Neural Architecture Search as Multiobjective Optimization Benchmarks: Problem Formulation and Performance Assessment

Zhichao Lu*, Member, IEEE, Ran Cheng*, Senior Member, IEEE, Yaochu Jin†, Fellow, IEEE, Kay Chen Tan*, Fellow, IEEE, and Kalyanmoy Deb, Fellow, IEEE

Abstract—The ongoing advancements in network architecture design have led to remarkable achievements in deep learning across various challenging computer vision tasks. Meanwhile, the development of neural architecture search (NAS) has provided promising approaches to automating the design of network architectures for lower prediction error. Recently, the emerging application scenarios of deep learning (e.g., autonomous driving) have raised higher demands for network architectures considering multiple design criteria: number of parameters/weights, number of floating-point operations, inference latency, among others. From an optimization point of view, the NAS tasks involving multiple design criteria are intrinsically multiobjective optimization problems; hence, it is reasonable to adopt evolutionary multiobjective optimization (EMO) algorithms to tackle them. Nonetheless, there is still a clear gap confining the related research along this pathway: on the one hand, there is a lack of a general problem formulation of NAS tasks from an optimization point of view; on the other hand, there are challenges in conducting benchmark assessments of EMO algorithms on NAS tasks. To bridge the gap: 1) we formulate NAS tasks into general multiobjective optimization problems and analyze the complex characteristics from an optimization point of view; 2) we present an end-to-end pipeline, dubbed EvoXBench, to generate benchmark test problems for EMO algorithms to run efficiently—without the requirement of GPUs or Pytorch/Tensorflow; and 3) we instantiate two test suites comprehensively covering two datasets, seven search spaces, and three hardware devices, involving up to eight objectives. Based on the above, we validate the proposed test suites using six representative EMO algorithms and provide some empirical analyses. The code of EvoXBench is available at https://github.com/EMI-Group/EvoXBench.

Index Terms—Deep learning, evolutionary multiobjective optimization (EMO), neural architecture search (NAS).

I. INTRODUCTION

W

ith the development of deep neural networks (DNNs), deep learning has seen widespread growth in both research and applications, driven by improvements in computing power and a massive amount of data [1], [2], [3]. As the DNN models become more complex, it has come to a consensus that the manual process of designing DNN architectures is beyond human labor and thus unsustainable [4], [5], [6]. Meanwhile, the emergence of neural architecture search (NAS) has paved a promising path toward alleviating this unsustainable process by automating the pipeline of designing DNN architectures [7].

In the early days, NAS was mainly dedicated to obtaining DNN models with low prediction error—the most important target of most deep learning tasks. More recently, however, the emerging application scenarios of deep learning (e.g., autonomous driving) have raised higher demands for DNN architecture designs [8]. Consequently, apart from the prediction error, there are also other design criteria to consider, such as the number of parameters/weights and the number of floating-point operations (FLOPs). Particularly, when deploying DNN models to edge computing devices, the hardware-related performance is equally important, such as inference latency and energy consumption. Hence, from an optimization point of view, NAS tasks of today are intrinsically multiobjective optimization problems (MOPs) aiming to achieve optimality of the multiple design criteria of the DNN architectures [9], [10].

Due to the black-box nature of NAS, the complex properties from the optimization point of view (e.g., discrete decision variables, multimodal and noisy fitness landscape, expensive and many objectives, etc.) have posed great challenges to conventional optimization methods driven by rigorous mathematical assumptions. Accordingly, it is intuitive to resort to evolutionary multiobjective optimization (EMO) algorithms,
which have well-established track records for handling complex MOPs [11], [12], [13]. Nonetheless, the progress of adopting EMO algorithms for NAS still lags well behind the progress of the general field of NAS research, due to the clear gap between the two fields caused by the following four main issues.

1. **Computational Cost**: Many baseline NAS methods require enormous computational cost [14], [15], [16], which is unfriendly to the researchers in academia; particularly, EMO algorithms often require a large number of function evaluations to run, such that the issue of expensive computational cost will become especially sharp.

2. **Algorithm Comparison**: Despite that the recent development has led to more efficient methods [17], [18], different NAS methods are not directly comparable to each other given the inherent differences in types of architectures (i.e., search spaces) and training procedures (i.e., fitness evaluations); hence, with the NAS methods in the current literature, it is very unlikely (if not impossible) to isolate the contribution of the optimization algorithm itself to the overall success of an NAS method [19].

3. **Problem Formulation**: The research in EMO is often dedicated to solving MOPs with specific complex characteristics (e.g., multiple modalities and many objectives), but there is still no general formulation and analysis of NAS from the optimization point of view, thus having hindered EMO researchers to look deeper into the problems.

4. **Running Environment**: Performing NAS usually requires expertise in deep learning—not only the experience in developing DNN models but also the knowledge in configuring the software/hardware environment; however, the conventional EMO researchers mainly focus on how to design efficient EMO algorithms, but not the environments to call/run the MOPs [20].

Among the four issues, issues 1 and 2 are not only for EMO but also for general NAS research. To address these two issues, the recent NAS literature has made some attempts to propose the so-called tabular NAS benchmarks [21], [22], [23]. The basic idea is to rely on an exhaustive evaluation of all attainable architectures for multiple repetitions, resulting in confined search spaces and toy-scale DNN architectures that are unrealistic for real-world deployments or downstream tasks. To this end, the concept of surrogate NAS benchmarks have been introduced [24]. It is expected that an accurate interpolation of the fitness landscape can be achieved on top of fix-budget training samples, thus extending NAS benchmarks to large search spaces with more than $10^{18}$ different DNN architectures. This recent progress in NAS benchmarks has led to a substantial reduction in the computational time required for NAS algorithm development and a significant improvement in the reproducibility of NAS research.

Despite the recent progress in addressing issues 1 and 2, the outcome architectures (from an NAS algorithm) still need to be retrained from scratch as the optimal weights are either not provided by or not available in the existing NAS benchmarks [21], [24]. Given a set of Pareto-optimal architectures returned by an EMO algorithm, such a retraining process itself can render the entire approach computationally prohibitive. Nonetheless, there is still little effort dedicated to addressing issues 3 and 4.

Generally speaking, the challenges of issues 3 and 4 are twofold: 1) from the research perspective, existing NAS benchmarks are merely provided as datasets, but not constructed as benchmark test problems on the basis of general formulation from the EMO point of view and 2) from an engineering perspective, existing NAS benchmarks still require installations/configurations of sophisticated deep learning software/hardware environments, e.g., GPU, TensorFlow, PyTorch, among others. Moreover, each of the existing NAS benchmarks merely covers one or two search spaces, which is far from sufficient for benchmark comparisons between EMO algorithms; however, if one hopes to test an algorithm over different benchmarks, he/she has to make repeated modifications to have the algorithm compatible to the complicated interfaces of each benchmark, thus making the development of NAS algorithms tedious and error-prone.

This article is dedicated to bridging the gap between EMO and NAS by addressing the four issues above. In summary, the main contributions are as follows.

1) We present a general formulation of NAS problems from the multiobjective optimization perspective, involving three categories of optimization objectives: a) the prediction error objective; b) the complexity-related objectives; and c) the hardware-related objectives. On top of this formulation, we demonstrate a series of complex characteristics of NAS problems, including multiple modalities, many objectives, noisy objectives, degenerated Pareto fronts (PFs), among others.

