Robust and Efficient Multi-Fidelity Neural Architecture Search with Zero-Cost Proxy-Guided Local Search

QUAN MINH PHAN and NGOC HOANG LUONG*, University of Information Technology, Viet Nam and Vietnam National University Ho Chi Minh City, Viet Nam

Training-free metrics, also known as zero-cost (ZC) proxies, enable efficient exploration in Neural Architecture Search (NAS) but are less effective than training-based metrics like validation accuracy in identifying high-performance networks. In this article, we investigate the effectiveness of ZC proxies by taking a deeper look into their fitness landscapes utilizing Local Optima Networks. We introduce MF-NAS, a two-stage NAS framework that first employs a ZC proxy-guided local search algorithm to explore the search space. Networks with the highest ZC scores are then fed into the Successive Halving (SH) algorithm to identify the top-performing architecture. However, we observe considerable performance gaps when different ZC metrics are employed. Analyzing those MF-NAS variants with Search Trajectory Networks, we find that high-performance networks could be encountered early or midway through the ZC proxy-guided local search process, making selection based solely on the highest ZC scores ineffective in certain cases. To address this issue, we propose the R-MF-NAS framework, where the selected networks for SH include not only those with the highest ZC scores but also promising solutions encountered during the local search stage. Experiments on diverse NAS benchmarks demonstrate the superiority of both MF-NAS and R-MF-NAS over state-of-the-art methods under a strict budget.

CCS Concepts: • **Computing methodologies** → **Neural networks**; • **Theory of computation** → **Randomized local search**.

Additional Key Words and Phrases: Neural Architecture Search, Zero-Cost Proxies, Local Optimal Network, Search Trajectory Network

ACM Reference Format:
Quan Minh Phan and Ngoc Hoang Luong. 2025. Robust and Efficient Multi-Fidelity Neural Architecture Search with Zero-Cost Proxy-Guided Local Search. *ACM Trans. Evol. Learn. Optim.* 1, 1 (November 2025), 31 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Neural architecture search (NAS) has demonstrated its potential in designing powerful neural networks when network models found by NAS might surpass the manually-designed ones [Chebykin et al. 2022; Tan and Le 2019; Zoph and Le 2017]. In NAS, *performance evaluation strategy*, which estimates the true performance of architectures (e.g., test accuracy), plays a vital role in guiding the search algorithms toward high-quality architectures. Many metrics have been utilized as proxies for test performance and they can be divided into two categories: *training-based* [Ru et al. 2021; Xie and Yuille 2017] and *training-free* [Abdelfattah et al. 2021; Mellor et al. 2021; Tanaka et al. 2020].

*Corresponding author.

Authors' Contact Information: Quan Minh Phan, quanphm@uit.edu.vn; Ngoc Hoang Luong, hoangln@uit.edu.vn, University of Information Technology, Ho Chi Minh City, Viet Nam and Vietnam National University Ho Chi Minh City, Ho Chi Minh City, Viet Nam.

ACM 2688-3007/2025/11-ART
https://doi.org/XXXXXXX.XXXXXXX

One of the commonly-used training-based metrics is validation accuracy [White et al. 2021a; Xie and Yuille 2017; Zoph and Le 2017]. Although the computing cost of training-based metrics are high (due to expensive network training to obtain proper network weights before evaluating on the validation set), these metrics are reliable and highly correlated to the test accuracy. Some techniques have been proposed to deal with the high cost of training-based metrics, such as learning curve extrapolation [Yan et al. 2021] or using *training-free* metrics [Abdelfattah et al. 2021; Mellor et al. 2021; Tanaka et al. 2020]. Training-free metrics (also known as *zero-cost* proxies in the literature) are performance indicators that can be computed with a single pass through the network (i.e., without a typical training procedure of many iterations); the costs for obtaining their values are thus trivial [Krishnakumar et al. 2022; Mellor et al. 2021]. The number of parameters and the floating-point operations (FLOPs) are also considered as the baselines among zero-cost (ZC) proxies [Krishnakumar et al. 2022; Ning et al. 2021; White et al. 2022]. In addition to the number of parameters and FLOPs, many ZC proxies have been proposed and have shown promising Spearman's rank correlations [Spearman 1904] with network accuracy such as Synaptic Flow [Tanaka et al. 2020] or NWOT [Mellor et al. 2021]. However, recent studies indicate that using *solely* ZC proxies as the search objective seems to be ineffective in finding truly good architectures [Luong et al. 2024; White et al. 2022]. Instead, ZC proxies are suggested to be used to accelerate the NAS process, and training-based metrics should still be the main objective to guide search algorithms [Abdelfattah et al. 2021; Krishnakumar et al. 2022; White et al. 2022].

In our preliminary study published in the proceedings of GECCO 2024 [Phan and Luong 2024], we first investigate the capability of ZC proxies to search for high-performance architectures. While previous studies mainly focus on rank correlations between ZC proxies and test performance, we put more attention on the *fitness landscapes* of these training-free metrics. We construct fitness landscapes by utilizing Local Optima Networks (LONs) [Ochoa et al. 2008], which is a visualization tool for the analysis of combinatorial optimization problems [Baioletti et al. 2019; Zou et al. 2022]. The LON analyses highlight the failure of ZC proxies in obtaining high-quality architectures because the majority of the networks that attain optimal ZC proxy scores actually yield sub-optimal test performance [Phan and Luong 2024]. Nevertheless, the insights from LONs demonstrate the suitability of the fitness landscapes of ZC proxies for a typical local search with a simple escape procedure to straightforwardly approach the regions containing promising networks. We then introduce **MF-NAS**, a novel two-stage multi-fidelity NAS framework that makes use of the strengths of both training-free and training-based metrics. The first stage of MF-NAS involves exploring the search space using a local search algorithm guided by a ZC proxy. In the second stage, the top-scored candidate architectures obtained in the first stage are evaluated using a training-based metric in an efficient manner via the Successive Halving procedure [Jamieson and Talwalkar 2016]. The network with the highest validation accuracy is selected as the final NAS result, yielding both the architecture together with its network weights. Empirical results on diverse NAS problems demonstrate the competitiveness of MF-NAS compared to state-of-the-art NAS algorithms in both efficacy and efficiency [Phan and Luong 2024]. However, the robustness of MF-NAS across different ZC metrics has not been thoroughly investigated. This is a crucial aspect because the ZC metrics suitable for solving an arbitrary NAS problem are often unknown a priori in practice. A robust NAS algorithm should exhibit stable performance regardless of the chosen ZC metric.

This article extends the original work [Phan and Luong 2024] with the following contributions:

- We evaluate the robustness of the MF-NAS framework across various ZC metrics. The results highlight the instability issue of MF-NAS due to the choice of ZC metrics, as it performs well with suitable ZC metrics but poorly with unsuitable ones.

- We utilize Search Trajectory Networks (STNs) [Ochoa et al. 2021] to investigate the search behavior of the ZC proxy-guided local search stage in MF-NAS. Our analysis reveals that for certain ZC metrics, the top-scoring networks at the end of the search do not exhibit high performance. Moreover, the local search algorithm often encounters truly high-performance networks at the beginning or middle of the search. Consequently, the selection mechanism in MF-NAS, i.e., choosing the top$-k$ networks with the highest ZC scores for Successive Halving, might be disadvantageous.
- Using insights gained from STN analysis, we introduce **R-MF-NAS**, a more robust variant of MF-NAS. In R-MF-NAS, the selection mechanism for Successive Halving incorporates not only networks with the highest ZC scores but also networks identified as starting and middle nodes in the STN. Extensive experiments demonstrate the superior stability of **R-MF-NAS** compared to **MF-NAS**.

The rest of the article is organized as follows. We provide an overview of the Successive Halving mechanism and summarize related studies on ZC proxies and NAS landscape analysis in Section 2. We visualize and analyze the fitness landscapes of various ZC metrics, highlighting the limitations of ZC metrics in identifying top-performing networks in Section 3. Section 4 presents the MF-NAS framework and evaluates its performance through extensive experiments. Note that the contents of Sections 2, 3, and 4 are also in the conference version published at GECCO 2024 [Phan and Luong 2024]. In Section 5, we analyze the search behavior of the ZC proxy-guided local search in MF-NAS with various ZC metrics using STNs. The obtained insights are utilized to develop R-MF-NAS and its effectiveness is then validated and compared to MF-NAS. We conclude the article in Section 6.

2 Related Work

Zero-cost proxy analysis. Relying solely on ZC metrics could lead the search algorithm away from its primary goal of discovering high-performing architectures [Chen et al. 2021b; Ning et al. 2021]. Certain ZC proxies have been found to introduce systematic biases, often favoring architectures with larger sizes [Ning et al. 2021], narrower structures, or wider channels [Chen et al. 2021b], rather than truly high-performing architectures. Several studies have evaluated the utility of ZC metrics for identifying top candidate architectures by examining their rank correlations with test accuracy [Abdelfattah et al. 2021; Krishnakumar et al. 2022; White et al. 2022]. These findings indicate that training-free metrics should be employed as "low-cost *surrogates*" to speed up the search process, rather than as the primary objective for directing the search.

NAS landscape analysis. Conducting fitness landscape analysis[1] offers an intuitive way to derive valuable insights from NAS search spaces [Ochoa and Veerapen 2022; Phan and Luong 2023; Thomson et al. 2023; White et al. 2021b]. Various NAS loss landscapes were analyzed in White et al. [2021b] and they identified local search as a strong baseline algorithm for NAS. They further argued that the search difficulty could be mitigated by reducing the noise present in the training pipeline. Similar findings were reported by Ochoa and Veerapen [2022], Thomson et al. [2023], and Phan and Luong [2023], where fitness landscapes were visualized using local optima networks (LONs)[2]. Across these studies, local search consistently outperformed other algorithms in NAS. However, Ochoa and Veerapen [2022] and Thomson et al. [2023] restricted their analysis to scenarios where

[1]A fitness landscape is composed of three components: (1) a set of all feasible solutions in the search space S, (2) a neighborhood function $N: S \rightarrow 2^S$ that assigns a set of neighboring solutions to every $s \in S$, and (3) a fitness function f: $S \rightarrow \mathbb{R}$ that maps solutions to their corresponding coordinates in the objective space [Reidys and Stadler 2001].

[2]Local optima network (LON) is defined as a weighted-oriented graph used to structure the fitness landscape. Nodes in an LON represent local optima, and the edge e_{ij} from node i to node j presents that we can move from the optimum i to the optimum j via a pre-defined escape procedure [Ochoa et al. 2008].

validation accuracy after 200 training epochs served as the search objective. To the best of our knowledge, no existing work has examined the landscapes of ZC proxies as objectives or employed LONs to visualize the fitness landscapes induced by these training-free metrics.

Successive Halving. Successive Halving (SH) [Jamieson and Talwalkar 2016] is a multi-fidelity mechanism that is often employed in NAS [Baker et al. 2018; Wang et al. 2021a]. Given a set of candidate architectures, SH begins by evaluating all candidates using a training-based metric at low fidelity, i.e., each architecture is trained for only a few epochs before being assessed using validation accuracy or loss. The bottom half of the candidates, based on these scores, is eliminated, while the top half undergoes higher-fidelity evaluations, where they are trained for more epochs. This procedure is repeated iteratively until either a single candidate remains or the available computational budget is fully expended. The core principle of SH is to progressively assign more computing resources to candidates that show higher potential. Architectures that remain after successive rounds are regarded as the top performers and undergo the highest-fidelity evaluations. In our framework, SH is utilized in the second stage as the selection mechanism to identify the best candidate architecture.

Non-weight-sharing methods. Our proposed methods in this article fall under the category of non-weight-sharing NAS methods [Abdelfattah et al. 2021; Cavagnero et al. 2023; Mellor et al. 2021; Real et al. 2019; Zoph and Le 2017], where each architecture is evaluated independently during the search process (i.e., the weights of one architecture are not shared with others, unlike in weight-sharing methods). Non-weight-sharing NAS methods can be further divided into three sub-categories: training-based [Falkner et al. 2018; Real et al. 2019], training-free [Cavagnero et al. 2023; Mellor et al. 2021], and hybrid [Abdelfattah et al. 2021; Lopes et al. 2022] approaches. In training-based methods, algorithms typically use the validation performance (which is obtained after training candidate networks for a few epochs) as a proxy to estimate their quality. The main advantage of these methods is that the architectures found often perform well on the test set, due to the strong correlation between validation and test performance. However, the major drawback lies in the high computational cost, which considerably limits the number of candidate networks that can be explored in the search space. In contrast, training-free NAS methods evaluate candidate architectures without performing any training by using training-free (zero-cost) metrics. As a result, these methods can explore a large number of architectures with trivial computational cost. For example, NWOT [Mellor et al. 2021] can randomly sample and evaluate 1,000 networks in approximately 300 seconds. FreeREA [Cavagnero et al. 2023] employs three ZC metrics simultaneously as search objectives to guide the REA algorithm in identifying high-performing networks. Experimental results show that FreeREA finds networks comparable to those found by training-based methods, while requiring only 45 seconds of search time on a single NVIDIA GTX 3090 GPU. Nevertheless, recent studies have pointed out the limitations of training-free NAS methods in consistently discovering top-performing architectures across different NAS search spaces [Luong et al. 2024; White et al. 2022]. A training-free metric may be effective in one search space but ineffective in another. Finally, hybrid approaches combine both training-based and training-free metrics during the NAS process. For instance, Abdelfattah et al. [2021] propose randomly sampling a large number of candidate architectures, evaluating their quality using ZC metrics, and then selecting the most promising ones to form the initial population of an evolutionary algorithm. Subsequently, a training-based metric is employed as the search objective to identify the best-performing networks.

The two NAS algorithms we propose in this article are categorized under the hybrid approach. Specifically, we first perform local search using a ZC metric to identify the top$-k$ promising architectures at trivial cost. Then, we apply Successive Halving guided by a training-based metric to return the final top-performing network from the selected top$-k$ candidates.

3 Failure of ZC Proxies in Seeking Top-Performing Architectures

We demonstrate the limitations of ZC proxies in identifying top-performing architectures by analyzing their behavior in the NAS-Bench-201 search space. Specifically, we explore NAS-Bench-201 [Dong et al. 2022; Dong and Yang 2020] using the **best-improvement hill climbing algorithm**, where at each step the algorithm examines the 1-opt neighborhood[3] of the current solution and moves to the neighbor with the highest objective value if it provides an improvement. As the search objective, we consider 13 zero-cost NAS proxies from *NAS-Bench-Suite-Zero* [Krishnakumar et al. 2022], including **EPE-NAS** [Lopes et al. 2021], **Fisher** [Turner et al. 2020], **Grad-norm** [Abdelfattah et al. 2021], **Grasp** [Wang et al. 2020], **L2-norm** [Abdelfattah et al. 2021], **Jacov** [Mellor et al. 2021], **NWOT** [Mellor et al. 2021], **Plain** [Abdelfattah et al. 2021], **Synflow** [Tanaka et al. 2020], **Snip** [Lee et al. 2019], **Zen** [Lin et al. 2021], **FLOPS**, and **Params** (i.e., the number of parameters). For comparison, we additionally experiment evaluate three training-based metrics: validation accuracy, validation loss, and training loss. To reduce computational overhead, efficient NAS methods typically train candidate architectures for only a few epochs during the search process [Zoph et al. 2018]. Following prior work on NAS-Bench-201 [Abdelfattah et al. 2021; Dong et al. 2022; Wang et al. 2021a], we query the benchmark at the 12th epoch for all training-based metric evaluations.

