8.

[42] X. Ning, C. Tang, W. Li, Z. Zhou, S. Liang, H. Yang, and Y. Wang, "Evaluating efficient performance estimators of neural architectures," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 265–12 277, 2021.

[43] C. White, A. Zela, R. Ru, Y. Liu, and F. Hutter, "How powerful are performance predictors in neural architecture search?" *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 454–28 469, 2021.

[44] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 428–10 436.