2) We provide a unified and comprehensive NAS benchmark, dubbed EvoXBench, comprising seven search spaces. Specifically, EvoXBench covers two types of architectures (convolutional neural networks and vision Transformers), two widely studied datasets (CIFAR-10 and ImageNet), six types of hardware (GPU, mobile phone, FPGA, among others), and up to six types of optimization objectives (prediction error, FLOPs, latency, among others).

3) We provide delicate engineering designs of EvoXBench to be user-friendly to EMO research. Specifically, EvoXBench can be easily installed without dependencies of deep learning software packages, such as Pytorch or TensorFlow; and it can be directly called in any programming language, such as Python, MATLAB, or Java; most importantly, running without GPU, EvoXBench provides instant feedback to MOEAs for real-time fitness evaluations.

4) On the basis of our EvoXBench, we generate two representative benchmark test suites, i.e., the C-10/MOP and the IN-1K/MOP, involving MOPs with up to six objectives. Furthermore, we conduct a series of experiments to study the properties of the test problems using...
In the remainder of this article, first, we provide necessary background information in Section II; second, we provide formal NAS problem formulation and analysis in Section III; third, we explain the design principles of our benchmark and test suite generation in Sections IV and V, respectively; fourth, we provide empirical evaluations in Section VI; and finally, conclusions and future studies are discussed in Section VII.

II. BACKGROUND

In this section, we provide a brief overview of the related concepts. The main notations used in this article are summarized in Table I.

A. Evolutionary Multiobjective Optimization

Generally, an MOP, involving more than one conflicting objective to be optimized simultaneously, can be formulated as

\[
\min_x F(x) = (f_1(x), f_2(x), \ldots, f_M(x))
\]

\[
s.t. \ x \in X, \ F \in Y
\]

where \( X \subseteq \mathbb{R}^D \) and \( Y \subseteq \mathbb{R}^M \) are known as the decision space and the objective space, respectively. Since \( f_1(x), \ldots, f_M(x) \) are often conflicting with each other, there is no single solution that can achieve optima on all objectives simultaneously; instead, the optima to such an MOP is often a set of solutions trading-off between different objectives, known as the Pareto optima. Specifically, the images of the Pareto optima are known as the Pareto set (PS) and the PF in the decision space and objective space, respectively.

In practice, the target of solving an MOP is to approximate the PS/PS with a limited number of candidate solutions. To this end, various EMO algorithms have been proposed during the past two decades. Generally, the EMO algorithms can be categorized into three categories: 1) dominance-based ones [26], [27], [28]; 2) decomposition-based ones [12], [29]; and 3) performance-indicator-based ones [30], [31]. Readers are referred to supplementary materials Section I-A for more details.

To assess the performance of various EMO algorithms in the literature, a number of challenging EMO benchmark test suites have also been designed to consider different complicated characteristics, e.g., the ZDT test suite [32], the DTLZ test suite [33], the WFG test suite [34], the MaF test suite [35], the LSNM test suite [36], etc. Undoubtedly, the EMO benchmark test suites play an important role in pushing the boundaries of EMO research, having promoted the ongoing delicate designs of EMO algorithms to face the new challenges of emerging benchmark test suites.

B. Neural Architecture Search

The goal of NAS is to automate the design of the architectures of DNN models by formulating it as an optimization problem [7]. As depicted in Fig. 1(a), a standard NAS pipeline involves three main components: 1) a search space that prescribes how a DNN architecture is represented; 2) a search algorithm for generating suitable architectures according to prespecified criteria; and 3) an evaluator to estimate the fitness of an architecture [19]. In the following, we provide a brief review of the literature related to NAS search spaces and refer readers to supplementary materials Section I-B for NAS search algorithms and fitness evaluators.

The architectural design of a DNN model can be decomposed into the design of: 1) the skeleton of the network (i.e., the depth and width of the network) and 2) the layer (i.e., the types and arrangements of the operators, such as convolution, pooling, etc.). Early work made attempts to design these two aspects simultaneously, resulting in DNN architectures with suboptimal performance [7], [14], [37]. Accordingly, recent efforts resort to only one aspect, and two types of search spaces have been proposed. Specifically, micro search spaces focus on designing a modular computational block (also known as a cell), which is repeatedly stacked to form a complete DNN architecture following a prespecified template [15], [17], [16], [18]; macro search spaces focus on designing the network skeleton while leaving layers to well-established designs [38], [39], [40]. A pictorial illustration is provided in Fig. 2.

On the other hand, micro search spaces confine the search to the inner configuration of a layer, leading to a considerable reduction in search space volume at the cost of structural optimization, leading to a considerable reduction in search space volume at the cost of structural

![Table I: Summary of Notations](image)

| Category | Notation | Description |
|----------|----------|-------------|
| Data     | \( D := \mathbb{R}^D \) | \( D_{train}, D_{valid}, D_{test} \) for train, validation, and test splits of a dataset |
| Hardware | \( h \in H \) | \( \Omega \) for a specific set of hardware device |
| Decision vectors | \( \omega(x) \) | \( \mathbb{R}^D \) for architecture decision variables and search space |
| Objectives | \( f^p \) | \( \mathbb{R}^D \) for model complexity, e.g., \# of parameters and FLOPs |
| Others | \( N \) | \( \mathbb{R}^D \) for population size |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
diversity among layers, which has been shown to be crucial for hardware efficiency [41]. On the other hand, despite compelling performance on hardware, macro search spaces are oftentimes criticized for producing architecturally similar DNN models [42].

C. Existing NAS Benchmarks

As one of the earliest tabular benchmarks, 1 NAS-Bench-101 [21] adopts a micro search space where approximately 423K unique cell structures are exhaustively evaluated three times on the CIFAR-10 dataset [25] for image classification. All results are stored in a TFRecord file, which is a binary file format designed specifically for TensorFlow—a dedicated software for machine learning with DNNs [43]. A subsequent tabular benchmark of NAS-Bench-201 [22], with a similar micro search space of 16K cell structures, extends the dataset to include the CIFAR-100 [25] and downsampled ImageNet 16 × 16 [44] datasets. Moving beyond tabular benchmarks, NAS-Bench-301 introduces the first surrogate-model-based benchmark on the CIFAR-10 dataset. With sophisticated regression models (e.g., graph isomorphism network [45] and tree-based gradient boosting methods [46], [47]) on top of regression models (e.g., graph isomorphism network [45] and benchmark on the CIFAR-10 dataset. With sophisticated NAS-Bench-301 introduces the first surrogate-model-based evaluator and stored as a tabular dataset a priori.

DNN models [42].

Concurrently, a steady stream of follow-up work have been reported, extending NAS benchmarks to include: 1) a macro search space for finding the optimal channels of each layer in a DNN (i.e., NATS-Bench [23]); 2) hardware-related performance (HW-NAS-Bench [49]); or 3) other application domains, such as natural language processing [50], speech recognition [51], transfer learning [52], among others.