For each metric, we conduct 15,625 independent hill climbing runs, corresponding to using every architecture in the NAS-Bench-201 search space as an initial solution. When the algorithm is unable to find any better solution than an architecture $\boldsymbol{x}$ within its 1-opt neighborhood, $\boldsymbol{x}$ is regarded as a local optimum in the search space. We further check whether there exist transitions between these optima by performing an **escape** procedure. In particular, when hill climbing reaches a local optimum $\boldsymbol{x}$, we randomly select another solution $\boldsymbol{x}'$ from the 2-opt neighborhood of $\boldsymbol{x}$ and relaunch the hill climbing process from this new point. By applying this procedure to all solutions in the 2-opt neighborhood, we can systematically examine whether the global optimum is accessible from arbitrary starting points. We then construct a Monotonic Local Optima Network (MLON) [Ochoa et al. 2017], a specialized variant of LON that connects nodes in a direction from lower to higher objective values, to characterize the exploration dynamics. In the MLON, each node represents a local optimum, and each directed edge corresponds to a transition discovered through the escape procedure. Node size and color indicate the total number of incoming edges and the corresponding metric value, respectively. The width of an edge e_{ij} reflects the number of architectures in the 2-opt neighborhood of node i that can guide the hill climbing process toward node j.

Fig. 1 and Fig. 2a present that the landscapes of **FLOPS** and **Params** are uni-modal with respect to the three datasets, as well as the **Synflow** metric for CIFAR-100 and ImageNet16-120. The corresponding LONs exhibit that when these metrics are used as the search objective, given sufficient runtime, hill climbing consistently converges to the architecture with the largest number of parameters or the highest FLOPs in NAS-Bench-201 without requiring any escape procedure. In contrast, the LONs of the remaining metrics exhibit multi-modal landscapes. Some of these landscapes are relatively sparse, as observed for **NWOT** and **Zen**, whereas others are more densely connected, such as those of **Jacov**, **EPE-NAS**, and **Plain**. As illustrated by the MLONs in Fig. 1, there consistently exist direct or indirect paths linking low-value optima to higher-value ones. This implies that, when NAS is conducted using any of these ZC proxies as the search objective, the optimal architecture (with respect to the chosen metric) within NAS-Bench-201 can always be reached by a local search algorithm augmented with a simple escape mechanism, even though the underlying landscapes are multi-modal.

[3]A k-opt neighborhood of a solution $\boldsymbol{x}$ contains neighboring solutions that can be reached by changing k elements of $\boldsymbol{x}$.

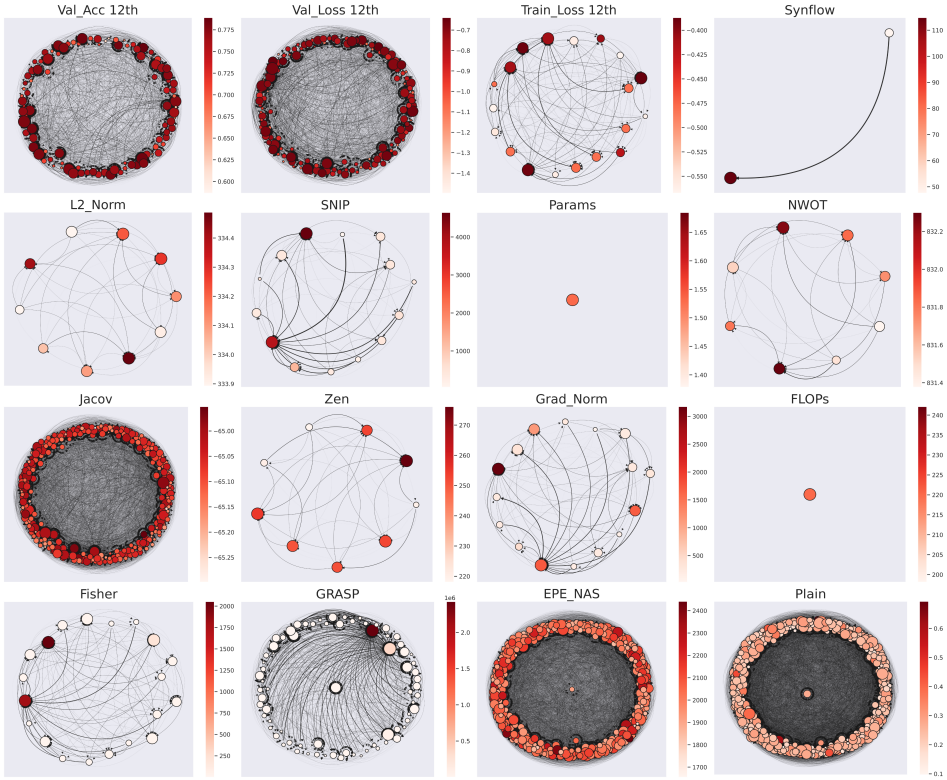
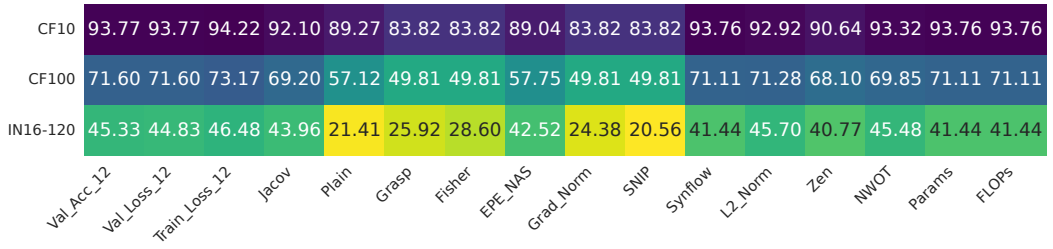


Fig. 1. MLONs of the best-improvement hill climbing algorithm with escape for CIFAR-10 on NAS-Bench-201. The MLONs for CIFAR-100 and ImageNet16-120 are in Appendix A of the Supplementary.

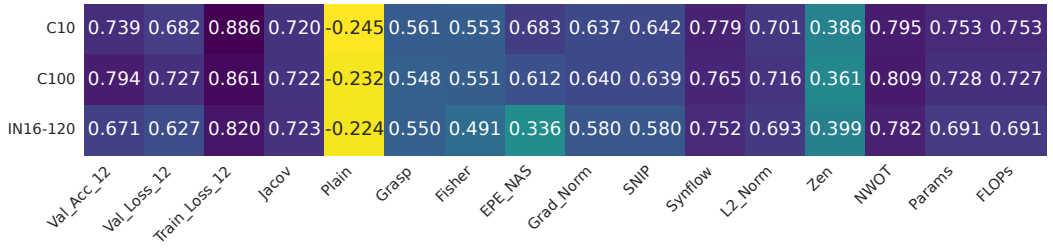
Fig. 2b shows a clear discrepancy in test performance between the optimal solutions, particularly those identified by ZC proxies. The global optima on the training loss landscapes correspond to some of the best-performing architectures in terms of test accuracy for training-based metrics. Combined with the insights from the MLON analysis, this observation provides further evidence supporting the findings of Ru et al. [2021], which showed that training loss is a stronger search objective than other training-based metrics on NAS-Bench-201. In contrast, most architectures located at the global optima of ZC proxy landscapes exhibit inferior test accuracy compared to those identified on training-based metric landscapes. A high rank correlation with test accuracy does not necessarily guarantee the discovery of top-performing architectures. For instance, the **NWOT** metric has a Spearman's correlation of approximately 0.8, which is the highest correlation among ZC proxies, but the optimal solutions found by **NWOT**-guided local search are not the best ones in terms of test performance compared to those of other ZC proxies (see Fig. 2b and Fig. 2c). Some ZC proxies such as **Snip** and **Grad-norm** have a correlation of approximately 0.6 but their optimal solutions are worse than the ones found by **Zen**, which only has a correlation of approximately 0.3. We also found that *none of the solutions at the global optima* on the fitness landscapes of ZC proxies achieve the optimal test accuracy (see Fig. 2b).

We further find that the regions corresponding to the top 1% of architectures ranked by ZC proxy scores contain only a small fraction of the truly top-performing networks (see Fig. 3). In addition, nearly all global optima on ZC proxy landscapes fall outside the top 1% of architectures in terms of

(a) The number of optima on the fitness landscape of each metric.



(b) The test accuracy of the architecture at the global optimum on the fitness landscape of each metric.



(c) The Spearman's rank correlation between the test accuracy values and the scores of each metric.

Fig. 2. The results of exploring the fitness landscapes of different metrics on NAS-Bench-201.

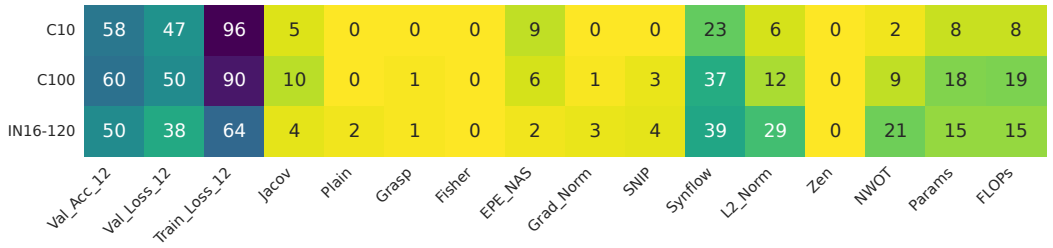
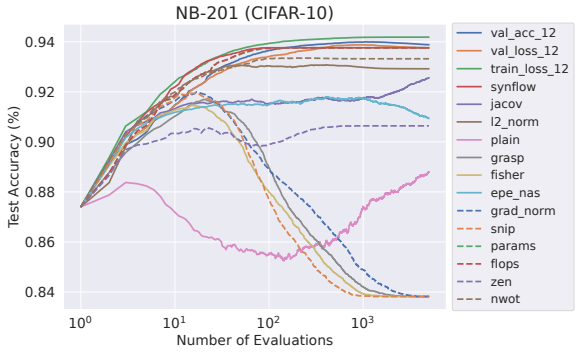
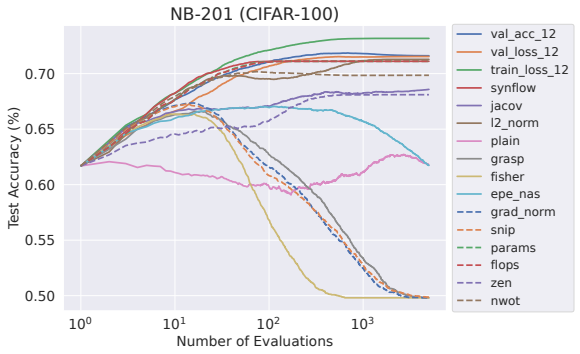


Fig. 3. The number of top-1% most-accurate architectures in NAS-Bench-201 (i.e., 156 best networks) within the top-1% architectures returned by each metric. The lowest accuracies among all solutions in the top 1% most-accurate architectures for the CIFAR-10, CIFAR-100, and ImageNet16-120 datasets are 93.78, 71.13, and 45.40, respectively.

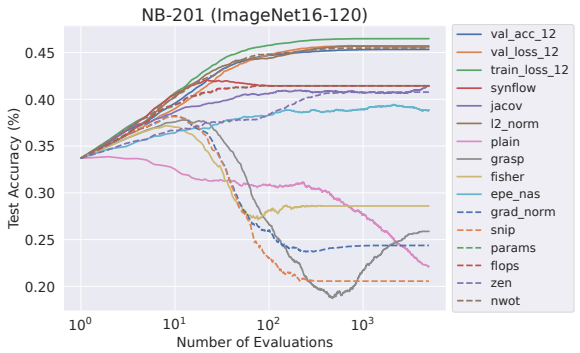
test accuracy. These findings indicate that even if we could approach the regions of top 1% in terms of ZC proxy scores, obtaining the architectures that truly have the highest test accuracies is difficult.



(a) CIFAR-10



(b) CIFAR-100



(c) ImageNet16-120

Fig. 4. The average performance trends of the first-improvement hill climbing algorithm with escape regarding 3 training-based metrics and 13 ZC proxies over 5,000 evaluations on (a) CIFAR-10, (b) CIFAR-100, and (c) ImageNet16-120. We repeat the search process 500 times for each metric.

NAS algorithms using these training-free metrics as the search objective might overlook the truly good architectures and return the poorer ones. To further investigate this phenomenon, we perform additional experiments on NAS-Bench-201 using **first-improvement hill climbing**, in which the algorithm moves to the first better solution encountered within the 1-opt neighborhood at each iteration. Fig. 4 illustrates how different ZC proxies influence the search dynamics of the

hill climbing algorithm with escape. Notably, when **Grasp**, **Grad-norm**, or **Snip** are used as the objective functions, the search trajectories initially approach high-performing architectures but subsequently bypass them and eventually converge to suboptimal solutions. Although training-based metrics require significantly higher computational cost, they prove to be more effective than training-free metrics in guiding the search toward top-performing architectures.

4 Multi-Fidelity Neural Architecture Search (MF-NAS)

4.1 Proposed Framework

We have two observations about the two categories of NAS metrics, i.e., training-based metrics and ZC proxies, via our landscape analysis in Section 3: **(1)** The characteristics of performance metrics' landscapes enable local search-based algorithms to reach the regions that potentially contain high-performance architectures. **(2)** Finding top-performing architectures by using *solely* ZC proxies is hard. Meanwhile, it is more straightforward if we utilize training-based metrics.

Algorithm 1: Multi-Fidelity Local Search Successive Halving (MF-NAS)

Input: Zero-Cost proxy Z, Training-based metric T, Total budget B (in seconds/epochs), Maximum number of evaluations in the first stage M
The number of top architectures for selection to the second stage k

Output: The best architecture found so far $\boldsymbol{x}^*$.

1 $\mathcal{H}, B \leftarrow$ Zero-Cost-Local-Search(Z, M, B) // First stage: Exploring search space using First-improvement Hill Climbing with zero-cost metric Z.

