Score Predictor-Assisted Evolutionary Neural Architecture Search

Pengcheng Jiang, Yu Xue, *Senior Member, IEEE*, and Ferrante Neri, *Senior Member, IEEE*

***Abstract*—Evolutionary neural architecture search (ENAS) treats neural network design as an optimisation problem and addresses it via evolutionary computation. Despite being flexible and enabling automated design, ENAS typically suffers from high computational costs due to the need to train a network at each fitness evaluation. Surrogate-assisted ENAS methods mitigate the severity of this challenge by replacing the computationally expensive fitness function with an approximate computationally cheap fitness function for some fitness evaluations of the run. Currently, a major research challenge in the field is the smooth integration of such surrogate models (and, often, data collection mechanisms) within ENAS frameworks. This paper puts forth a simple yet effective way to address this challenge. During the initial stage of the optimisation, the proposed algorithm, score predictor-assisted ENAS (SPNAS), evolves a small population of candidate architectures using ground truth fitness, i.e., the testing error rate of the network following its training. The data collected in this stage are then used to train a multi-layer perceptron network that builds an alternative fitness function. Unlike algorithms in previous studies, this novel alternative fitness does not approximate the error rate but is designed to preserve its order relation over populations of candidate architectures. Thus, this approach naturally allows for a computationally cheap population ranking. Most of the evolution is then carried out with the surrogate (i.e., alternative) fitness on a large population without retraining the surrogate model or calculating the ground truth fitness. Experiments conducted on the EvoXBench platform show that on its seven search spaces, SPNAS achieves excellent results in terms of error rate despite the modest use of ground-truth fitness calls.**

***Index Terms*—Evolutionary neural architecture search, surrogate model, genetic algorithm, score prediction.**

I. Introduction

DEEP neural networks (DNNs) are popular machine learning algorithms that can demonstrate exceptional performance in many current vision domains, including image classification [1], object detection [2], [3] and semantic segmentation [4]. However, for each problem and dataset, a DNN's performance depends on its architecture. Designing a neural architecture to address a specific problem is a complex task for human and relies on their experience. Furthermore, an extremely specialised architecture might be not universal since the vast majority of real-world problems must show some flexibility as well as good performance across various scenarios.

Neural architecture search (NAS) is a method used for automatically designing neural architecture, thus precluding human expertise and enabling *a priori* encoding of the design requirements. In other words, a NAS problem can be formulated as the search of that architecture that, once trained, provides minimal error on the validation set. More formally, a NAS can be described as the following optimisation problem:

$$\begin{aligned} &\min_{\mathcal{A}} Error(\mathcal{A}, W^*, D_{val}) \\ &\text{s.t. } W^* = \arg\min_{W} Loss(\mathcal{A}, W, D_{train}), \end{aligned} \tag{1}$$

where D_{train} denotes the dataset used for training, D_{val} denotes the dataset used for validation, $\mathcal{A}$ denotes the candidate neural network architecture, W denotes the weights in the corresponding architecture, and W^* denotes the weights obtained after training the neural network on the training dataset. The final desired neural network architecture is obtained by minimally optimising $\mathcal{A}$.

If the NAS problem in (1) is tackled via an evolutionary computation approach, the corresponding NAS subfield is often referred to as evolutionary NAS (ENAS) [5].

Analogous to graphs, neural architectures are naturally encoded as vectors of discrete numbers. Thus,once endowed with appropriate search operators, evolutionary algorithms are a natural option for NAS. Conversely, approaches that transform a discrete problem into a continuous one to then exploit its gradient, such as differentiable neural architecture search methods, generate and explore ill-formed architectures [5], [6].

Neural architecture search methods based on reinforcement learning, or the use of a reward strategy to find the optimal neural network, may be computationally expensive, e.g., 800 GPUs searching for 28 days (22,400 GPU days), as reported in Reinforce NAS [7]. Although ENAS is usually much cheaper than reinforcement learning, its implementation can still be computationally demanding, often taking several days to find a high-performing architecture. For example, the AE-CNN proposed by Sun et al. takes 27 GPU days [8]. The search time for CNN-GA is 35 GPU days [9], while NAT uses about 59 GPU

Received 4 March 2024; revised 16 August 2024; accepted 6 November 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62376127, Grant 61876089, Grant 61876185, and Grant 61902281, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20141005, and in part by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China through the Jiangsu Distinguished Professor Programme under Grant 14KJB520025. *(Corresponding author: Yu Xue.)*

Pengcheng Jiang and Yu Xue are with the School of Software, Nanjing University of Information Science and Technology, Jiangsu 210044, China (e-mail: pcjiang@nuist.edu.cn; xueyu@nuist.edu.cn).