III. PROBLEM FORMULATION

Given a target dataset $D = \{D_{trn}, D_{vld}, D_{std}\}$ and target hardware device set $H = \{h_1, \ldots, h_{|H|}\}$, without loss of generality, an NAS task can be formulated as a multiobjective optimization problem

$$
\begin{align*}
\min_{x} & \quad F(x) = \left(f^c(x; \mathbf{w}^c(x)), f^e(x), f^H(x) \right) \\
\text{s.t.} & \quad \mathbf{w}^c(x) \in \text{argmin} \ L_{trn}(x; \mathbf{w}), x \in \Omega 
\end{align*}
$$

where $\Omega$ is the search space, $x$ and $\mathbf{w}(x)$ denote an encoded network architecture (i.e., decision vector) and the corresponding weights of the decoded network, respectively, and $\mathbf{w}^c(x)$ denotes the optim weights achieving the minimal loss on the training set $D_{trn}$, $f^c$ denotes the objective function indicating the prediction error of the model, and $f^e$ and $f^H$ denote the objective functions indicating the performance related to model complexity and hardware devices, respectively.

In practice, the objective functions can be formulated in various ways. For example, to formulate $f^e$, the most straightforward approach is to associate it with the validation loss on $D_{vld}$; to formulate $f^c$, we usually refer to the number of parameters (i.e., weights) and FLOPs, respectively; to formulate $f^H$, the most commonly considered metrics are hardware latency and hardware energy consumption. Besides, since it can be computationally prohibitive to fully train a model on $D_{trn}$ to obtain the exact validation loss on $D_{vld}$ for $f^e$, surrogate-modeling approaches are also adopted.

Based on the formulation above, the following subsections will elaborate on the complex characteristics of NAS tasks from the optimization point of view.

A. Multimodal Fitness Landscapes

The fitness landscape of an optimization problem depicts how fitness values change over the decision space. Specifically, a multimodal landscape refers to the case involving multiple optimum solutions (also known as multimodality) having very close (or even the same) fitness values [54]. When searching over a multimodal fitness landscape, the target is often to find the multiple optimum solutions once, such that the decision-maker is able to choose among those solutions according to personal preferences. Particularly, there are emerging research interests in studying multimodal MOPs in the EMO community [55].

In the context of NAS, it is very likely that different network architectures can lead to very close (or even the same) prediction accuracy or hardware-related metrics such as latency. As evidenced in Fig. 3, for example, the fitness landscapes of $f^e$ on search spaces NB101 and NB201 contain a number of optima having very close fitness values (i.e., prediction errors); correspondingly, the solutions having very close fitness values could represent very different architectures.

B. Noisy Objectives

In practical optimization problems, a solution is often evaluated through stochastic simulations, physical experiments, or even interactions with users. As a result, the outputs for repeated evaluations at the same decision vector are
not deterministic. From an optimization point of view, we characterize such an evaluation process as a noisy objective [56], [57]. Evolutionary computation is believed to be well suited for tackling noisy objectives given its population-based nature and randomized search heuristics. Accordingly, developing effective algorithms for solving noisy optimization problems has always been an active research topic in the EMO community [58], [59].

In the context of NAS, the evaluation of $f^e$ (prediction error) of an architecture is almost always noisy due to various stochastic components involved in training the corresponding weights of the architecture, such as randomized data augmentation and loading order. Fig. 4 provides examples of visualizing the noises in evaluating $f^e$ on NB101 and

![Fig. 3. Fitness landscapes of $f^e$ (i.e., prediction error) on (a) NB101 and (b) DARTS. We project the original high-dimensional decision space to 2-D latent space via t-SNE [53]. We random sample 10K solutions from each search space and average $f^e$ within each small area. The top-performing solutions are highlighted in red. (c)–(e) are three solutions with similar $f^e$ but different decision vectors (i.e., architectures) sampled from the NB101 search space.](image)

![Fig. 4. Mean $f^e$ (prediction error) of the top 1000 solutions (according to $f^e$) on (a) NB101 and (b) NB201 search spaces from three repetitions. The standard deviations (i.e., noise) of $f^e$ are visualized with the shaded error bars.](image)

Fig. 3. Fitness landscapes of $f^e$ (i.e., prediction error) on (a) NB101 and (b) DARTS. We project the original high-dimensional decision space to 2-D latent space via t-SNE [53]. We random sample 10K solutions from each search space and average $f^e$ within each small area. The top-performing solutions are highlighted in red. (c)–(e) are three solutions with similar $f^e$ but different decision vectors (i.e., architectures) sampled from the NB101 search space.

![Fig. 4. Mean $f^e$ (prediction error) of the top 1000 solutions (according to $f^e$) on (a) NB101 and (b) NB201 search spaces from three repetitions. The standard deviations (i.e., noise) of $f^e$ are visualized with the shaded error bars.](image)

Fig. 4. Mean $f^e$ (prediction error) of the top 1000 solutions (according to $f^e$) on (a) NB101 and (b) NB201 search spaces from three repetitions. The standard deviations (i.e., noise) of $f^e$ are visualized with the shaded error bars.

Fig. 5. Parallel coordinates of nondominated solutions of architectures sampled from the NB021 search space with hardware device of Eyeriss (i.e., $\Omega = MNV3$, $\mathcal{H} = \{h_1 = \text{Eyeriss}\}$). The six optimization objectives: $f^e$: prediction error, $f_{c1}^e$: number of parameters (M), $f_{c2}^e$: number of FLOPs (M), $f_{H1}^e$: latency (ms), $f_{H3}^e$: energy (mJ), and $f_{H3}^e$: arithmetic intensity (ops/byte).

NB201 search spaces, respectively. Furthermore, hardware-related objectives also critically depend on the operating conditions of the hardware, such as current loading, ambient temperature, etc. Hence, the evaluation of $f^H$ is also subject to high variance.

C. Many Objectives

The number of objectives in an MOP is an important character to be considered in EMO. Particularly, if the number of objectives increases to more than three, an MOP is known to be a many-objective optimization problem (MaOP), posing great challenges to the EMO algorithms [60]. On the one hand, it is challenging to obtain the sparsely distributed candidate solutions when considering both convergence and diversity in the high-dimensional objective space; on the other hand, it is challenging to perform decision making when trading-off among many objectives.

As formulated above, an NAS task can involve up to a number of $1 + M_c + \sum_{i=1}^{\vert \mathcal{H} \vert} M_h^i$ objective functions to be optimized simultaneously, where $M_c$ denotes the number of objective functions related to model complexity, and $M_h^i$ denotes the number of objective functions related to the $i$th hardware device in $\mathcal{H}$. As exemplified in Fig. 5, an NAS task on the search space of NB201 consists of six objectives with $M_c = 2$ and $M_h^1 = 3$.

D. Badly Scaled Objectives

Ideally, the objective functions of an MOP should be scaled to the same (or similar) values such that EMO algorithms could work effectively. This is particularly important to the decomposition-based EMO algorithms whose performances largely rely on the predefined weight/reference vectors. Without any preference or a priori knowledge, the weight/reference vectors are uniformly sampled from a unit hyper-plane/sphere, assuming that the objective functions of the MOP to be solved are also well normalized. In practice, however, the objective functions can be of very different scales and there is no way to formulate them in a normalized manner. An MOP of such a character is often known to be badly scaled [61].
Considering the three types of objective functions in a general formulation of an NAS task, the corresponding MOPs are intrinsically badly scaled. As exemplified in Fig. 6, the prediction error ($f^e$) falls into the range of $[0,1]$; by contrast, since the complexity-related objectives ($f^c$) and hardware-related objectives ($f^H$) are formulated to indicate different physical quantities, their objective values are of very different scales. Moreover, it is difficult to normalize these objective values into the same scale due to the unknown properties of various search spaces and hardware devices.