2 $P \leftarrow$ Get-TopK-Solutions($\mathcal{H}$, k)

3 $\boldsymbol{x}^* \leftarrow$ Successive-Halving(P, T, B) // Second stage: Selecting the best architecture $\boldsymbol{x}^*$ from the candidate pool P using Successive Halving with the training-based metric T.

4 **return** $\boldsymbol{x}^*$

We propose utilizing ZC proxies, which offer low computational overhead, to efficiently guide the search algorithm toward regions of the search space that are likely to contain high-performing architectures. Once such regions are identified, training-based metrics (e.g., validation accuracy) are employed to seek for the most accurate architecture within these regions. To achieve this, we introduce the Multi-Fidelity Neural Architecture Search (MF-NAS) framework, which decomposes the search process into two distinct stages (see Algorithm 1 for pseudocode). In the first stage, the search space is explored using the **first-improvement hill climbing algorithm with escape**, where a ZC proxy Z is used as the objective function. The purpose of this stage is to guide the algorithm toward regions containing promising architectures in the search space at *minimal computational cost*. The budget for this stage is set to M architecture evaluations, i.e., computing ZC proxy scores for M candidate architectures. We log the ZC proxy values and configurations of all M architectures found during the local search trajectory in the set $\mathcal{H}$ (line 1) and then select the top-k architectures having the highest scores from $\mathcal{H}$ to form a candidate pool P at the end (line 2). This first stage is similar to the *Zero-Cost Warmup* procedure in Abdelfattah et al. [2021], but we note that they use Random Search to sample M random architectures while we use a local search algorithm instead.

In the second stage (i.e., the high-fidelity stage), the architectures in the candidate pool P are evaluated using a training-based metric T to identify the best-performing architecture. To reduce the computational cost associated with full network training, we employ the Successive Halving (SH) mechanism. At the beginning of the second stage, all architectures in the candidate pool P are evaluated for their training-based metric values after undergoing a few training epochs. The better

half of the candidate pool with the highest training-based metric scores is kept while the worse half is eliminated from P. The selected architectures are then trained for more epochs (i.e., increasingly higher fidelity), and we repeat this evaluation-and-selection process on P till the computing budget is exhausted or there remains only one architecture in P. In case the budget runs out before the pool P is narrowed down to a single candidate, the architecture having the highest training-based metric value so far is considered the resulting solution found by the algorithm.

4.2 Benchmark NAS Problems

The search effectiveness of MF-NAS significantly depends on the quality of candidate networks provided as input to Successive Halving (i.e., the top architectures with the highest ZC scores). If the input set lacks high-performance networks, executing Successive Halving at later stages becomes ineffective. As noted in Abdelfattah et al. [2021] and White et al. [2022], the limitation of ZC metrics is their varying stability across different NAS problems. A ZC metric may perform well on one specific problem but poorly on another. Finding a single ZC proxy that consistently performs well across diverse NAS problems remains a challenge. Besides, determining the most effective ZC metric for each specific NAS problem is impractical.

Table 1. Description of NAS problems.

Problem	Search Space	Dataset	Task - Metric	Search Budget
NB101-CF10	NAS-Bench-101	CIFAR-10	Image classification - Test Accuracy	20,000 seconds
NB201-CF10	NAS-Bench-201	CIFAR-10	Image classification - Test Accuracy	20,000 seconds
NB201-CF100		CIFAR-100		40,000 seconds
NB201-IN16		ImageNet16-120		120,000 seconds
NBASR-TIMIT	NAS-Bench-ASR	TIMIT	Speech recognition - Test PER	300 epochs

In this article, we evaluate the *generalizability* and *robustness* of MF-NAS[1] through experiments conducted on five NAS problems covering three NAS search spaces (i.e., NAS-Bench-101 [Ying et al. 2019], NAS-Bench-201 [Dong and Yang 2020], and NAS-Bench-ASR [Mehrotra et al. 2021]) and four datasets (i.e., CIFAR-10, CIFAR-100, ImageNet16-120 and TIMIT). Details of each NAS problem are provided in Table 1. For problems built on the search spaces NAS-Bench-101 and NAS-Bench-201, the search effectiveness of algorithms is assessed using the test accuracies of their discovered architectures. When comparing between two NAS algorithms, the one that finds the network with higher accuracy is considered better. For the NBASR-TIMIT problem, the architectures discovered by the algorithms are evaluated based on their test phoneme error rates (PER) on the TIMIT dataset. When comparing two algorithms for this problem, the algorithm that identifies networks with lower test PER is deemed better. Furthermore, we constrain the search time (e.g., the GPU time or the total training epochs) for each problem to ensure a fair comparison between MF-NAS and other state-of-the-art NAS algorithms.

4.3 Experiments

[1]Source code of MF-NAS [Phan and Luong 2024] is available at: https://github.com/ELO-Lab/MF-NAS

Table 2. List of training-based and training-free NAS metrics that we deploy on MF-NAS for each problem.

Problem	Training-based metrics	Training-free metrics
NB101-CF10	Validation error rate	**(1)** Synflow **(2)** Grad-norm **(3)** #Parameters **(4)** Snip **(5)** Grasp **(6)** Jacov **(7)** FLOPs
NB201-CF10 NB201-CF100 NB201-IN16	Validation error rate	**(1)** EPE-NAS **(2)** Fisher **(3)** Grad-norm **(4)** FLOPs **(5)** Grasp **(6)** L2-norm **(7)** NWOT **(8)** #Parameters **(9)** Jacov **(10)** Plain **(11)** Synflow **(12)** Snip **(13)** Zen
NBASR-TIMIT	Validation PER	**(1)** Jacov **(2)** Snip **(3)** Grad-norm **(4)** #Parameters **(5)** Fisher **(6)** L2-norm **(7)** Plain **(8)** Synflow **(9)** FLOPs

4.3.1 Results of MF-NAS with Various ZC metrics. To evaluate the general performance of the MF-NAS framework with different ZC metrics, we experiment with 7, 13, and 9 variants of MF-NAS on problems built on NAS-Bench-101, NAS-Bench-201, and NAS-Bench-ASR, respectively. Each variant is run 500 times independently as in Ying et al. [2019], Mellor et al. [2021], Dong et al. [2022]. For the local search stage, the maximum number of evaluations M for all MF-NAS variants is set to 2,000 for each problem (the results of experiments with other values of M are presented in Appendix C of the Supplementary). For the Successive Halving stage, the number of input architectures k is set to 32 for the three problems in the NAS-Bench-201 search space and 16 for the two remaining problems, NB101-CF10 and NBASR-TIMIT. We note that the values of training-based metrics in our experiments are queried from the NAS benchmarks, while the values of training-free metrics are queried from the logged results published by NAS-Bench-Suite-Zero [Krishnakumar et al. 2022] and Zero-Cost-NAS [Abdelfattah et al. 2021]. The training-based and training-free metrics used in each problem are detailed in Table 2. Note that the difference in the number of training-free metrics used for each problem is due to the NAS benchmarks we utilize to query the values of ZC metrics. For instance, NAS-Bench-Suite-Zero provides values for 13 ZC metrics for all candidate networks within the NAS-Bench-201 search space but does not include the scores of all architectures in the NAS-Bench-101 search space. In contrast, Zero-Cost-NAS supports fewer ZC metrics than NAS-Bench-Suite-Zero but provides scores for all networks within NAS-Bench-101 and NAS-Bench-ASR. Additionally, the training-based metrics we experiment with are inspired by previous works [Dong and Yang 2020; Mehrotra et al. 2021; Ying et al. 2019]. For the three problems in the NAS-Bench-201 search space (i.e., NB201-CF10/CF100/IN16), we further perform MF-NAS using training loss as the training-based metric, as discussed in Section 4.3.3.

Table 3 summarizes the results of MF-NAS variants across 500 runs. These results demonstrate the effectiveness of our proposed framework, as the performance of the architectures discovered by the best MF-NAS variants is nearly optimal for all problems. Although the MF-NAS framework is designed based on the analysis of the NAS-Bench-201 search space, its impressive performance on other search spaces (e.g., NAS-Bench-101 and NAS-Bench-ASR) highlights the high generalizability of the framework. However, the results also reveal the weak robustness of MF-NAS when using different ZC metrics. Table 3 indicates a considerable performance gap between the best and worst MF-NAS variants. For example, on the NB201-CF100 problem, the performance difference between MF-NAS (**Zen**) and MF-NAS (**Synflow**) is 12.79%. While metrics such as **Synflow**, **FLOPs**, and **Params** consistently yield high effectiveness across most problems, MF-NAS performs poorly when other ZC metrics, such as **Grad-norm**, **Zen**, and **Snip**, are employed. Moreover, no single ZC metric consistently outperforms others across all problems. For instance, the ZC metric **L2-norm** is

Table 3. Average test performance (Mean ± SD) of best architectures found by MF-NAS variants on NAS problems over 500 runs. For problems having symbol ↑, the algorithms achieving **higher** scores are better. For problems having symbol ↓, the algorithms achieving **lower** scores are better. The best and worst results are exhibited in **red** and **blue**, respectively. Symbol '-' means that those ZC proxies are not supported for the considering problems. The results are exhibited in the following three blocks. The first block presents the results of MF-NAS with various ZC proxies for each problem. The second block presents the average results (Mean ± SD) of MF-NAS with various ZC proxies and the difference in the performance between the best and the worst variants. The last block presents the optimal test performance for each problem, which is queried from the NAS benchmark. According to the Student's t-test, the algorithms that are significantly worse than the best ones at a significance level of 0.01 with Bonferroni correction are presented in gray cells.

ZC Metrics	NB101-CF10 ↑	NB201-CF10 ↑	NB201-CF100 ↑	NB201-IN16 ↑	NBASR-TIMIT ↓
Snip	84.67 ± 1.58	93.13 ± 0.00	69.25 ± 0.42	**35.96 ± 0.31**	23.17 ± 0.83
Grasp	93.15 ± 0.33	93.06 ± 0.18	69.46 ± 0.21	42.82 ± 1.43	-
Fisher	85.51 ± 2.72	92.21 ± 0.67	65.60 ± 0.83	35.97 ± 0.06	23.43 ± 0.76
Plain	-	92.35 ± 0.76	69.02 ± 1.17	42.82 ± 2.49	22.96 ± 1.15
Grad-norm	**84.01 ± 1.50**	93.13 ± 0.06	68.82 ± 0.81	35.97 ± 0.00	23.19 ± 0.81
EPE-NAS	-	93.73 ± 0.40	71.44 ± 1.17	44.28 ± 1.22	-
Jacov	92.20 ± 1.51	93.61 ± 0.44	71.81 ± 0.55	44.47 ± 0.98	**83.14 ± 7.99**
L2-norm	-	93.45 ± 0.09	71.28 ± 0.02	**46.48 ± 0.21**	23.44 ± 0.72
Zen	-	**89.42 ± 1.04**	**60.72 ± 0.75**	40.77 ± 0.00	-
NWOT	-	93.58 ± 0.01	71.51 ± 0.16	45.33 ± 0.09	-
Synflow	93.82 ± 0.56	**94.36 ± 0.00**	**73.51 ± 0.05**	46.34 ± 0.00	**21.77 ± 0.00**
FLOPs	93.88 ± 0.25	**94.36 ± 0.00**	**73.51 ± 0.00**	46.34 ± 0.00	21.80 ± 0.37
Params	**93.89 ± 0.25**	**94.36 ± 0.00**	**73.51 ± 0.00**	46.34 ± 0.00	21.81 ± 0.26
Mean ± SD	90.18 ± 4.20	93.13 ± 1.26	69.96 ± 3.45	42.61 ± 3.98	29.41 ± 19.01
Gap (Best, Worst)	9.88	4.94	12.79	10.52	61.37
Optimal	*94.32*	*94.37*	*73.51*	*47.31*	*21.40*

the most effective for the NB201-IN16 problem but it is less effective for other problems. The results in Table 3 reveal that the average accuracies of MF-NAS (Synflow), MF-NAS (Params), and MF-NAS (FLOPs) are identical across all datasets in the NAS-Bench-201 search space. This phenomenon can be explained by the characteristics of the fitness landscapes associated with these metrics in the NAS-Bench-201 search space. As visualized in Fig. 1, the corresponding LONs for these metrics exhibit only one or two optima. Moreover, we find that the network with the highest Synflow score, the network with the highest Params score, and the network with the highest FLOPs score correspond to the same architecture. When there is a single path to the global optimum and the global optima for all three metrics coincide, the resulting search trajectories are likely to be similar, leading to the same average accuracy for all three MF-NAS variants.

In addition to conducting experiments with the default number of zero-cost evaluations set to M = 2,000, we further evaluate MF-NAS using different values of M in Appendix C of the Supplementary. The results indicate that although performance varies slightly when M is increased (e.g., M = 3,000 or 5,000) or decreased (e.g., M = 1,000), the differences are negligible, even in large-scale search spaces such as NAS-Bench-101 (which contains approximately 423,000 unique networks). Therefore, when faced with a new NAS problem involving a large search space, we recommend using MF-NAS with M = 2,000, and optionally increasing M to assess whether performance improves further. We

note that since all evaluations in the local search stage are training-free, increasing M incurs a negligible computational cost.

Among the NAS problems we evaluated, our proposed method MF-NAS successfully identified the optimal network only for the NB201-CF100 task. MF-NAS returned this network in all 500 independent runs. Although the optimal architectures are not found for the remaining tasks, we note that no other NAS method is better than MF-NAS under the same computational budget (which is discussed later in Section 4.3.2).

4.3.2 Comparisons to Other NAS Algorithms. Among the experimental MF-NAS variants, we select three variants that sustain high effectiveness across all problems, i.e., MF-NAS (**Synflow**), MF-NAS (**FLOPs**), and MF-NAS (**Params**), for comparison with other NAS methods. We re-implement several baseline NAS algorithms, including Random Search (RS), Local Search (LS), Successive Halving (SH), Regularized Evolution Algorithm (REA) [Real et al. 2019], REA using the *Warmup* method (REA+W) [Abdelfattah et al. 2021], NASWOT [Mellor et al. 2021], and FreeREA [Cavagnero et al. 2023], as their implementations are straightforward to reproduce. Hyperparameters for all algorithms are detailed in Appendix B of the Supplementary. Additionally, we compare MF-NAS to state-of-the-art methods that report results on similar NAS problems. The results of methods that we do not reproduce are taken directly from the relevant literature [Chen et al. 2021a; Dong et al. 2022; Lopes et al. 2021, 2022; Mellor et al. 2021; Peng et al. 2022; Wang et al. 2021a; Xiao et al. 2022; Yan et al. 2020; Yang et al. 2023; Yu et al. 2020].