Ferrante Neri is with the School of Software, Nanjing University of Information Science and Technology, Jiangsu 210044, China, and also with the NICE group, School of Computer Science and Electronic Engineering, University of Surrey, GU2 7XH Guildford, U.K. (e-mail: f.neri@surrey.ac.uk).

Recommended for acceptance by Y. Jin.

Digital Object Identifier 10.1109/TETCI.2025.3526179

days [10]. FairNAS uses 12 days [11], and Evo-OSNet searches for 8.6 GPU days [12]. This demonstrates that ENAS has several limitations when applied to real-world scenarios. Since the time consumed by this approach mainly stems from the training of the weights of a candidate architecture, a technique that reduces the training's computational time would drastically reduce the computational cost of the entire search process, thus making the ENAS design much more viable.

One popular approach to reduce the computational cost of NAS is to employ one-shot learning techniques for training, meaning the training of the candidate architecture is performed based on a single or a very limited set of data. This is often interpreted as inheriting weights from an external network. One early example of this approach was proposed by Brock et al. [13], where an auxiliary HyperNet generates the weights for the candidate architecture. Following a similar idea, the Single Path One-Shot (SPOS) model uses a simplified supernet that samples candidate architectures as single paths of such a supernet [14]. Lu et al. have used an approach based on weight sharing to search for optimal network architectures from the knowledge of weights in the super-net, thus shortening the training process of the candidate networks [15]. Huang et al. have presented an approach based on weight inheritance to initialise the network weights of part of the offspring architectures using the already trained weight knowledge of the parent generations, thus shortening the training process for the offspring and assisting the evolutionary search process [16].

The work of Bender et al. performs an experimental study to interpret and understand the one-shot mechanisms in NAS based on weight inheritance (a.k.a. weight sharing) [17]. Dong et al. propose a different approach where the auxiliary network is used to predict the quality-based ranking of several architectures [18]. PRENAS combines the weight inheritance and performance prediction approaches [19]. Specifically, the algorithm reduces the sample space by a zero-cost selector and performs one-shot training through a weight inheritance mechanism. Following analogous logic, ShiftNAS [20] makes use of a supernet but adjusts the sampling probability based on the complexity of subnets. This is achieved by evaluating the performance variation of subnets with different complexity and designing an architecture generator that samples subnets with the desired complexity.

As an alternative approach to expedite the calculation time in NAS, surrogate-assisted models—using computationally inexpensive models that predict the performance of a candidate architecture—are employed to reduce the number of architecture training instances [21], [22]. For example, Sun et al. have used random forests for end-to-end performance prediction, significantly reducing the search time of AE-CNN [23]. Direct prediction of architecture performance is likely to lead to inaccurate comparisons and, thus, ranking due to unavoidable error propagation. Wang et al. have overcome this issue by predicting the outcome of a comparison rather than each individual performance. The method proposed by Wang et al. uses support vector machine (SVM) as a surrogate model to assist the search of a PSO-like ENAS. The already evaluated individuals are used as empirical knowledge to train a 'comparator' and thus quickly compare two architectures to predict which could demonstrate better performance [24]. However, this estimation of each pairwise comparison, albeit powerful within a PSO framework, carries an important limitation when integrated within an ENAS that requires ranking of the entire population. Each comparison typically contains a small error with respect to the ground truth, and these errors propagate and become amplified when multiple comparisons are predicted, resulting in an unreliable population raking.

The multi-layer perceptron network (MLP) is widely used as a regression and classification model and is also employed as a surrogate model for NAS. Lu et al. employed four surrogate models, including MLP, and adaptively selected the optimal surrogate model based on the results of K-fold validation [15]. White et al. built a feed-forward neural network and trained it with the mean absolute error loss function to fit limited data [25]. Hassantabar et al. used MLP to construct a regression model and fit it using the MSE loss function [26]. Lu et al. constructed an accuracy predictor using MLP and designed a loss function to reflect the ranking information between any two samples [27]. Additionally, they utilised other mature machine learning models such as radial basis function, decision tree, gradient boosting as teacher models to assist in training MLP. Xue et al. utilised a shared-parameter MLP to distinguish the similarity between any two networks, aiming to find better solutions near the optimal network [28].