E. Degenerate Pareto Fronts

In multiobjective optimization, an important assumption is that the objectives to be optimized are often in conflict, such that there does not exist a single solution achieving optima on all the objectives. Alternatively, an EMO algorithm aims to find a set of solutions as an approximation to the PF. As formulated above, the PF of an $m$-objective MOP is an $m-1$-D manifold iff all the objectives are conflicting with each other. In practice, however, the model may not always hold since some objectives of an MOP may be positively correlated to each other, thus leading to a degenerate PF.

As formulated above, there are generally three types of objectives in an NAS problem: 1) the error-related objective $f^e$; 2) the complexity-related objectives $f^c$; and 3) the hardware-related objectives $f^H$. Normally, $f^e$ is conflicting with $f^c$ and $f^H$, respectively, i.e., a model with lower prediction error usually has higher complexity and lower hardware performance. By contrast, however, the objectives in $f^c$ or $f^H$ may not be fully conflicting among themselves. As exemplified in Fig. 7(a), $f^e$ is conflicting with $f^H$, respectively, while $f^c_1$ and $f^c_2$ are positively correlated, thus leading to a degenerate PF (i.e., the 1-D curve) in the three objective space.

IV. BENCHMARK DESIGN

In this section, we introduce the overall pipeline of EvoXBench, followed by detailed design principles of each main component.

A. Overview

In order to facilitate seamless communication, an API layer is designed to handle: 1) the reception of the candidate decision vectors generated by an EMO algorithm and 2) the feedback of the fitness values evaluated by our EvoXBench. As shown in Fig. 8, given a candidate decision vector received by the API layer, EvoXBench first passes it through the search space module to create its corresponding architecture; then, the architecture is processed by the fitness evaluator module; finally, the optimization objectives are routed back to the API layer for output. For EMO algorithms implemented in non-Python programming languages (e.g., MATLAB or Java), the communication is routed through an extra remote procedure call (RPC) module built on top of the widely used transmission control protocol (TCP).

B. Design of Search Spaces

As formulated in (2), a search space (also known as decision space) $\Omega$ defines all attainable solutions to an optimization algorithm. In general, every search space follows the schema (outlined in Algorithm 1 from supplementary materials). Specifically, a string that defines a DNN architecture is referred to as a phenotype, from which an actual DNN model can be created; an integer-valued vector is referred to as a genotype on which genetic operators, such as crossovers and mutations.
are carried out; and the interfaces between genotypes and phenotypes are referred to as \( f_e \) and \( f_c \), respectively.

In \textit{EvoXBench}, we currently consider seven search spaces for image classification on CIFAR-10 and ImageNet datasets, covering both convolutional DNNs and Transformers from micro and macro search spaces. A summary of these search spaces is provided in Table II, and visualizations of architectures from these search spaces are provided in Fig. 9.

It is worth noting that apart from the existing seven search spaces, any other user-specific search space can also be easily incorporated into \textit{EvoXBench} by following a routine pipeline. Due to the page limit, readers are referred to Section IV in the supplementary materials for more details.

C. Design of Fitness Evaluator

As formulated in (2), we consider three categories of objectives, i.e., the prediction error objective \( f^p \), the model complexity-related objectives \( f^c \), and hardware efficiency \( f^h \). Dedicated methods are devoted to handling these objectives efficiently and reliably.

1) Evaluation of \( f^p \): As illustrated in Section III-B, \( f^p \) is a noisy optimization objective. To simulate the noise in evaluating \( f^p \), the following two strategies are employed.

1) For NB101, NB201, and NATS search spaces, we leverage the exhaustive evaluation results provided by the original papers [21], [22], [23], [49], where all solutions (architectures) are thoroughly trained from scratch with three repetitions. On top of these results, we construct a unified database solely based on Python. In other words, the \( f^p \) values of solutions from these search spaces can be queried via a canonical interface without configurations of sophisticated software, such as TensorFlow or PyTorch. Thereafter, each evaluation value of \( f^p \) is randomly selected from the three repetitions of the corresponding evaluation results stored in the database.

2) Considering the total volume of the DARTS, ResNet50, Transformer, and MNV3 search spaces, we resort to surrogate models for evaluating \( f^p \). We choose a multilayer perceptron (MLP) as our surrogate model in this work. Apart from its well-established track record for predicting the \( f^p \) values of DNNs accurately [64], [66], [67], implementing an MLP (i.e., a loop of matrix multiplications and additions) is straightforward and canonical in most programming languages without additional dependencies. Furthermore, we construct a pool of MLPs from \( k \)-fold cross-validation of the training data, where \( k \) is set to ten in this work. Thereafter, each evaluation value of \( f^p \) is predicted by a single MLP randomly selected from the pool.

For generating samples (i.e., variable-objective pairs) to train an MLP, we utilize the \textit{supernet} idea that has emerged as a standard technique in state-of-the-art (SotA) NAS methods [42]. Especially, first, we follow the progressive shrinking algorithm [64] to train a supernet from which the optimal weights of a candidate architecture solution \( w(x) \) (in (2)) can be directly inherited without the costly iterations of SGD;\(^2\) then, we use the weights provided by the trained supernet to evaluate the \( f^p \) values of the samples. As opposed to generating samples by independently training every one of them from scratch, this pipeline offers two appealing properties: 1) it is more scalable to a large number of samples which is crucial for learning an accurate approximation\(^3\) and 2) the post-search retraining of the obtained architectures can be avoided as the corresponding weights are readily available from the supernet.

A pictorial illustration of the \( f^p \) evaluators in \textit{EvoXBench} is provided in the top part of Fig. 10. And the performance

\(^2\)SGD is a standard method for solving the inner optimization problem in (2), i.e., \( \min \mathcal{L}_{trn}(x; \omega) \).

\(^3\)Assuming that the training cost of a large number of sample architectures, 60K in our case, is much greater than the training cost of a single supernet.
and NATS search spaces. These f are the macro search spaces. Hence, we can exhaustively evaluate layer variations to enumerate, respectively, for micro and search spaces (i.e., ResNet50, Transformer, and MNV3).

The efficiency of a network can be decomposed into the accumulative efficiency of the participating operations or layer variations. For the DARTS, ResNet50, Transformer, and MNV3 search spaces, we opt for the loop-up table route—i.e., a standard approach among existing NAS methods where an exhaustive exploration of the search space is not possible [39], [49].

For the DARTS, ResNet50, Transformer, and MNV3 search spaces, we visualize the correlation between real measurements and approximations by our surrogate model for prediction error \( f_e \), look-up tables for the number of parameters \( f_p \) and FLOPs \( f_f \), from left to right. The mean absolute error (MAE) and Kendall rank correlation values are annotated in every subfigure. (a) DARTS search space. (b) ResNet50 search space. (c) MNV3 search space.

of the adopted surrogate model, an ensemble of MLPs, is provided in Fig. 11.