We first compare MF-NAS to other NAS algorithms on three problems within the NAS-Bench-201 search space. As shown in Table 4, MF-NAS demonstrates superiority over other *non-weight-sharing* approaches (the results are checked with Student's t-tests at a significance level of 0.01 with Bonferroni correction). Among non-weigh-sharing methods, MF-NAS variants yield the best performance across all three datasets in terms of the test accuracies of the discovered architectures. The results also highlight that MF-NAS is significantly more effective than standalone local search or Successive Halving. Furthermore, MF-NAS outperforms REA+W [Abdelfattah et al. 2021], which is an NAS algorithm using simultaneously a ZC proxy and the validation accuracy metric.

When compared to *weight-sharing* NAS methods, Table 4 shows that MF-NAS variants are only slightly worse than Shapley-NAS [Xiao et al. 2022] on CIFAR-10 (94.36% compared to 94.37%) and ImageNet16-120 (46.34% compared to 46.85%). However, the main drawback of weight-sharing methods is that the resulting architectures extracted from the supernet often exhibit suboptimal test performance when directly employed with the weights obtained during the supernet training [Wang et al. 2021b]. Most weight-sharing NAS methods therefore require further training the resulting networks for a certain amount of epochs or training them from scratch to achieve optimal weights [Chen et al. 2021c; Xiao et al. 2022]. Moreover, the overhead costs of weight-sharing methods are often much higher than non-weight-sharing methods like MF-NAS, as they involve pretraining a supernet and maintaining it during the search.

For the two remaining problems NB101-CF10 and NBASR-TIMIT, Table 5 exhibits that no approach achieved comparable performance to MF-NAS variants under the same computational budgets (i.e., 20,000 seconds for NB101-CF10 and 300 training epochs for NBASR-TIMIT). Comparing MF-NAS to state-of-the-art *non-weight-sharing* and *weight-sharing* NAS methods on NAS-Bench-101 search space, the results demonstrate that all MF-NAS variants significantly outperform weight-sharing methods and perform better than nearly all non-weight-sharing methods (except for RankNOSH [Wang et al. 2021a]). However, it is worth noting that RankNOSH incurs a considerably higher search cost, requiring 8,400 training epochs, which is over 20 times greater than MF-NAS (i.e., 368 epochs). We further re-run MF-NAS with adjusted checkpoints for evaluating architectures

Table 4. Test accuracy performance comparisons with other *non-weight-sharing* methods on three problems within NAS-Bench-201. The results of Random Search, Local Search, Successive Halving (SH), REA, REA+W, NASWOT, and FreeREA are reproduced by ourselves. The results of other non-weight-sharing methods are taken from Yan et al. [2020] (for arch2vec-RL and arch2vec-BO), [Mellor et al. 2021] (for REINFORCE), Chen et al. [2021a] (for TE-NAS), Lopes et al. [2021] (for EPE-NAS), Lopes et al. [2022] (for G-EA), Yang et al. [2023] (for ST-NAS), Dong et al. [2022] (for BOHB), Peng et al. [2022] (for PRE-NAS). The results of weight-sharing methods are taken from Dong et al. [2022] (for ENAS, RSPS, DARTS (1st), DARTS (2nd), GDAS, SETN, and PC-DARTS) and Xiao et al. [2022] (for DrNAS and ShapleyNAS). For non-weight-sharing methods, the first and second blocks represent the algorithms using training-based metrics and training-free metrics as search objective, respectively. The best results are presented in **red**. Best algorithms are statistically significantly better than the others (tested using Student's t-tests at a significance level of 0.01 with Bonferroni correction). The information not reported in the original study is presented as '-'.

Algorithm	**NB201-CF10**		**NB201-CF100**		**NB201-IN16**	
	Search Cost (seconds/epochs)	Accuracy ↑ (%)	Search Cost (seconds/epochs)	Accuracy ↑ (%)	Search Cost (seconds/epochs)	Accuracy ↑ (%)
Weight-sharing						
ENAS [Pham et al. 2018]	-	93.76 ± 0.00	-	70.67 ± 0.62	-	41.44 ± 0.00
RSPS [Li and Talwalkar 2019]	-	91.05 ± 0.66	-	68.26 ± 0.96	-	40.69 ± 0.36
DARTS (1st) [Liu et al. 2019]	-	59.84 ± 7.84	-	61.26 ± 4.43	-	37.88 ± 2.91
DARTS (2nd) [Liu et al. 2019]	-	65.38 ± 7.84	-	60.49 ± 4.95	-	36.79 ± 7.59
GDAS [Dong and Yang 2019b]	-	93.23 ± 0.58	-	68.17 ± 2.50	-	39.40 ± 0.00
SETN [Dong and Yang 2019a]	-	92.72 ± 0.73	-	69.36 ± 1.72	-	39.51 ± 0.33
PC-DARTS [Xu et al. 2020]	-	93.41 ± 0.30	-	67.48 ± 0.89	-	41.31 ± 0.22
DrNAS [Chen et al. 2021c]	-	94.36 ± 0.00	-	**73.51 ± 0.00**	-	46.34 ± 0.00
ShapleyNAS [Xiao et al. 2022]	-	**94.37 ± 0.00**	-	**73.51 ± 0.00**	-	**46.85 ± 0.12**
Non-weight-sharing						
Random Search‡	20,000	93.73 ± 0.36	40,000	71.62 ± 0.91	120,000	45.30 ± 1.00
Local Search‡	20,000	94.03 ± 0.38	40,000	72.35 ± 0.93	120,000	45.94 ± 0.73
SH‡ [Jamieson and Talwalkar 2016]	20,000	93.22 ± 0.66	40,000	70.07 ± 1.41	120,000	43.59 ± 1.80
REA‡ [Real et al. 2019]	20,000	93.85 ± 0.40	40,000	72.32 ± 0.81	120,000	45.72 ± 0.79
REA+W‡ [Abdelfattah et al. 2021]	20,000	94.21 ± 0.20	40,000	72.75 ± 0.52	120,000	46.21 ± 0.52
REINFORCE [Williams 1992]	20,000	93.85 ± 0.40	40,000	72.32 ± 0.81	120,000	45.72 ± 0.79
BOHB [Falkner et al. 2018]	20,000	93.94 ± 0.28	40,000	72.00 ± 0.86	120,000	45.70 ± 0.86
arch2vec-RL [Yan et al. 2020]	12,000	94.12 ± 0.42	-	73.15 ± 0.78	-	46.16 ± 0.38
arch2vec-BO [Yan et al. 2020]	12,000	94.18 ± 0.24	-	73.37 ± 0.30	-	46.27 ± 0.37
G-EA [Lopes et al. 2022]	26,911	93.99 ± 0.23	-	72.36 ± 0.66	-	46.04 ± 0.67
PRE-NAS [Peng et al. 2022]	20,000	94.04 ± 0.34	40,000	72.02 ± 1.22	120,000	45.34 ± 1.03
EPE-NAS [Lopes et al. 2021]	207	91.31 ± 1.69	207	69.58 ± 0.83	207	41.84 ± 2.06
TE-NAS [Chen et al. 2021a]	1,558	93.90 ± 0.47	-	71.24 ± 0.56	-	42.38 ± 0.46
ST-NAS [Yang et al. 2023]	437	93.46 ± 0.59	-	70.58 ± 0.82	-	43.74 ± 1.48
NASWOT‡ [Mellor et al. 2021]	20,000	93.37 ± 0.11	40,000	69.84 ± 0.04	120,000	45.48 ± 0.00
FreeREA‡ [Cavagnero et al. 2023]	20,000	94.36 ± 0.00	40,000	**73.51 ± 0.00**	120,000	46.34 ± 0.00
MF-NAS (Synflow)	18,632 (= 1,192 epochs)	94.36 ± 0.00	35,118 (= 1,192 epochs)	**73.51 ± 0.05**	106,362 (= 1,192 epochs)	46.34 ± 0.00
MF-NAS (FLOPs)	19,391 (= 1,192 epochs)	94.36 ± 0.00	38,346 (= 1,192 epochs)	**73.51 ± 0.00**	115,218 (= 1,192 epochs)	46.34 ± 0.00
MF-NAS (Params)	19,383 (= 1,192 epochs)	94.36 ± 0.00	38,335 (= 1,192 epochs)	**73.51 ± 0.00**	115,238 (= 1,192 epochs)	46.34 ± 0.00
Optimal	-	*94.37*	-	*73.51*	-	*47.31*

‡ reproduced by ourselves

during the Successive Halving procedure (denoted as MF-NAS† in Table 5). Remarkably, MF-NAS† surpasses RankNOSH with a search cost that is 7 times less.

Table 5. Test performance comparisons with other methods on two problems NB101-CF10 and NBASR-TIMIT. We reproduced the results of Random Search, Local Search, Successive Halving (SH), REA, REA+W, NASWOT, and FreeREA. The results of the remaining methods are taken from Yu et al. [2020] (for ENAS, NAO, FBNet, DARTS, SPOS, and FairNAS) and Wang et al. [2021a] (for RankNOSH). The first and second blocks represent the weight-sharing algorithms and non weight-sharing algorithms, respectively. The best results are presented in **red**. Symbol '-' means that the original study does not provide that information. For the NB101-CF10 problem, MF-NAS variants evaluate candidate networks at four different epochs {4, 12, 36, 108}, and the MF-NAS† variant evaluates candidate networks at two different epochs {36, 108}. Best algorithms are statistically significantly better than the others (tested using Student's t-tests at a significance level of 0.01 with Bonferroni correction). We do not perform a statistical comparison with RankNOSH [Wang et al. 2021a] because its report does not provide standard deviation values.

Algorithm	**NB101-CF10**		**NBASR-TIMIT**	
	Search Cost (seconds/epochs)	Accuracy ↑ (%)	Search Cost (epochs)	PER ↓ (%)
Weight-sharing				
ENAS [Pham et al. 2018]	-	91.83 ± 0.42	-	-
NAO [Luo et al. 2018]	-	92.59 ± 0.59	-	-
FBNet [Wu et al. 2019]	-	92.29 ± 1.25	-	-
DARTS [Liu et al. 2019]	-	92.21 ± 0.61	-	-
SPOS [Guo et al. 2020]	-	89.85 ± 3.80	-	-
FairNAS [Chu et al. 2021]	-	91.10 ± 1.84	-	-
Non-weight-sharing				
Random search‡	20,000	93.16 ± 0.26	300	22.15 ± 0.45
Local search‡	20,000	93.16 ± 0.56	300	22.51 ± 2.85
SH‡ [Jamieson and Talwalkar 2016]	20,000	93.19 ± 0.46	300	22.29 ± 0.60
REA‡ [Real et al. 2019]	20,000	93.24 ± 0.27	300	22.32 ± 0.71
REA+W‡ [Abdelfattah et al. 2021]	20,000	93.22 ± 0.28	300	22.02 ± 0.25
RankNOSH [Wang et al. 2021a]	8,400 epochs	93.97	-	-
NASWOT‡ [Mellor et al. 2021]	20,000	92.76 ± 2.57	-	-
FreeREA‡ [Cavagnero et al. 2023]	20,000	93.75 ± 0.33	-	-
MF-NAS (Synflow)	14,742 (= 368 epochs)	93.82 ± 0.56	300	**21.77 ± 0.00**
MF-NAS (FLOPs)	13,487 (= 368 epochs)	93.88 ± 0.25	300	21.80 ± 0.37
MF-NAS (Params)	13,517 (= 368 epochs)	93.89 ± 0.25	300	21.81 ± 0.26
MF-NAS† (Params)	42,924 (= 1,152 epochs)	**94.07 ± 0.20**	-	-
Optimal	-	*94.32*	-	*21.40*

‡ reproduced by ourselves

Comparing with training-free NAS methods such as NASWOT and FreeREA under the same computational budget, MF-NAS consistently outperforms NASWOT across all problems. In contrast, MF-NAS and FreeREA achieve comparable performance on all datasets within the small-scale NAS-Bench-201 search space. However, MF-NAS significantly outperforms FreeREA in the larger search space NAS-Bench-101 (see Table 5). This suggests that architectures discovered solely based on training-free metrics may suffer from sub-optimal performance, and that utilizing both training-free and training-based metrics in a complementary manner can lead to better architectures.

4.3.3 *Ablation Study.*

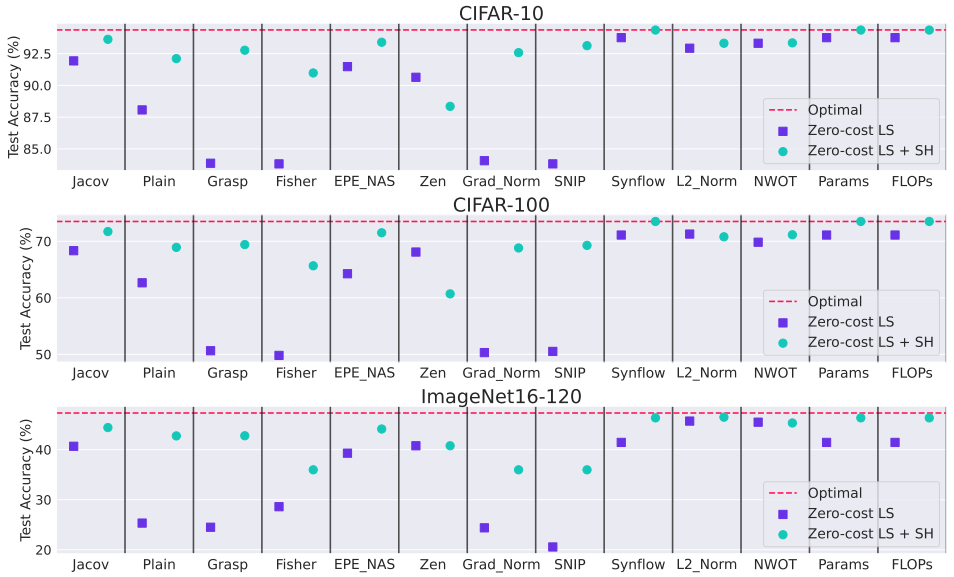


Fig. 5. The average test accuracy performance of the best architectures returned at the end of the 1st stage (LS) and the 2nd stage (SH) of MF-NAS over 500 trials on NAS-Bench-201.

Effectiveness of Successive Halving in selecting top-performing architectures. Fig. 5 exhibits the average test accuracy of the best architectures found at the two stages of MF-NAS. The performance gaps between the architectures having the best ZC proxy scores in the local search stage (i.e., the square markers) and the optimal test accuracy (i.e., the dashed lines) are substantial. However, the gaps are clearly reduced after the Successive Halving stage (i.e., the gap between the circle markers and the dashed lines). We emphasize the contribution of Successive Halving (SH) to the final performance of MF-NAS, particularly when the resulting networks from MF-NAS outperform those found by zero-cost local search (ZC-LS) across most ZC metrics. In cases where the networks returned by ZC-LS are already near-optimal (e.g., Synflow, Params, or FLOPs), executing SH still yields better-performing networks. The contribution of SH to the overall performance of MF-NAS is more clearly demonstrated by examining improvements under ineffective ZC metrics such as Grasp and Grad-Norm. Given that no single ZC metric is consistently effective across all NAS benchmarks, the ability of SH to select out better architectures is noteworthy. Furthermore, the computational cost of SH is reasonable in real-world NAS settings. For example, the SH stage on NAS-Bench-201 requires the equivalent of training six networks from scratch (i.e., each network in NAS-Bench-201 is trained for 200 epochs, and one SH run consumes 1,148 epochs in total under the current settings).

We consider the effectiveness of Successive Halving (SH) in selecting truly top-performing architectures by comparing the test accuracy between the architectures chosen by SH and the true best architectures from the input candidate pool (the results are reported in Table 6). The majority of the architectures selected by SH are in the top three most-accurate architectures out of 32 input candidates. However, there are instances where SH returns poor candidates, such as in the case of the **Zen** metric on CIFAR-10. We then experiment with training loss as the training-based metric T for the second stage SH of MF-NAS, and this change leads to an improvement in the quality of

Table 6. The average test accuracy (·/) of the architectures chosen by Successive Halving (SH) and the true best architectures in the top-32 candidate architectures over 500 runs of MF-NAS on NAS-Bench-201. For the results of SH, we also report the average rank (/·) of the selected architecture out of 32 candidates (rank 1 is the best).

ZC Metric	NB201-CF10		NB201-CF100		NB201-IN16	
	Best in top-32	Returned by SH	Best in top-32	Returned by SH	Best in top-32	Returned by SH
Jacov	93.82	93.61 / 2.2	71.91	71.81 / 1.9	45.33	44.47 / 4.3
Plain	92.49	92.35 / 2.3	69.14	69.02 / 1.5	43.40	42.82 / 1.9
Grasp	93.11	93.06 / 2.4	69.47	69.46 / 1.2	42.87	42.82 / 1.2
Fisher	92.35	92.21 / 15.3	66.72	65.60 / 6.0	38.83	35.97 / 2.9
EPE-NAS	93.89	93.73 / 5.7	71.76	71.44 / 2.1	44.94	44.28 / 3.1
Grad-norm	93.13	93.13 / 1.0	68.85	68.82 / 1.1	38.83	35.97 / 3.1
Snip	93.13	93.13 / 1.0	69.33	69.25 / 1.1	38.83	35.96 / 3.5
Synflow	94.37	94.36 / 2.0	73.51	73.51 / 1.0	46.48	46.34 / 2.6
L2-norm	93.76	93.45 / 14.8	71.59	71.28 / 10.4	46.53	46.48 / 1.3
Zen	90.74	89.42 / 10.8	68.55	60.72 / 21.7	40.77	40.77 / 1.0
NWOT	93.89	93.58 / 14.1	71.74	71.51 / 5.2	46.55	45.33 / 13.0
Params	94.36	94.36 / 1.0	73.51	73.51 / 1.0	46.34	46.34 / 1.0
FLOPs	94.36	94.36 / 1.0	73.51	73.51 / 1.0	46.34	46.34 / 1.0

architecture selections for most cases (see Table 7). Training loss indeed exhibits a higher rank correlation with test accuracy compared to validation accuracy (see Fig. 2c).

Effectiveness of local search in exploring the search space. We assess the impact of the first-improvement local search on the overall performance of the MF-NAS framework by comparing with the MF-NAS variants that use random search in the first stage. Using random search with ZC proxies to sample the search space is exactly the *Zero-Cost Warmup* method in Abdelfattah et al. [2021].

Table 8 exhibits that random search (RS) brings better results than local search (LS) in the case of **Plain**, **Fisher**, and **Zen** while local search is better for **Synflow**, **Params**, and **FLOPs**. The differences are because the top regions of the latter metrics consist of a great number of high-performance architectures while the former metrics do not have such networks in their top regions (see Fig. 3). LS is demonstrated as effective in approaching the top regions of ZC proxies but the non-existence of truly top-performing architectures in these regions makes its exploration less useful than RS, which randomly samples from the search space and possibly figures out more promising solutions.

Performance of MF-NAS with different k values. We evaluate the performance of MF-NAS with varying k values. Increasing the value of k naturally raises the search cost because we need to train more network architectures during the Successive Halving stage. We assess the impact of the hyperparameter k on both efficiency and efficacy of MF-NAS.

Fig. 6 illustrates the performance and the search cost of the three least effective MF-NAS variants (as identified in Table 3) across different k values for three problems built on NAS-Bench-201 (i.e., NB201-CF10, NB201-CF100, and NB201-IN16). Additional results are provided in Appendix D of the

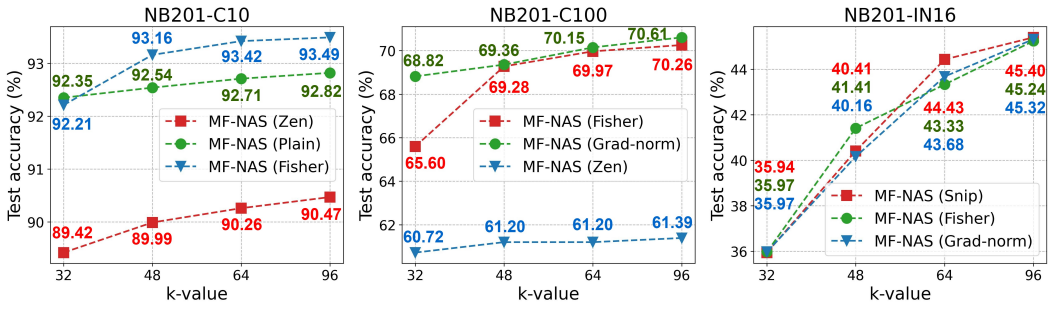
Table 7. Performance of MF-NAS variants using the validation accuracy and training loss in the SH procedure (both metrics are queried at the 12th epoch). The enhanced performance obtained by replacing validation accuracy with training loss is shown in underline, and statistically significant improvements (verified using Student's t-test at a significance level of 0.01 with Bonferroni correction) are highlighted in **red**.

	NB201-CF10		**NB201-CF100**		**NB201-IN16**	
ZC Metric	Val Acc	Train Loss	Val Acc	Train Loss	Val Acc	Train Loss
Jacov	93.61 ± 0.44	**93.79 ± 0.23**	71.81 ± 0.55	71.79 ± 0.50	44.47 ± 0.98	**44.84 ± 0.74**
Plain	92.35 ± 0.76	**92.48 ± 0.60**	69.02 ± 1.17	**71.90 ± 0.44**	42.82 ± 2.49	43.13 ± 2.02
Grasp	93.06 ± 0.17	**93.09 ± 0.15**	69.46 ± 0.21	69.48 ± 0.18	42.82 ± 1.43	**42.92 ± 1.12**
Fisher	92.21 ± 0.67	92.29 ± 0.55	65.60 ± 0.83	**66.68 ± 0.50**	35.97 ± 0.06	**38.82 ± 0.14**
EPE-NAS	93.73 ± 0.40	**93.85 ± 0.28**	71.44 ± 1.17	71.58 ± 0.98	44.28 ± 1.22	**44.62 ± 0.90**
Grad-norm	93.13 ± 0.06	93.13 ± 0.04	68.82 ± 0.81	68.86 ± 0.66	35.97 ± 0.00	**38.83 ± 0.00**
Snip	93.13 ± 0.00	93.13 ± 0.00	69.25 ± 0.42	**69.33 ± 0.00**	35.96 ± 0.31	**38.84 ± 0.12**
Synflow	94.36 ± 0.00	**94.37 ± 0.00**	73.51 ± 0.05	73.51 ± 0.00	46.34 ± 0.00	**46.48 ± 0.02**
L2-norm	93.45 ± 0.09	**93.67 ± 0.03**	71.28 ± 0.02	**71.38 ± 0.19**	46.48 ± 0.21	**46.53 ± 0.02**
Zen	89.42 ± 1.04	**90.64 ± 0.00**	60.72 ± 0.75	**66.11 ± 0.02**	40.77 ± 0.00	40.32 ± 0.00
NWOT	93.58 ± 0.01	**93.67 ± 0.00**	71.51 ± 0.16	**71.67 ± 0.07**	45.33 ± 0.09	**46.53 ± 0.05**
Params	94.36 ± 0.00	94.36 ± 0.00	73.51 ± 0.00	73.51 ± 0.00	46.34 ± 0.00	46.34 ± 0.00
FLOPs	94.36 ± 0.00	94.36 ± 0.00	73.51 ± 0.00	73.51 ± 0.00	46.34 ± 0.00	46.34 ± 0.00

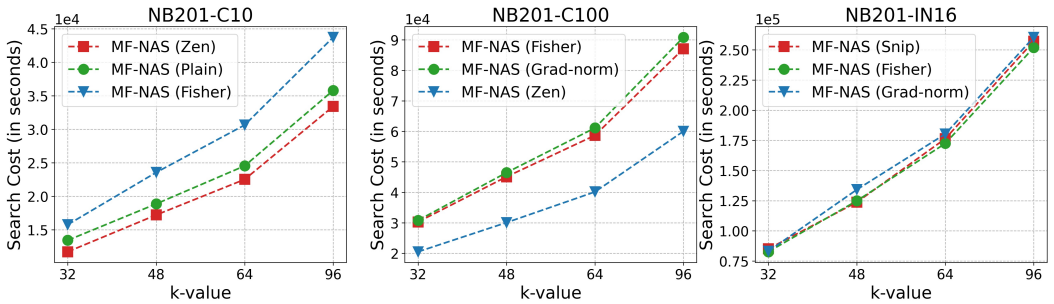
Table 8. The average test accuracy of MF-NAS variants using random search (RS) and local search (LS) in the first stage. We experiment with the three worst-performance ZC proxies (i.e., Plain, Fisher, Zen) and the three best-performance ZC proxies (i.e., Synflow, FLOPs, Params). The *better* method between RS and LS when using the same metric is presented in **red**.

Problem	**Plain**		**Fisher**		**Zen**		**Synflow**		**FLOPs**		**Params**	
	RS	LS	RS	LS	RS	LS	RS	LS	RS	LS	RS	LS
NB201-CF10	**92.65**	92.35	**93.35**	92.21	89.36	**89.42**	94.17	**94.36**	94.11	**94.36**	94.11	**94.36**
NB201-CF100	**69.69**	68.93	**70.08**	65.67	60.32	**60.70**	72.72	**73.51**	72.64	**73.51**	72.62	**73.51**
NB201-IN16	**43.49**	42.74	**44.73**	35.97	35.76	**40.77**	46.24	**46.34**	46.03	**46.34**	46.03	**46.34**

Supplementary. The findings suggest that increasing the number of networks k input to Successive Halving can improve the performance of MF-NAS. However, the magnitude of improvement varies depending on the specific metric and problem (see Fig. 6a). For example, significant improvements are observed for all three metrics in the NB201-IN16 problem, with performance increasing from approximately 36% ($k = 32$) to approximately 45.20% ($k = 96$). In contrast, the improvements for MF-NAS (**Zen**) and MF-NAS (**Plain**) in NB201-CF10 and NB201-CF100 are minimal, with increases of only around 1% (see Fig. 6a). To understand these variations, we visualized 2,000 visited networks of MF-NAS (**Fisher**) and MF-NAS (**Plain**) during the local search stage in terms of their ZC scores and test accuracies for the NB201-CF10 problem in Fig. 7. The results show that performance improvements with increased k values are significant for MF-NAS (**Fisher**) but minimal for MF-NAS (**Plain**). For MF-NAS (**Fisher**), increasing the size of the input network set from $k = 32$ to $k = 48$ results in identifying a higher-performing network compared to the best one in the top-32



(a) Test Accuracy (%)



(b) Search Cost (in seconds)

Fig. 6. Average performance and search cost (in seconds) across 500 independent runs of the three least effective MF-NAS variants (as identified in Table 3) with different k values on NB201-CF10, NB201-CF100, and NB201-IN16 problems.

region (93.13 compared to 91.87). Conversely, there is no improvement in the best found network when k is increased from 32 to 96 for MF-NAS (**Plain**).

Regarding search cost, Fig. 6b shows that the incurred cost scales proportionally with k. Doubling the value of k approximately doubles the search cost, reflecting a trade-off between efficiency and efficacy: higher resource usage yields better results. Despite the increased search cost, it is worth noting that our cost includes fully training the resulting networks, and the overall cost remains feasible for practical applications. For instance, the MF-NAS (**Fisher**) search run on the NB201-IN16 problem required 2.5×10^5 seconds, equivalent to approximately 2.89 GPU days, which is acceptable in real-world scenarios.

5 R-MF-NAS: Robust Multi-Fidelity Neural Architecture Search

5.1 Search Trajectory Networks of Zero-Cost Proxy-Guided Local Search

Fig. 7 partially highlights a drawback of the current mechanism used to select the top$-k$ networks with the highest ZC scores as input for Successive Halving. This mechanism is highly effective when the top$-k$ networks in terms of ZC scores also exhibit high performance (e.g., MF-NAS (**Synflow**)). However, if the top$-k$ networks do not align with top-performing architectures, the performance of the architecture returned by MF-NAS can be poor. Moreover, increasing the value of k in such cases does not necessarily improve the search result (e.g., MF-NAS (**Plain**)).