2) Evaluation of \( f_e \) and \( f_H \): Similar to the evaluation of \( f_e \), we leverage the exhaustive evaluation history provided by the original papers [21], [22], [23], [49] for NB101, NB201, and NATS search spaces. These \( f_e \) and \( f_H \) values are, again, stored in a unified database along with \( f_c \) values.

For the DARTS, ResNet50, Transformer, and MNV3 search spaces, we opt for the loop-up table route—a standard approach among existing NAS methods where an exhaustive exploration of the search space is not possible [39], [49]. A pictorial overview is provided in the bottom part of Fig. 10. In general, on the one hand, the efficiency of a cell structure can be decomposed into the accumulative efficiency of the participated operations for a micro search space (i.e., DARTS); while on the other hand, the efficiency of a network can be decomposed into the accumulative efficiency of each layer for macro search spaces (i.e., ResNet50, Transformer, and MNV3).

Under such assumptions, there exist limited operations and layer variations to enumerate, respectively, for micro and macro search spaces. Hence, we can exhaustively evaluate the \( f_e \) and \( f_H \) values of all operations and layer variations. Afterward, the efficiency of any given operation or layer variation can be queried from a look-up table with a designated key. Then, the \( f_e \) and \( f_H \) values of an architecture solution can be obtained by summing over all operations or layer variations.

D. Design of RPC Module

As depicted in Fig. 8, EMO algorithms written in Python can call EvoXBench directly, while queries from EMO algorithms in other programming languages are processed through the RPC module. For compatibility reasons, the communication functionalities of RPC are built upon the TCP. Technically, it allows programs written in almost any programming language on the operating system to call EvoXBench via localhost. Furthermore, by setting up EvoXBench once on a central server, it allows additional users to use EvoXBench remotely as a Web service without any local installation/configuration.

More specifically, after establishing a successful TCP connection, EMO algorithms (as the users) and EvoXBench communicate by exchanging JSON strings. Since this protocol itself is stateless, we assign an integer string to every object created in EvoXBench as its unique identifier. This identifier needs to be included as a part of the JSON strings to facilitate the communication between users and EvoXBench.

As depicted in Table III, we empirically demonstrate that real-time feedback of fitness values is archived by EvoXBench via the RPC module. Specifically, the evaluation of an architecture in EvoXBench is facilitated by either querying a database or a surrogate model. On the one hand, querying a database is intrinsically fast; on the other hand, the surrogate model used in EvoXBench is a simple yet effective three-layer perceptron. Additionally, we batch all evaluations in a matrix to further boost the speed. Readers may refer to supplementary materials for more technical details.

| Search space | Query method | Python | MATLAB | Java |
|--------------|--------------|--------|--------|------|
| NB101        | Database     | 0.1139 ± 0.013 | 0.1716 ± 0.031 | 0.1970 ± 0.036 |
| MNV3         | Surrogate    | 0.0380 ± 0.002 | 0.0528 ± 0.018 | 0.0574 ± 0.020 |

V. Benchmarks Test Suite Generation

On the basis of EvoXBench, specific test suites (i.e., collections of test instances/problems) can be generated by combining search spaces and their supported objectives. For the purpose of demonstration, we generate two tailored multiobjective NAS test suites for image classification on datasets CIFAR-10 and ImageNet, respectively. Following the formulations in Section III, the test instances are defined by specifying the search space, the hardware, and the metrics for measuring model complexity and hardware efficiency. In the remaining of this section, we present these two test suites in detail.

A. C-10/MOP Test Suite

As summarized in Table IV, there are nine instances in C-10/MOP tailored for image classification on CIFAR-10, considering various search spaces and hardware devices. The test instances are arranged in the order of ascending number of objectives, i.e., from two to eight objectives. For most test instances, we consider only one targeted hardware (i.e., GPUs or Eyeriss [68]) except C-10/MOP7 where both hardwares are considered simultaneously.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
The properties of the test instances are summarized in Table V in correspondence with the characters illustrated in Section III. In general, the fitness landscapes of prediction error ($f^*$) are noisy and multimodal, and the PFs are degenerated for most test instances. Since the fitness landscape of C-10/MOP1 to C-10/MOP7 are completely known via exhaustive samples, the ranges of all objectives are known a priori and thus can be properly scaled. By contrast, the objectives of C-10/MOP8 and C-10/MOP9 remain un-normalized (i.e., badly scaled) as the ranges cannot be estimated via exhaustive samples.

### Table IV: Definition of the Proposed C-10/MOP Test Suite

| Problem   | $\Omega$ | $D$ | $M$ | Objectives                  |
|-----------|----------|-----|-----|-----------------------------|
| C-10/MOP1 | NB01     | 26  | 2   | $f^*$, $f_1^*$              |
| C-10/MOP2 | NB01     | 26  | 3   | $f^*$, $f_1^*$, $f_2^*$    |
| C-10/MOP3 | NATS     | 5   | 3   | $f^*$, $f_1^*$, $f_2^*$    |
| C-10/MOP4 | NATS     | 5   | 4   | $f^*$, $f_1^*$, $f_2^*$, $f_3^*$ |
| C-10/MOP5 | NB201    | 6   | 5   | $f^*$, $f_1^*$, $f_2^*$, $f_3^*$, $f_4^*$ |
| C-10/MOP6 | NB201    | 6   | 6   | $f^*$, $f_1^*$, $f_2^*$, $f_3^*$, $f_4^*$, $f_5^*$ |
| C-10/MOP7 | DARTS    | 8   | 6   | $f^*$, $f_1^*$, $f_2^*$, $f_3^*$, $f_4^*$, $f_5^*$, $f_6^*$ |
| C-10/MOP8 | DARTS    | 32  | 2   | $f^*$, $f_1^*$, $f_2^*$    |
| C-10/MOP9 | DARTS    | 32  | 3   | $f^*$, $f_1^*$, $f_2^*$, $f_3^*$ |

1 indicates that $f^*$ (validation error) is based on surrogate modeling.

### Table VI: Property of Test Instances in C-10/MOP

| Problem | Multimodal | Many objectives | Noisy objectives | Badly-scaled | Degenerated PF |
|---------|------------|----------------|------------------|--------------|---------------|
| C-10/MOP1 | ✓          | ✓              | ✓                | ✓            | ✓             |
| C-10/MOP2 | ✓          | ✓              | ✓                | ✓            | ✓             |
| C-10/MOP3 | ✓          | ✓              | ✓                | ✓            | ✓             |
| C-10/MOP4 | ✓          | ✓              | ✓                | ✓            | ✓             |
| C-10/MOP5 | ✓          | ✓              | ✓                | ✓            | ✓             |
| C-10/MOP6 | ✓          | ✓              | ✓                | ✓            | ✓             |
| C-10/MOP7 | ✓          | ✓              | ✓                | ✓            | ✓             |
| C-10/MOP8 | ✓          | ✓              | ✓                | ✓            | ✓             |
| C-10/MOP9 | ✓          | ✓              | ✓                | ✓            | ✓             |

B. IN-1K/MOP Test Suite

The IN-1K/MOP test suite also comprises nine test instances. Since the search spaces of ResNet50 and Transformer are resource intensive and thus unsuitable for efficient hardware deployment, we do not consider hardware-related objectives as architectures from them. As a result, most test instances are multiobjective problems except IN-1K/MOP9 where an additional objective of the latency on a mobile phone is considered. A summary of IN-1K/MOP test suite is provided in Table VI.

The properties of the test instances of IN-1K/MOP are summarized in Table VII. Given the volume of the involved search spaces, it is impossible to evaluate every attainable solution. Accordingly, for each test instance, the fitness landscape of is approximated via a surrogate model, and the objectives are badly scaled (i.e., without normalization). Note that the objectives can be normalized to the bounds derived from the sampled architectures (for building the surrogate models). However, due to the fact that the sampled architectures may not exactly cover the full range of the search spaces, the “normalized” objectives (according to the bounds derived from the sampled architectures) may not be strictly between zero and one. Accordingly, we opt for leaving the objectives as un-normalized. The character of multimodality for all test instances is verified on the basis of extensive optimization using a large number of fitness evaluations.

### Table VII: Property of Test Instances in IN-1K/MOP

| Problem   | $\Omega$ | $D$ | $M$ | Objectives                  |
|-----------|----------|-----|-----|-----------------------------|
| IN-1K/MOP1 | ResNet50 | 25  | 2   | $f^*$, $f_1^*$              |
| IN-1K/MOP2 | ResNet50 | 25  | 3   | $f^*$, $f_1^*$, $f_2^*$    |
| IN-1K/MOP3 | ResNet50 | 25  | 3   | $f^*$, $f_1^*$, $f_2^*$, $f_3^*$ |
| IN-1K/MOP4 | Transformer | 34  | 2   | $f^*$, $f_1^*$              |
| IN-1K/MOP5 | Transformer | 34  | 3   | $f^*$, $f_1^*$, $f_2^*$    |
| IN-1K/MOP6 | Transformer | 34  | 3   | $f^*$, $f_1^*$, $f_2^*$, $f_3^*$ |
| IN-1K/MOP7 | MNV3     | 21  | 2   | $f^*$, $f_1^*$              |
| IN-1K/MOP8 | MNV3     | 21  | 3   | $f^*$, $f_1^*$, $f_2^*$    |
| IN-1K/MOP9 | MNV3     | 21  | 4   | $f^*$, $f_1^*$, $f_2^*$, $f_3^*$ |

All $f^*$ (validation errors) are based on surrogate modeling.

### VI. Experimental Study

In this section, we provide an experimental study using the two test suites proposed in the previous section. First, we explain the experimental setup, including the selected EMO algorithms and the performance metric; then, we present experimental results followed by a discussion of the observations. Constrained by space, readers are referred to supplementary materials for implementation details.

A. Experimental Settings

A plethora of algorithms have been proposed in the EMO literature for solving MOPs (involving two/three objectives) or MaOPs (involving more than three objectives).

Accordingly, we select six representative algorithms from each category as introduced in Section II-A, including NSGA-II [11] (dominance-based), IBEA [30] (indicator-based), MOEA/D [12] (decomposition-based), NSGA-III [69] (dominance-based), HypE [70] (indicator-based), and RVEA4 [71] (decomposition-based), where the first three were

4We use the version with the reference vector regeneration strategy, which is also known as the RVEA*.
Fig. 12. Nondominated solutions obtained by each algorithm on (a) C-10/MOP1 and (b) C-10/MOP6. We select the run associated with the median HV value. For each row, the subfigures correspond to NSGA-II, IBEA, MOEA/D, NSGA-III, HypE, and RVEA, respectively.

Table VIII

| M | \( (H_1, H_2) \) | N |
|---|---|---|
| 2 | (99, 0) | 100 |
| 3 | (13, 0) | 105 |
| 4 | (7, 0) | 120 |
| 5 | (5, 0) | 126 |
| 6 | (4, 1) | 132 |
| 8 | (3, 2) | 156 |

C. Results

In the following, we present results achieved by each of the considered algorithm on the two test suites.

1) Results on C-10/MOP: Table IX summarizes the statistical values of the HV metric achieved by the six algorithms. In general, we can observe that none of the six algorithms can effectively solve all instances in the C-10/MOP test suite. More specifically, on MOPs with two or three objectives (i.e., C-10/MOP1, C-10/MOP2, C-10/MOP8, and C-10/MOP9), we can observe that NSGA-II consistently outperforms other algorithms, except C-10/MOP3 where IBEA performs slightly better, while on MaOPs involving more than three objectives (i.e., C-10/MOP4–C-10/MOP7), we can observe that HypE yields generally the best performance. For further observations, we visualize the final nondominated solutions obtained by each algorithm in the median run on two typical test instances in Fig. 12(a) and (b) for C-10/MOP1 and C-10/MOP6, respectively. In the following, we provide some discussions based on the observations made from Fig. 12.

C-10/MOP1 is a problem with a simple bi-objective convex PF; nevertheless, given its multimodal and noisy fitness landscape, the problem is still nontrivial to solve. First, we can observe that all algorithms fail to converge to the low-\( f_1 \) regime (top-left corner), indicating the challenges for obtaining nondominated solutions with low \( f_1 \) (i.e., \( f_{e} \); prediction error). Second, we can observe that indicator-based algorithms (i.e., IBEA and HypE) perform significantly worse than other peer methods. Third, both NSGA-II and NSGA-III achieve the best performance, where NSGA-II has slightly better coverage of the low-\( f_2 \) regime while NSGA-III has a slightly better converge of the low-\( f_1 \) regime.

C-10/MOP6 is relatively more complex with six (but partially correlated) objectives. The objective function of \( f_1 \) (i.e., \( f_e \); prediction error) is also multimodal and noisy, but the number of variables is small, i.e., six. First, we can observe that objective 2 (\( f_1^c \); No. of parameters) and objective 3 (\( f_2^c \); No. of FLOPs), objective 4 (\( f_1^h \); latency on Eyeriss), and objective 5 (\( f_2^h \); energy consumption on Eyeriss) are correlated, thus leading to a degenerated PF. Second, we can observe that the decomposition-based algorithms (i.e., MOEA/D and RVEA) are significantly worse than those with classic ones for solving MOPs and the rest were tailored for solving MaOPs.

All experiments are conducted on PlatEMO [20]—an EMO algorithm library implemented in MATLAB. The population size is set in correspondence with the number objectives, as shown in Table VIII. We perform 31 independent runs for each algorithm on each test instance using 10,000 fitness evaluations, and the statistical results are compared using Wilcoxon rank sum test.

B. Performance Indicator

In this article, we adopt hypervolume (HV) to quantitatively compare the performance among the considered algorithms. Let us denote \( y_{ref} = (y_1, \ldots, y_m) \) as a reference point dominated by all Pareto-optimal solutions in the objective space, and \( Y \) as the Pareto-front approximated by an algorithm. The HV value of \( Y \) (with respect to \( y_{ref} \)) is the volume of the region dominating \( y_{ref} \) and dominated by \( Y \).