To investigate the behavior of the ZC proxy-guided local search of MF-NAS, we visualize its search process with Search Trajectory Networks (STNs) [Ochoa et al. 2021]. STNs are *graph-based models*

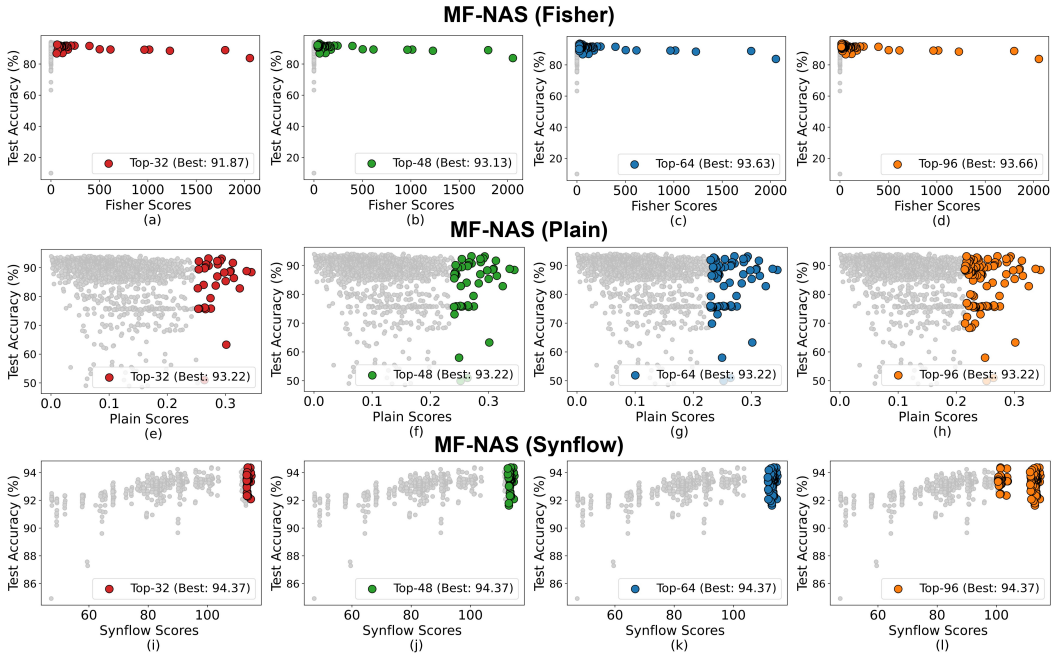
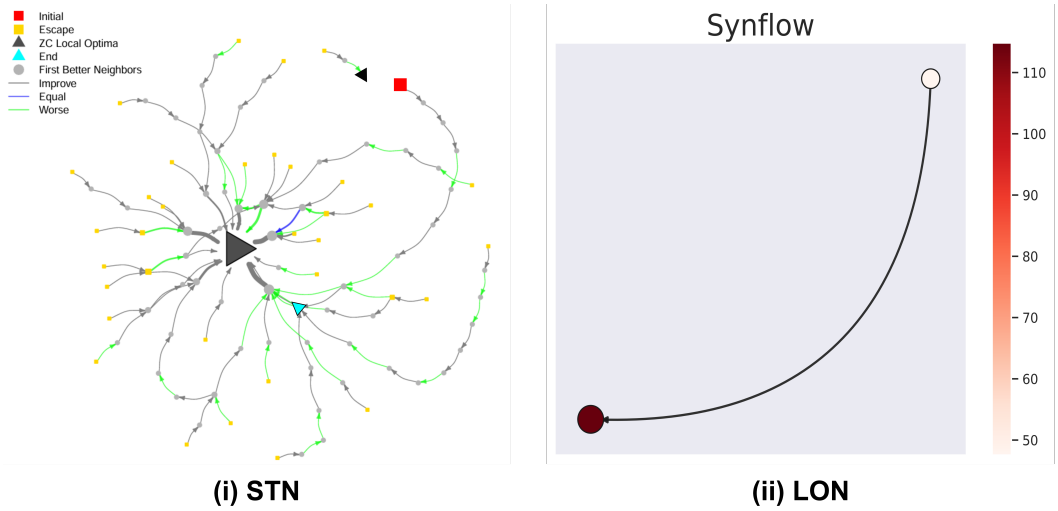


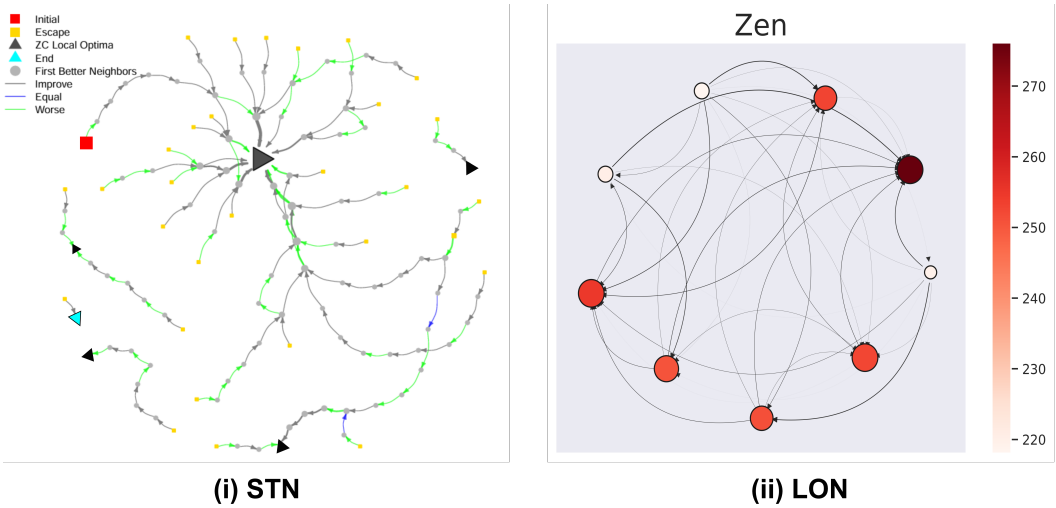
Fig. 7. The ZC scores and test accuracies of 2,000 networks explored at the zero-cost local search stage of MF-NAS (**Fisher**), MF-NAS (**Plain**), and MF-NAS (**Synflow**) with various k values at the median run when solving NB201-CF10 problem across 500 independent runs.

for analyzing and visualizing the behavior of metaheuristics. An STN is constructed from a sequence $\boldsymbol{S}$ of n solutions and their corresponding fitness values $\{(\boldsymbol{x}_1; f(\boldsymbol{x}_1)), (\boldsymbol{x}_2; f(\boldsymbol{x}_2)), \ldots, (\boldsymbol{x}_n; f(\boldsymbol{x}_n))\}$, where each solution $\boldsymbol{x}_t$ and its fitness value $f(\boldsymbol{x}_t)$ are logged at time t during the search process (e.g., the best solution in the population at each generation of an evolutionary algorithm). In an STN, each node represents a solution in $\boldsymbol{S}$ and the edge $e_{i,j}$ $(i < j)$ connects nodes i and j, based on the order of solutions in the sequence $\boldsymbol{S}$. Nodes take one of three shapes: *square* nodes and *triangle* nodes corresponds to the first and the last solutions in $\boldsymbol{S}$, respectively, while circle nodes represent the remaining solutions. Edges in an STN are colored to distinguish the three cases: (1) *Improve*: the fitness value of the next node is better than the current node, (2) *Equal*: the fitness value of the next node is equal to the current node, and (3) *Worse*: the fitness of the next node is worse than the current node.

In this article, we adapt STNs to visualize the ZC proxy-guided local search in MF-NAS. The first-improvement hill climbing algorithm begins by randomly sampling an *initial solution* $\boldsymbol{x}$ (i.e., a network architecture) from the search space. The algorithm then examines the 1-opt neighborhood (i.e., solutions differing by only one structural change from the current solution) and moves to the *first neighbor* $\boldsymbol{x}'$ that improves upon the current one in terms of the chosen ZC metric. This neighborhood exploration continues iteratively until the current solution $\boldsymbol{x}$ is the best in its 1-opt neighborhood (i.e., $\boldsymbol{x}$ is considered a local optimum). At this point, the algorithm executes an *escape operator* by randomly sampling an *escape solution* from the 2-opt neighborhood (i.e., solutions with two structural differences) of the current solution. A new local search is then started from the escape solution. This iterated local search procedure is performed until a stopping criterion is met (e.g., reaching the maximum number of evaluations). We then construct an STN to visualize the



(a) MF-NAS (**Synflow**)



(b) MF-NAS (**Zen**)

Fig. 8. The STNs of the local search processes of MF-NAS (**Synflow**) and MF-NAS (**Zen**) variants at the median run in solving the NB201-CF10 problem.

trajectory of candidate architectures that the ZC proxy-guided local search has explored. In our STNs, the nodes represent: the initial network which corresponds to the start of the sequence (a red square), the first better neighbors encountered by the algorithm during the search (gray circles), the escape networks (yellow squares), and the last network which corresponds to the end of the sequence (a blue triangle). Our STNs also depict three types of edges: (1) *Improve* (gray), (2) *Equal* (blue), and (3) *Worse* (green). The type of a specific edge e_{ij} is determined by comparing the fitness value $f(\cdot)$ of two solutions representing nodes i and j, where the fitness value $f(\boldsymbol{x})$ of a network $\boldsymbol{x}$ corresponds to its test accuracy. We note that the local search in MF-NAS is guided by a ZC metric and the test accuracies of the architectures are computed *a posteriori* to color the edges of

its STN for the analysis purpose. It means that the first better neighbors and the local optima in the sequence $\boldsymbol{S}$ are determined based on their ZC scores instead of test accuracy.

Fig. 8 exhibits two STNs of the ZC proxy-guided local search processes of MF-NAS (**Synflow**) and MF-NAS (**Zen**) in solving the NB201-CF10 problem and we could observe a clear difference between these two STNs. While both STNs have numerous escape nodes (i.e., yellow squares), the STN of MF-NAS (**Synflow**) only has two gray triangles representing two distinct ZC local optima, unlike MF-NAS (**Zen**), which has many gray triangles. The difference can be explained by considering the LONs corresponding to the Synflow and Zen metrics. As shown in Fig. 8, the LON for Zen contains 9 distinct optima, whereas the LON for Synflow has only two. Consequently, the neighborhood exploration using Synflow is more likely to converge to the same optimum, even when we execute this process with different solutions (i.e., escape solutions).

Table 9. Analysis results of STNs (at the median run across 500 independent runs) of MF-NAS with different ZC metrics when solving the NB201-CF10 problem.

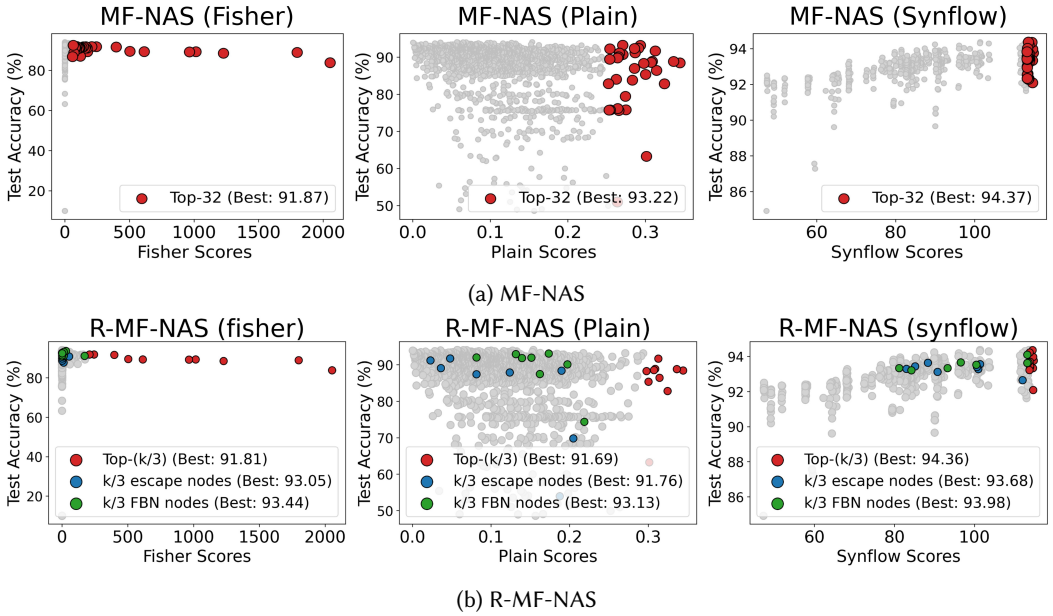
Value	**Node**	MF-NAS (**Zen**)	MF-NAS (**Plain**)	MF-NAS (**Fisher**)	MF-NAS (**Synflow**)	MF-NAS (**Params**)	MF-NAS (**FLOPs**)
Average	Escape	90.97	87.11	77.76	93.13	93.21	93.10
	First Better Neighbors	91.00	85.77	90.12	93.36	93.38	93.35
	ZC Local Optima	88.63	84.98	87.85	93.76	93.76	93.76
Best	Escape	94.12	93.66	93.15	94.02	94.29	94.26
	First Better Neighbors	94.36	93.37	93.96	94.10	94.36	94.36
	ZC Local Optima	90.64	93.48	90.78	93.76	93.76	93.76

In addition to MF-NAS (**Zen**) and MF-NAS (**Synflow**) metrics, we use STNs to visualize the search processes of MF-NAS (**Plain**) and MF-NAS (**Fisher**) (two ineffective variants for the NB201-CF10 problem) as well as MF-NAS (**Params**) and MF-NAS (**FLOPs**) (two effective variants for the same problem). Analyzing the STNs of the top-performing variants (i.e., MF-NAS (**Synflow**), MF-NAS (**Params**), MF-NAS (**FLOPs**)), we could observe a consistent trend: *the test accuracies of architectures at first-better-neighbor nodes are higher than those at escape nodes*, and *the test accuracies of architectures at ZC local optima nodes are also higher than those at first-better-neighbor nodes* (see Table 9). In contrast, this trend does not exist for ineffective variants (MF-NAS (Zen), MF-NAS (Plain), and MF-NAS (Fisher)), where the average test accuracy of ZC local optima nodes is lower than that of escape or first-better-neighbor nodes. This discrepancy partially explains why the inputs for Successive Halving in these ineffective variants primarily consist of low-performance networks. Interestingly, the analysis in Table 9 reveals that high-performance networks exist among escape and first-better-neighbor nodes, even for the ineffective variants. If these high-performance networks could be effectively sampled and used as inputs for Successive Halving, the final efficacy of MF-NAS could remain competitive, even when using an unsuitable ZC metric during local search.

5.2 Proposed Framework

Based on the above insights from STN analyses of different ZC proxy-guided local searches, we propose a more robust version of MF-NAS, namely R-MF-NAS. The key difference between MF-NAS and R-MF-NAS lies in the mechanism for selecting network architectures as inputs for the Successive Halving stage. Instead of exclusively selecting the top-k candidate networks with the highest ZC scores, R-MF-NAS additionally includes architectures that correspond to the escape

and the first better-neighbor nodes. The ratios among the three types of solutions in the input set for SH are set equally. Specifically, the k input architectures consist of $k/3$ networks with the highest ZC scores, $k/3$ escape networks, and $k/3$ networks representing the first better neighbors (FBN) encountered by the local search algorithm. Fig. 9 illustrates the difference in the selection mechanisms between MF-NAS $k = 32$ and R-MF-NAS $k = 32$ at the same run.



(a) MF-NAS

(b) R-MF-NAS

Fig. 9. Differences in the selection mechanisms of MF-NAS $k = 32$ and R-MF-NAS $k = 32$ at the same run.