Specifically, we follow two rules to set the reference point for calculating HV: 1) for problems derived from search spaces that are exhaustively evaluated (i.e., \( \Omega = \{NB101, NB201, NATS\} \)), we set \( y_{ref} \) to the nadir point since the actual PF is available and 2) for problems derived on the basis of surrogate models (i.e., \( \Omega = \{DARTS, MNV3, ResNet50, Transformer\} \)), we set \( y_{ref} \) to the worst point among the samples collected for training the surrogate models. Readers are referred to Table A.I in the supplementary materials for the specific reference point for each test instance.
Table IX

Statistical Results (Median and Standard Deviation) of the HV Values on C-10/MOP Test Suite. The Best Results of Each Instance Are in Bold

| Problem   | NSGA-II | IBEA  | MOEA/D | NSGA-III | HypE   | RVEA   |
|-----------|---------|-------|--------|----------|--------|--------|
| C-10/MOP1 | 0.9367  | 0.8627| 0.9069 | **0.9379** | 0.8488 | 0.9297 |
| C-10/MOP2 | **0.9176** | 0.8341| 0.8727 | 0.7108 | 0.7879 | 0.9091 |
| C-10/MOP3 | 0.8335  | 0.8533| 0.8000 | 0.8083 | 0.8183 | 0.8262 |
| C-10/MOP4 | 0.7656  | 0.7941| 0.7277 | 0.7633 | 0.7692 | 0.7544 |
| C-10/MOP5 | 0.7127  | 0.7094| 0.6721 | 0.7040 | **0.7569** | 0.7521 |
| C-10/MOP6 | 0.7404  | 0.7302| 0.6807 | 0.6387 | 0.7743 | 0.7549 |
| C-10/MOP7 | 0.5696  | 0.5892| 0.5108 | 0.5417 | **0.6322** | 0.6122 |
| C-10/MOP8 | 0.9769  | 0.9751| 0.8870 | 0.9741 | 0.9486 | 0.9185 |
| C-10/MOP9 | **0.9630** | 0.9609| 0.7633 | 0.9579 | 0.9253 | 0.8952 |

- † indicates a method achieving significantly better performance.
- ‡ indicates a method achieving similar performance as the best-performing method.
- ‡ indicates a method achieving significantly worse performance.

Table X

Statistical Results (Median and Standard Deviation) of the HV Values on IN-1K/MOP Test Suite. The Best Results of Each Instance Are in Bold

| Problem   | NSGA-II | IBEA  | MOEA/D | NSGA-III | HypE   | RVEA   |
|-----------|---------|-------|--------|----------|--------|--------|
| IN-1K/MOP1| 0.9299  | 0.9246| 0.7447 | 0.9196  | **0.9020** | 0.7662 |
| IN-1K/MOP2| **0.8849** | 0.8833| 0.5239 | 0.8799  | 0.8592 | 0.2038 |
| IN-1K/MOP3| 0.7945  | 0.8187| 0.6063 | 0.7881  | 0.8079 | 0.7382 |
| IN-1K/MOP4| 0.9930  | 0.9787| 0.6224 | 0.9882  | 0.9311 | 0.3572 |
| IN-1K/MOP5| 0.9795  | 0.9711| 0.6145 | 0.9908  | 0.9311 | 0.3416 |
| IN-1K/MOP6| 0.9832  | 0.9761| 0.1570 | 0.9541  | 0.8971 | 0.6495 |
| IN-1K/MOP7| **0.9049** | 0.8895| 0.6324 | 0.8843  | 0.8687 | 0.8119 |
| IN-1K/MOP8| 0.6884  | 0.7267| 0.0533 | 0.6989  | 0.7135 | 0.5918 |
| IN-1K/MOP9| 0.5783  | 0.6514| 0.1416 | 0.5956  | 0.6269 | 0.4946 |

- † indicates a method achieving significantly better performance.
- ‡ indicates a method achieving significantly worse performance.

Fig. 13. Nondominated solutions obtained by each algorithm on (a) IN-1K/MOP1 and (b) IN-1K/MOP8. We select the run associated with the median HV value. For each row, the subfigures correspond to NSGA-II, IBEA, MOEA/D, NSGA-III, HypE, and RVEA, respectively.

2) Results on IN-1K/MOP: This test suite comprises primarily bi-/three-objective MOPs except for the last one—IN-1K/MOP9 with four objectives. The statistical values of the HV metric achieved by the six algorithms are summarized in Table X. In general, similar to the results of the previous test suite, we can observe that none of the six algorithms can effectively solve all problems in IN-1K/MOP. In particular, on bi-objective problems (i.e., IN-1K/MOP1, IN-1K/MOP2, IN-1K/MOP4, IN-1K/MOP5, and IN-1K/MOP7), NSGA-II performs consistently better than other algorithms; by contrast, on problems with more than two objectives (i.e., IN-1K/MOP3, IN-1K/MOP8, and IN-1K/MOP9), IBEA performs the best among six algorithms. Additionally, without an explicit normalization mechanism, the decomposition-based algorithms (i.e., MOEA/D and RVEA) perform substantially worse than the others. For further analysis, we plot the nondominated solutions achieved by each algorithm in the median run on two typical test instances in Fig. 13(a) and (b) for IN-1K/MOP1 and IN-1K/MOP8, respectively. In the following, we provide some discussions based on the observations made from Fig. 13.

IN-1K/MOP1 is a bi-objective problem of simultaneous minimization of prediction error ($f_e$) and No. of parameters ($f_1$). In addition to the multimodal and noisy purely Pareto-dominance-based (i.e., NSGA-II) or indicator-based (i.e., IBEA and HypE) ones.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
fitness landscape, another primary challenge of this problem is that the two objectives are of different scales: the objective of $f_e$ falls into the range of $[0, 1]$, while the objective of $f_h$ is in the range of millions. Consequently, as depicted in Fig. 13(a), we can observe that both MOEA/D and RVEA fail to converge to the PF. By contrast, NSGA-III achieves comparable performance to indicator-based algorithms (i.e., IBEA and HypE) owing to its internal normalization mechanism. Similar observations can be made on IN-1K/MOP8 test datasets, seven search spaces, three types of hardware, and EMO algorithms to run efficiently without the requirements of GPUs.

D. Further Discussion

Fig. 14. Architectures obtained by NGS-II on C-10/MOP5 in the run associated with the median HV value. We visualize architectures from three different tradeoff subsets: (a) subset preferred for prediction error and model complexity, i.e., nondominated considering prediction error ($f_e$), No. of parameters ($f_p$), and FLOPs ($f_c$); (b) subset preferred for prediction error and hardware efficiency, i.e., nondominated considering prediction error ($f_e$), GPU latency ($f_{hp}$), and energy ($f_{he}$); and (c) subset striking a balance between prediction error, model complexity, and hardware efficiency. (d) We also provide the frequency of the operators (i.e., decision variables of C-10/MOP5) used by these three subsets, respectively, (from left to right).

In the meantime, the results of these test suites also provide valuable feedback on hardware devices. For instance, as depicted in Fig. 12(b), the inference latency, on the one hand, is correlated [objective 4 in Fig. 12(b)] with the energy consumption [objective 5 in Fig. 12(b)] on Eyeriss, i.e., a DNN with a faster inference speed tends to draw less energy as well; on the other hand, tradeoffs exist between the inference speed/energy consumption and the arithmetic intensity (objective 6 in Fig. 12(b); defined as operations per unit memory traffic).

As demonstrate above, running EMO algorithms on EvoXBench provides rich results worthy of deep investigations. Beyond the main target of NAS for automating hardware-related design/deployment of DNNs, the empirical observations can be also useful in hardware designs.

VII. CONCLUSION

This article was devoted to bridging the gap between EMO and NAS. First, we provided a general multiobjective formulation of NAS together with the analyses of complex characteristics from the optimization point of view. Then, we presented an end-to-end pipeline—EvoXBench, to streamline the generation of benchmark test problems for EMO algorithms to run efficiently without the requirements of GPUs or sophisticated software such as Pytorch/TensorFlow. Next, we initiated two test suites comprising two image classification datasets, seven search spaces, three types of hardware,

5The lines between objective 4 and objective 5 are in parallel.

6The lines between objectives 5 and 6 are in a crisscross pattern.
REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2012, pp. 1–9.

[2] I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to sequence learning with neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 27, 2014, pp. 1–9.

[3] D. Silver et al., “Mastering the game of go without human knowledge,” Nature, vol. 550, no. 7676, pp. 354–359, 2017.

[4] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2015, pp. 1–14.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.

[6] A. Dosovitskiy et al., “An image is worth 16 × 16 words: Transformers for image recognition at scale,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2021, pp. 1–21.

[7] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2016, pp. 1–16.

[8] Z. Lu, R. Cheng, S. Huang, H. Zhang, C. Qiu, and F. Yang, “Surrogate-assisted multiobjective neural architecture search for real-time semantic segmentation,” IEEE Trans. Artif. Intell., early access, Oct. 11, 2022, doi: 10.1109/TSAI.2022.3215352.

[9] L. Lu et al., “Multiobjective evolutionary design of deep convolutional neural networks for image classification,” IEEE Trans. Evol. Comput., vol. 25, no. 2, pp. 277–291, Apr. 2021.

[10] Z. Lu, G. Sreekumar, E. Goodman, W. Banzhaf, K. Deb, and V. N. Boddeti, “Neural architecture transfer,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 9, pp. 2971–2989, Sep. 2021.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 185–200, Apr. 2002.

[12] Z. Lu et al., “Evolutionary constrained multi-objective optimization algorithms, early access, Jul. 2022.

[13] L. Liang et al., “A survey on evolutionary constrained multi-objective optimization,” IEEE Trans. Evol. Comput., vol. 20, no. 1, pp. 1997–2017, 2019.

[14] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, “PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum],” IEEE Comput. Intell. Mag., vol. 12, no. 4, pp. 73–87, Nov. 2017.

[15] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, “NAS-bench-101: Towards reproducible neural architecture search,” in Proc. Int. Conf. Mach. Learn. (ICML), 2019, pp. 7105–7114.

[16] X. Dong and Y. Yang, “NAS-BENCH-201: Extending the scope of reproducible neural architecture search,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2020, pp. 1–16.

[17] X. Dong, L. Liu, K. Musial, and B. Gabrys, “NAS-bench: Benchmarking NAS algorithms for architecture topology and size,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 7, pp. 3634–3640, Jul. 2022.

[18] A. Zela, J. N. Siems, L. Zimmer, J. Lukasik, M. Keuper, and F. Hutter, “Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2022, pp. 1–16.

[19] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Dept. Comput. Sci., Univ. Toronto, Rep. TR-2009, 2009.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, Apr. 2002.

[21] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength Pareto evolutionary algorithm,” in Proc. Evol. Methods Design Optim. Comput. Syst. Eng., vol. 27, 2014, pp. 1–9.

[22] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, “PESA-II: Region-based selection in evolutionary multi-objective optimization,” in Proc. Genet. Evol. Comput. Conf. (GECCO), 2001, pp. 283–290.

[23] H.-L. Liu, F. Gu, and Q. Zhang, “Decomposition of a multi-objective optimization problem into a number of simple multi-objective sub-problems,” IEEE Trans. Evol. Comput., vol. 18, no. 3, pp. 2450–2455, Jun. 2014.

[24] E. Zitzler and S. Künzi, “Indicator-based selection in multi-objective search,” in Proc. Parallel Problem Solving Nat., 2004, pp. 832–842.

[25] M. Emmerich, N. Beume, and B. Naujoks, “An EMO algorithm using the hypervolume measure as selection criterion,” in Proc. Evol. Multi-Criterion Optim., 2005, pp. 62–76.

[26] E. Zitzler, K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: Empirical results,” Evol. Comput., vol. 8, no. 2, pp. 173–195, 2000.

[27] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable test problems for evolutionary multiobjective optimization,” in Evolutionary Multiobjective Optimization. London, U.K.: Springer, 2005.

[28] S. Huband, P. Hingston, L. Barone, and L. While, “A review of multiobjective test problems and a scalable test problem toolkit,” IEEE Trans. Evol. Comput., vol. 10, no. 5, pp. 477–506, Oct. 2006.

[29] K. Cheng et al., “A benchmark test suite for many-objective optimization,” Complex Intell. Syst., vol. 3, no. 1, pp. 67–81, 2017.

[30] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, “Test problems for large-scale multiobjective and many-objective optimization,” IEEE Trans. Cybern., vol. 47, no. 12, pp. 4108–4121, Dec. 2017.

[31] B. Baker, O. Gupta, N. Naik, and R. Raskar, “Designing neural network architectures using reinforcement learning,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2017, pp. 1–18.

[32] M. Tan et al., “MnasNet: Platform-aware neural architecture search for mobile,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 2820–2828.

[33] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2019, pp. 1–13.

[34] A. Howard et al., “Searching for MobileNetV3,” in Proc. Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 1–11.

[35] K. T. Chitty-Venkata and A. K. Somani, “Neural architecture search based on decomposition,” in Proc. ICSI Conf. Learn. Represent. (ICLR), 2020, pp. 1–16.

[36] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, [Online]. Available: https://www.tensorflow.org/
Kay Chen Tan (Fellow, IEEE) received the B.Eng. (First Class Hons.) and Ph.D. degrees from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is currently a Chair Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong.

Dr. Tan was the Editor-in-Chief of IEEE Transactions on Evolutionary Computation, and is currently the editorial board member of more than ten journals. He is also the Vice-President (Publications) of IEEE Computational Intelligence Society, IEEE Distinguished Lecturer Program Speaker, an Honorary Professor with the University of Nottingham, Nottingham, U.K., and the Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.

Kalyanmoy Deb (Fellow, IEEE) received the B.Sc. degree in mechanical engineering from IIT Kharagpur, Kharagpur, India, in 1985, and the M.Sc. and Ph.D. degrees in engineering mechanics from the University of Alabama, Tuscaloosa, AL, USA, in 1989 and 1991, respectively.

He is a University Distinguished Professor and a Koenig Endowed Chair Professor with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, USA.

He has written more than 580 research papers. His research interests are in evolutionary optimization algorithms and application in optimization and machine learning.

Prof. Deb received many awards, including the Infosys Prize and Edgeworth Pareto Award. He is a Fellow of ASME.