5.3 Experiments and Discussion

5.3.1 Performance of R-MF-NAS on NAS benchmark problems. We verify the robustness of the R-MF-NAS framework on the problems listed in Table 1. The experimental settings are consistent with the experiments for MF-NAS in Section 4.3.1. Table 10 presents the average results of R-MF-NAS variants with various ZC metrics over 500 independent runs. The capability of R-MF-NAS to identify top-performing networks is verified, as the networks discovered by the best R-MF-NAS variants are close to the optimal ones across all problems. Table 11 provides the comparisons in the test performance of best and worst architectures found by MF-NAS and R-MF-NAS frameworks with different ZC metrics. Overall, the average performance of R-MF-NAS variants surpasses that of MF-NAS for all problems. The results also highlight the higher robustness of R-MF-NAS over MF-NAS, as the performance gap between the worst and the best R-MF-NAS variants is significantly smaller than the gap observed in MF-NAS. Specifically, the performance of the worst R-MF-NAS variant for each problem is substantially better than that of the worst MF-NAS variant across all problems. Moreover, the best R-MF-NAS variant achieves performance that is either comparable to (for NB201-CF100 and NBASR-TIMIT) or better than (for NB101-CF10, NB201-CF10, and NB201-IN16) the best MF-NAS variants. However, the new selection mechanism slightly decreases performance in certain cases (e.g., comparing the performance of MF-NAS (**Params**) (Table 3) and R-MF-NAS (**Params**) (Table 10) at the NB201-CF100 problem). This may occur because top-performing networks in the region between the top$-(k/3)$ and top$-k$ networks may be discarded when selecting only the

Table 10. Average test performance (Mean ± SD) of best architectures found by R-MF-NAS variants on NAS problems over 500 runs. For problems having symbol ↑, the algorithms achieving **higher** scores are better. For problems having symbol ↓, the algorithms achieving **lower** scores are better. The best and worst results are exhibited in **red** and **blue**. Symbol '-' means that those ZC proxies are unsupported for the NAS problems under concern. The results are exhibited in the following three blocks. The first block presents the results of R-MF-NAS with various ZC proxies for each problem. The second block presents the average results (Mean ± SD) of R-MF-NAS with various ZC proxies and the difference in performance between the best and worst variants. The last block presents the test performance of the optimal architectures for each problem. Note that these optimal architectures are queried from the NAS benchmark. According to the Student's t-test, the algorithms that are significantly worse than the best ones at a significance level of 0.01 with Bonferroni correction are presented in gray cells.

ZC Metric	NB101-CF10 ↑	NB201-CF10 ↑	NB201-CF100 ↑	NB201-IN16 ↑	NBASR-TIMIT ↓
Snip	92.06 ± 1.04	93.51 ± 0.42	70.71 ± 1.10	44.91 ± 1.21	22.36 ± 0.57
Grasp	93.19 ± 0.31	93.54 ± 0.41	70.79 ± 0.94	44.77 ± 1.30	-
Fisher	91.94 ± 1.02	93.35 ± 0.45	70.22 ± 1.07	44.29 ± 1.52	22.75 ± 0.81
Plain	-	**92.90 ± 0.68**	**69.72 ± 1.19**	**43.03 ± 2.07**	22.37 ± 0.60
Grad-norm	**91.86 ± 1.24**	93.47 ± 0.43	70.66 ± 1.05	44.77 ± 1.32	22.47 ± 0.64
EPE-NAS	-	93.63 ± 0.50	70.86 ± 1.15	44.32 ± 1.34	-
Jacov	92.94 ± 0.75	93.69 ± 0.39	71.67 ± 0.85	44.45 ± 1.15	22.48 ± 0.40
L2-norm	-	93.79 ± 0.35	71.67 ± 0.92	45.72 ± 0.66	**23.28 ± 0.74**
Zen	-	93.99 ± 0.31	72.23 ± 1.06	45.73 ± 0.82	-
NWOT	-	93.79 ± 0.38	71.66 ± 0.88	45.72 ± 0.65	-
Synflow	93.78 ± 0.24	**94.32 ± 0.08**	**73.51 ± 0.00**	**46.41 ± 0.14**	**21.78 ± 0.08**
FLOPs	93.91 ± 0.24	94.06 ± 0.36	72.58 ± 0.97	45.82 ± 1.05	22.48 ± 0.40
Params	**93.92 ± 0.23**	94.05 ± 0.36	72.69 ± 0.92	45.86 ± 0.98	21.86 ± 0.08
Mean ± SD	92.95 ± 0.84	93.70 ± 0.35	71.46 ± 1.05	45.06 ± 0.89	22.43 ± 0.42
Gap (Best, Worst)	2.06	1.42	3.79	3.38	1.50
Optimal	*94.32*	*94.37*	*73.51*	*47.31*	*21.40*

top$-(k/3)$ networks with the highest ZC scores. Nonetheless, the performance of R-MF-NAS can be improved by increasing the value of k, similarly to MF-NAS. Fig. 10 illustrates the performance of MF-NAS (**Params**) and R-MF-NAS (**Params**) at the NB201-CF100 problem with varying k values $\{32, 48, 64, 96\}$. The results indicate that the performance gap between R-MF-NAS and MF-NAS is narrowed at $k = \{48, 64\}$ and no longer exists at $k = 96$ (the difference is 0.0). Additionally, comparisons of MF-NAS and R-MF-NAS frameworks across different k values show that almost all R-MF-NAS variants outperform MF-NAS variants when executed with the same value of k (see Table 12 for the comparisons on the NB101-CF10 problem. The results for the remaining problems are provided in Appendix D of the Supplementary).

5.3.2 Comparison of Candidate Architecture Selection Mechanisms for Successive Halving. We compare R-MF-NAS with the following variants to validate the effectiveness of our newly-proposed selection mechanism:

- **RS + SH**: This variant replaces Local Search with Random Search to explore 2,000 networks within the search space in the first stage. The top$-k$ networks with the highest ZC scores are then selected as input for Successive Halving (SH).

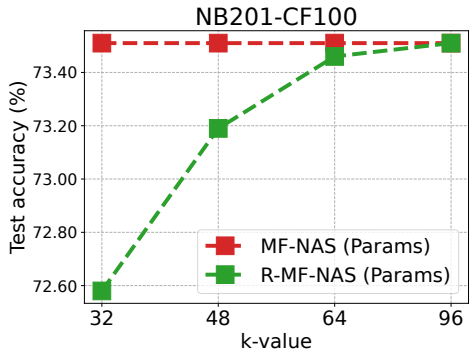


Fig. 10. The average test accuracy of MF-NAS (**Params**) and R-MF-NAS (**Params**) on the NB201-CF100 problem with varying k values {32, 48, 64, 96} across 500 independent runs.

- **MF-NAS**: The candidate set for SH consists of the top$-k$ networks with the highest ZC scores, selected from the 2,000 networks explored using Local Search.
- **MF-NAS***: This variant also explores the search space at the first stage using Local Search similar MF-NAS. However, instead of selecting top$-k$ networks with the highest ZC scores, MF-NAS* *randomly samples* $k-1$ networks from the 2,000 explored networks, along with the network having the highest ZC score, as input for SH.

Each variant is executed over 500 independent runs, with the experimental settings as described in Section 4.3.1.

Comparing the two variants **MF-NAS** and **MF-NAS***, Table 13 highlights both the strengths and limitations of the two approaches for creating the input of SH: *selecting networks with the highest ZC scores* and *randomly sampling networks* from the pool of networks explored via local search. When a suitable ZC metric is used, **MF-NAS** identifies networks with superior test performance compared to those found by **MF-NAS*** (comparing the best variants of each method). However, with an unsuitable ZC metric, **MF-NAS** may also return networks with poor performance. In contrast, **MF-NAS*** demonstrates better robustness, as the performance gap between its best and worst variants is significantly smaller than that of **MF-NAS**. Additionally, the average test performance of networks identified by **MF-NAS*** is higher than that of **MF-NAS** across various ZC metrics.

The results further emphasize the effectiveness of local search in the first stage over random search. Networks discovered by the best variant of **MF-NAS** are better than those identified by the best variant of **RS + SH**. Moreover, **MF-NAS*** exhibits greater robustness with different ZC metrics compared to **RS + SH**. Despite both methods relying on random sampling, **RS + SH** samples directly from the search space, while **MF-NAS*** samples from the set of networks explored by local search. This suggests that exploring the search space via local search and sampling from the resulting pool yields higher-quality candidates for SH compared to direct sampling from the search space. In the case of **R-MF-NAS**, by incorporating both the networks with the highest ZC scores and those sampled from the explored pool, its best variant outperforms **MF-NAS*** and is only slightly inferior to **MF-NAS** on certain problem instances. Similar to **MF-NAS***, **R-MF-NAS** demonstrates impressive robustness across different ZC metrics, significantly outperforming methods that rely solely on the top$-k$ networks with the highest ZC scores, such as **RS + SH** and **MF-NAS**.

Table 11. Test performance comparisons between MF-NAS and R-MF-NAS with different ZC metrics for all problems listed in Table 1. The better results are presented in **red**. The results that are *significantly worse* than the best one (by checking with Student's t-tests at a significance level of 0.01 with Bonferroni correction) are displayed in the gray cells.

	Performance	MF-NAS	R-MF-NAS
NB101-CF10 ↑	Best	93.89 ± 0.25	**93.92 ± 0.23**
	Worst	84.01 ± 1.50	**91.86 ± 1.24**
	Mean ± SD	90.18 ± 4.20	**92.95 ± 0.84**
	Gap (Best, Worst)	9.88	**2.06**
NB201-CF10 ↑	Best	**94.36 ± 0.00**	94.32 ± 0.08
	Worst	89.42 ± 1.04	**92.90 ± 0.68**
	Mean ± SD	93.13 ± 1.26	**93.70 ± 0.35**
	Gap (Best, Worst)	4.94	**1.40**
NB201-CF100 ↑	Best	**73.51 ± 0.00**	**73.51 ± 0.00**
	Worst	60.72 ± 0.75	**69.72 ± 1.19**
	Mean ± SD	69.96 ± 3.45	**71.46 ± 1.05**
	Gap (Best, Worst)	12.79	**3.79**
NB201-IN16 ↑	Best	**46.48 ± 0.21**	46.41 ± 0.14
	Worst	35.96 ± 0.31	**43.03 ± 2.07**
	Mean ± SD	42.61 ± 3.98	**45.06 ± 0.89**
	Gap (Best, Worst)	10.52	**3.38**
NBASR-TIMIT ↓	Best	**21.77 ± 0.00**	21.78 ± 0.08
	Worst	83.14 ± 7.99	**23.28 ± 0.74**
	Mean ± SD	29.41 ± 19.01	**22.42 ± 0.42**
	Gap (Best, Worst)	61.37	**1.50**

6 Conclusion

In this study, we investigated the effectiveness of zero-cost (ZC) proxies in guiding a typical local search algorithm to seek top-performing neural network architectures. Fitness landscape analyses for a series of ZC proxies showed that using solely ZC proxies was not effective and the obtained architectures were inferior compared to using training-based metrics such as validation accuracy or training loss. The accuracy values of architectures at the global optima of ZC proxy landscapes are much lower than the optimal accuracy, and few architectures simultaneously have high ZC proxy scores and high test accuracies. This explains why NAS algorithms using a ZC proxy as the search objective might return subpar architectures although they identify the networks that have high ZC proxy scores. Next, we observed that the landscapes of ZC proxies allowed local search algorithms to approach efficiently the regions containing promising architectures although the landscapes

Table 12. Test accuracy performance comparison between MF-NAS and R-MF-NAS variants with different k values {16, 32, 48, 64} for the NB101-CF10 problem. The better results are presented in **red**. The results that are significantly worse than the best one (by checking with Student's t-tests at a significance level of 0.01 with Bonferroni correction) are displayed in the gray cells.

Variant	$k = 16$	$k = 32$	$k = 48$	$k = 64$
MF-NAS (**Jacob**)	92.20 ± 1.51	92.93 ± 0.68	93.19 ± 0.51	93.26 ± 0.49
R-MF-NAS (**Jacob**)	**92.94 ± 0.75**	**93.29 ± 0.27**	**93.38 ± 0.21**	**93.43 ± 0.19**
MF-NAS (**Fisher**)	85.51 ± 2.72	86.94 ± 1.76	87.51 ± 1.48	87.88 ± 1.46
R-MF-NAS (**Fisher**)	**91.94 ± 1.02**	**92.54 ± 0.65**	**92.78 ± 0.48**	**92.90 ± 0.43**
MF-NAS (**Grad-norm**)	84.01 ± 1.50	87.17 ± 2.71	87.86 ± 2.57	88.38 ± 2.41
R-MF-NAS (**Grad-norm**)	**91.86 ± 1.24**	**92.53 ± 0.70**	**92.76 ± 0.58**	**92.91 ± 0.50**
MF-NAS (**Snip**)	84.67 ± 1.58	88.83 ± 2.97	89.40 ± 2.66	89.97 ± 2.37
R-MF-NAS (**Snip**)	**92.06 ± 1.04**	**92.61 ± 0.69**	**92.84 ± 0.55**	**92.99 ± 0.46**
MF-NAS (**Synflow**)	**93.82 ± 0.56**	93.94 ± 0.36	93.98 ± 0.32	93.99 ± 0.30
R-MF-NAS (**Synflow**)	93.78 ± 0.24	**93.97 ± 0.20**	**94.08 ± 0.16**	**94.13 ± 0.13**
MF-NAS (**Params**)	93.89 ± 0.25	94.02 ± 0.19	94.06 ± 0.17	94.07 ± 0.17
R-MF-NAS (**Params**)	**93.92 ± 0.23**	**94.05 ± 0.14**	**94.11 ± 0.11**	**94.13 ± 0.09**
MF-NAS (**FLOPs**)	93.88 ± 0.25	94.02 ± 0.19	94.06 ± 0.18	94.07 ± 0.17
R-MF-NAS (**FLOPs**)	**93.92 ± 0.23**	**94.06 ± 0.15**	**94.11 ± 0.11**	**94.13 ± 0.10**
MF-NAS (**Grasp**)	93.15 ± 0.33	93.34 ± 0.40	93.43 ± 0.45	93.46 ± 0.47
R-MF-NAS (**Grasp**)	**93.19 ± 0.31**	**93.39 ± 0.23**	**93.47 ± 0.20**	**93.52 ± 0.20**
Optimal (in the benchmark)		*94.32*		

are multi-modal. Based on these insights, we introduced the Multi-Fidelity Neural Architecture Search (MF-NAS) framework, which makes use of both ZC metrics and training-based metrics to find high-performance architectures with an acceptable cost. MF-NAS first explores the search space using a ZC-proxy-guided local search and the architectures that have the highest ZC proxy scores are then forwarded to the Successive Halving procedure that uses a training-based metric to yield a final best architecture with its associated network weights. Extensive experiments on NAS benchmarks demonstrated the consistent efficacy and efficiency of our MF-NAS in achieving high-quality architectures compared to state-of-the-art training-free NAS algorithms.

However, the limitation of MF-NAS is its robustness with different ZC metrics. The effectiveness of ZC proxy-guided NAS relies substantially on the correlation between the employed ZC metric and accuracy performance, which depends on the specific NAS problem under concern and is not available a priori. Using search trajectory networks (STNs), we analyzed the search behavior of the training-free local search in MF-NAS with respect to different ZC metrics. The analysis exhibited that, for ZC metrics with poor correlations, the top-performing networks might appear at the beginning or middle stages of the local search process rather than at the end. We thus proposed a more robust variant of MF-NAS, namely R-MF-NAS, that modified the candidate selection mechanism for Successive Halving. In addition to selecting networks with the highest ZC scores,

Table 13. Test performance comparisons between different approaches of selecting candidate networks for the Successive Halving stage for all problems listed in Table 1. The best results are presented in **red**. The results that are significantly worse than the best one (by checking with Student's t-tests at a significance level of 0.01 with Bonferroni correction) are displayed in the gray cells.

	Performance	RS + SH	MF-NAS	MF-NAS★	R-MF-NAS
NB101-C10 ↑	Best	93.80 ± 0.24	93.89 ± 0.25	93.79 ± 0.25	**93.92 ± 0.23**
	Worst	91.93 ± 0.76	84.01 ± 1.50	**92.25 ± 0.89**	91.86 ± 1.24
	Mean ± SD	92.85 ± 0.81	90.18 ± 4.20	**93.08 ± 0.63**	92.95 ± 0.84
	Gap (Best, Worst)	1.87	9.88	**1.54**	2.06
NB201-C10 ↑	Best	94.17 ± 0.20	**94.36 ± 0.00**	94.01 ± 0.29	94.32 ± 0.02
	Worst	89.36 ± 1.11	89.42 ± 1.04	**93.00 ± 0.63**	92.90 ± 0.68
	Mean ± SD	93.30 ± 1.20	93.13 ± 1.26	**93.71 ± 0.26**	93.70 ± 0.35
	Gap (Best, Worst)	4.81	4.94	**1.01**	1.42
NB201-C100 ↑	Best	72.64 ± 0.77	**73.51 ± 0.00**	72.26 ± 0.90	**73.51 ± 0.00**
	Worst	60.32 ± 0.92	60.72 ± 0.75	**69.92 ± 1.26**	69.72 ± 1.19
	Mean ± SD	70.48 ± 3.08	69.96 ± 3.45	71.43 ± 0.66	**71.46 ± 1.05**
	Gap (Best, Worst)	12.32	12.79	**2.34**	3.79
NB201-IN16 ↑	Best	46.24 ± 0.39	**46.48 ± 0.21**	45.82 ± 0.86	46.41 ± 0.05
	Worst	35.76 ± 2.33	35.96 ± 0.31	**43.32 ± 1.86**	43.03 ± 2.07
	Mean ± SD	44.63 ± 2.67	42.61 ± 3.98	**45.12 ± 0.70**	45.06 ± 0.89
	Gap (Best, Worst)	10.49	10.52	**2.50**	3.38
NBASR-TIMIT ↓	Best	22.00 ± 0.18	**21.77 ± 0.00**	22.06 ± 0.20	21.78 ± 0.08
	Worst	84.24 ± 0.17	83.14 ± 7.99	23.77 ± 5.72	**23.28 ± 0.74**
	Mean ± SD	29.14 ± 19.49	29.41 ± 19.01	22.86 ± 0.56	**22.43 ± 0.40**
	Gap (Best, Worst)	62.24	61.37	1.71	**1.50**

R-MF-NAS also included architectures that were (1) escape solutions created when the algorithm tried to escape a local optimum and (2) first-improvement neighbor solutions encountered during the local search process. Experiments on diverse NAS benchmarks demonstrated the superior robustness of R-MF-NAS compared to MF-NAS, while also showing its ability to identify high-performance architectures under a limited budget. Extending MF-NAS and R-MF-NAS to the multi-objective NAS scenario, as well as validating the use of multiple training-free metrics to guide local search in exploring the search space, are potential future works.

Acknowledgments

This research was supported by The VNUHCM-University of Information Technology's Scientific Research Support Fund.

References

Mohamed S. Abdelfattah, Abhinav Mehrotra, Lukasz Dudziak, and Nicholas Donald Lane. 2021. Zero-Cost Proxies for Lightweight NAS. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net.

Marco Baioletti, Alfredo Milani, Valentino Santucci, and Marco Tomassini. 2019. Search Moves in the Local Optima Networks of Permutation Spaces: The QAP Case. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019*. ACM, 1535–1542.

Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2018. Accelerating Neural Architecture Search using Performance Prediction. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, Canada.

Niccolò Cavagnero, Luca Robbiano, Barbara Caputo, and Giuseppe Averta. 2023. FreeREA: Training-Free Evolution-based Architecture Search. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2023*. IEEE, 1493–1502.

Alexander Chebykin, Tanja Alderliesten, and Peter A. N. Bosman. 2022. Evolutionary neural cascade search across supernetworks. In *GECCO*. Boston, Massachusetts, USA, 1038–1047.

Hanlin Chen, Ming Lin, Xiuyu Sun, and Hao Li. 2021b. NAS-Bench-Zero: A Large Scale Dataset for Understanding Zero-Shot Neural Architecture Search. In *OpenReview.net*.

Wuyang Chen, Xinyu Gong, and Zhangyang Wang. 2021a. Neural Architecture Search on ImageNet in Four GPU Hours: A Theoretically Inspired Perspective. In *International Conference on Learning Representations (ICLR)*.

Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. 2021c. DrNAS: Dirichlet Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*.

Xiangxiang Chu, Bo Zhang, and Ruijun Xu. 2021. FairNAS: Rethinking Evaluation Fairness of Weight Sharing Neural Architecture Search. In *International Conference on Computer Vision (ICCV)*. 12219–12228.

Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. 2022. NATS-Bench: Benchmarking NAS Algorithms for Architecture Topology and Size. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 7 (2022), 3634–3646.

Xuanyi Dong and Yi Yang. 2019a. One-Shot Neural Architecture Search via Self-Evaluated Template Network. In *International Conference on Computer Vision (ICCV)*. 3680–3689.

Xuanyi Dong and Yi Yang. 2019b. Searching for a Robust Neural Architecture in Four GPU Hours. In *Computer Vision and Pattern Recognition (CVPR)*. 1761–1770.

Xuanyi Dong and Yi Yang. 2020. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*.

Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *International Conference on Machine Learning (ICML)*. PMLR, 1436–1445.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020. Single Path One-Shot Neural Architecture Search with Uniform Sampling. In *Computer Vision (ECCV)*. 544–560.

Kevin G. Jamieson and Ameet Talwalkar. 2016. Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 240–248.

Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. 2022. NAS-Bench-Suite-Zero: Accelerating Research on Zero Cost Proxies. In *Neural Information Processing Systems (NeurIPS)*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2019. Snip: Single-Shot Network Pruning based on Connection sensitivity. In *International Conference on Learning Representations (ICLR)*.

Liam Li and Ameet Talwalkar. 2019. Random Search and Reproducibility for Neural Architecture Search. In *Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 367–377.

Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. 2021. Zen-NAS: A Zero-Shot NAS for High-Performance Image Recognition. In *International Conference on Computer Vision (ICCV)*. 337–346.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. In *ICLR*.

Vasco Lopes, Saeid Alirezazadeh, and Luís A. Alexandre. 2021. EPE-NAS: Efficient Performance Estimation Without Training for Neural Architecture Search. In *International Conference on Artificial Neural Networks (ICANN)*. 552–563.

Vasco Lopes, Miguel Santos, Bruno Degardin, and Luís A. Alexandre. 2022. Efficient Guided Evolution for Neural Architecture Search. In *GECCO*. 655–658.

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2018. Neural Architecture Optimization. In *Neural Information Processing Systems (NeurIPS)*. 7827–7838.

Ngoc Hoang Luong, Quan Minh Phan, An Vo, Tan Ngoc Pham, and Dzung Tri Bui. 2024. Lightweight multi-objective evolutionary neural architecture search with low-cost proxy metrics. *Inf. Sci.* 655 (2024), 119856. doi:10.1016/J.INS.2023.119856

Abhinav Mehrotra, Alberto Gil C. P. Ramos, Sourav Bhattacharya, Lukasz Dudziak, Ravichander Vipperla, Thomas Chau, Mohamed S. Abdelfattah, Samin Ishtiaq, and Nicholas Donald Lane. 2021. NAS-Bench-ASR: Reproducible Neural Architecture Search for Speech Recognition. In *International Conference on Learning Representations (ICLR)*.

Joe Mellor, Jack Turner, Amos J. Storkey, and Elliot J. Crowley. 2021. Neural Architecture Search without Training. In *International Conference on Machine Learning (ICML)*. PMLR, 7588–7598.

Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. 2021. Evaluating Efficient Performance Estimators of Neural Architectures. In *NeurIPS*. 12265–12277.

Gabriela Ochoa, Katherine M. Malan, and Christian Blum. 2021. Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. *Appl. Soft Comput.* 109 (2021), 107492.

Gabriela Ochoa, Marco Tomassini, Sébastien Vérel, and Christian Darabos. 2008. A study of NK landscapes' basins and local optima networks. In *GECCO*. 555–562.

Gabriela Ochoa and Nadarajen Veerapen. 2022. Neural Architecture Search: A Visual Analysis. In *Parallel Problem Solving from Nature (PPSN)*. 603–615.

Gabriela Ochoa, Nadarajen Veerapen, Fabio Daolio, and Marco Tomassini. 2017. Understanding Phase Transitions with Local Optima Networks: Number Partitioning as a Case Study. In *EvoCOP*. 233–248.

Yameng Peng, Andy Song, Vic Ciesielski, Haytham M. Fayek, and Xiaojun Chang. 2022. PRE-NAS: predictor-assisted evolutionary neural architecture search. In *GECCO*. 1066–1074.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. 2018. Efficient Neural Architecture Search via Parameter Sharing. In *International Conference on Machine Learning (ICML)*. PMLR, 4092–4101.

Quan Minh Phan and Ngoc Hoang Luong. 2023. Pareto Local Search is Competitive with Evolutionary Algorithms for Multi-Objective Neural Architecture Search. In *GECCO*. Lisbon, Portugal, 348–356.

Quan Minh Phan and Ngoc Hoang Luong. 2024. Efficient Multi-Fidelity Neural Architecture Search with Zero-Cost Proxy-Guided Local Search. In *GECCO*. Melbourne, VIC, Australia.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. 2019. Regularized Evolution for Image Classifier Architecture Search. In *AAAI Conference on Artificial Intelligence (AAAI)*. 4780–4789.

Christian M. Reidys and Peter F. Stadler. 2001. Neutrality in fitness landscapes. *Appl. Math. Comput.* 117, 2-3 (2001), 321–350.

Robin Ru, Clare Lyle, Lisa Schut, Miroslav Fil, Mark van der Wilk, and Yarin Gal. 2021. Speedy Performance Estimation for Neural Architecture Search. In *Neural Information Processing Systems (NeurIPS)*. 4079–4092.

C. Spearman. 1904. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology* 15, 1 (1904), 72–101.

Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning (ICML)*. PMLR, 6105–6114.

Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Neural Information Processing Systems (NeurIPS)*.

Sarah L. Thomson, Gabriela Ochoa, Nadarajen Veerapen, and Krzysztof Michalak. 2023. Channel Configuration for Neural Architecture: Insights from the Search Space. In *GECCO*. 1267–1275.

Jack Turner, Elliot J. Crowley, Michael F. P. O'Boyle, Amos J. Storkey, and Gavin Gray. 2020. BlockSwap: Fisher-guided Block Substitution for Network Compression on a Budget. In *International Conference on Learning Representations (ICLR)*.

Chaoqi Wang, Guodong Zhang, and Roger B. Grosse. 2020. Picking Winning Tickets Before Training by Preserving Gradient Flow. In *International Conference on Learning Representations (ICLR)*.

Ruochen Wang, Xiangning Chen, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. 2021a. RANK-NOSH: Efficient Predictor-Based Architecture Search via Non Uniform Successive Halving. In *ICCV*. 10357–10366.

Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. 2021b. Rethinking Architecture Selection in Differentiable NAS. In *International Conference on Learning Representations (ICLR)*.

Colin White, Mikhail Khodak, Renbo Tu, Shital Shah, Sébastien Bubeck, and Debadeepta Dey. 2022. A Deeper Look at Zero-Cost Proxies for Lightweight NAS. In *ICLR Blog Track*. https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/

Colin White, Willie Neiswanger, and Yash Savani. 2021a. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search. In *AAAI Conference on Artificial Intelligence (AAAI)*. 10293–10301.

Colin White, Sam Nolen, and Yash Savani. 2021b. Exploring the Loss Landscape in Neural Architecture Search. In *Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 654–664.

Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* 8 (1992), 229–256.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. In *Computer Vision and Pattern Recognition (CVPR)*. 10734–10742.

Han Xiao, Ziwei Wang, Zheng Zhu, Jie Zhou, and Jiwen Lu. 2022. Shapley-NAS: Discovering Operation Contribution for Neural Architecture Search. In *Computer Vision and Pattern Recognition (CVPR)*. 11882–11891.

Lingxi Xie and Alan L. Yuille. 2017. Genetic CNN. In *ICCV*. IEEE Computer Society, 1388–1397.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. 2020. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In *ICLR*.

Shen Yan, Colin White, Yash Savani, and Frank Hutter. 2021. NAS-Bench-x11 and the Power of Learning Curves. In *Neural Information Processing Systems (NeurIPS)*. 22534–22549.

Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. 2020. Does Unsupervised Architecture Representation Learning Help Neural Architecture Search?. In *Neural Information Processing Systems (NeurIPS)*.

Taojiannan Yang, Linjie Yang, Xiaojie Jin, and Chen Chen. 2023. Revisiting Training-free NAS Metrics: An Efficient Training-based Method. In *Winter Conference on Applications of Computer Vision (WACV)*. 4740–4749.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. 2019. NAS-Bench-101: Towards Reproducible Neural Architecture Search. In *International Conference on Machine Learning (ICML)*. PMLR, 7105–7114.

Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. 2020. Evaluating The Search Phase of Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*.

Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *ICLR*.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning Transferable Architectures for Scalable Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. UT, USA, 8697–8710.

Feng Zou, Debao Chen, Hui Liu, Siyu Cao, Xuying Ji, and Yan Zhang. 2022. A Survey of Fitness Landscape Analysis for Optimization. *Neurocomputing* 503 (2022), 129–139.

Received 7 December 2024; revised 2 July 2025; accepted 30 October 2025