The various methods to expedite NAS and ENAS present relative drawbacks. The methods that make use of a supernet or an external network for weight sharing may require extensive pre-training, as well as fine-tuning or augmentation of the designed architecture [29]. The supernet provides performance estimations for the architectures within the search space that it defines, enabling the search procedure to quickly filter out less promising configurations and focus on more promising regions. This approach helps streamline the search process by improving the efficiency of exploring the architecture space. Moreover, the supernet provides a search space that allows for the discovery of high-performance networks within a constrained resource budget, without incurring the cost of exploring a much larger, unconstrained space. The methods that use surrogate models may incur a high computational cost for building/training the approximating function. Additionally, the surrogate function might be unreliable and may mislead the search of the NAS algorithm. The latter two problems are especially relevant when the surrogate model approximates complex objective functions.

The present study proposes a method called score predictor-assisted ENAS (SPNAS). The design of the proposed SPNAS has been motivated by two principles:

- A surrogate model that simply respects the order relation of the objective function is much easier to define and train than a surrogate that faithfully approximates an entire objective function's landscape.
- The efficiency of a surrogate model can be significantly enhanced if its design naturally integrates within the operations of the evolutionary framework that embeds it.

By adhering to these principles, the proposed SPNAS, in contrast to other surrogate-assisted NAS algorithms that utilise

MLP, leverages MLP as the surrogate model to expedite the evaluation process in a distinct manner from existing works. It should be noted that simplifying the problem to an alternative function to the objective function, which preserves only its order relation (and thus the ranking of the candidate architectures), allows for the use of a straightforward and easily trainable model like MLP. The proposed surrogate model is designed to predict the ranking of individuals within a population to facilitate the selection of the most promising candidates. This model effectively ensures that the top individuals are retained for the next generation. The main characterising elements of the proposed SPNAS are as follows:

- The network architecture is represented using the **encoding** in EvoXBench for evolution and augmented via using one-hot conversion of all non-binary regions to achieve full binary encoding of each candidate architecture for the score predictor.
- An MLP network is used as a **score predictor** model, and a special loss function is designed to ensure that the predicted score ranking is consistent with the ranking of the ground-truth information available.
- A GA-based **search framework** whose evolution is composed of a training stage occurring on a small population and a prediction stage occurring on a large population is designed.
- The MLP score predictor is **designed to assign ranking scores** to candidate architectures within the population. These scores are then used by the GA selection mechanisms to process and select individuals at each generation.

The remainder of this paper is organised as follows: Section II introduces related work. Section III describes the proposed SPNAS in detail. To verify the effectiveness and efficiency of the proposed method, the experimental design and results are given in Section IV. Finally, the conclusions and future work are discussed in Section V.

II. Related Work

This section first introduces the encoding strategy used in EvoXBench [30], which is the basis for the sampling and search in this paper. Then,state-of-the-art neural network performance predictors are reviewed, and the limitations of current NAS methods are analysed.

A. Search Space and Encoding Strategy in EvoXBench

There are seven search spaces in EvoXBench, including NASBench-101, NASBench-201, NATS, DARTS, Moblie NetV3, ResNet50 and Transformer. Among these, NASBench-101, NASBench-201 and DARTS are encoded using micro-encoding, and only the in-cell connections are represented. Previously, Ying et al. trained all candidate networks in the search space of graph-based representations and collapsed all of the data into NASBench-101 [31]. All candidate networks in NASBench-101 are encoded as 26 bits, of which the first 21 bits are the upper triangular region of a 7 × 7 adjacency matrix, using '1' and '0' in the i-th column; the j-th line representing the i-th operation is connected to the j-th operation. The first and last operations are the input and the output, respectively. For the last five bits in the encoding, the ordinal indexes of operation types are represented as the middle five nodes, including 'conv3x3-bn-relu', 'maxpool3x3' and 'conv1x1-bn-relu'. In their work, Dong et al. used the architectures of three cell stages mixed with two residual blocks. They placed five cells in each stage and used the same topology for all cells in one network, NASBench-201 [32], [33]. To encode the networks, the cells contained six operations represented by edges, and the operations were represented by $\{0, 1, 2, 3, 4\}$ corresponding to the five possible operations. The last micro search space is DARTS. Utilising the same method used for the original search space, Zela et al. represented the entire network using an architecture with two inputs, four middle nodes, and one output; there were eight total operations between all nodes [34]. The difference stemmed from the removal of 'none' operations from the optional operations.

The other four search spaces are represented using a macro encoding approach with hyper-parameters in the macro architecture. Previously, based on the macro architecture of NASBench-201, Dong et al. used only one predefined cell topology in each stage and set the number of channels for all cells and residual blocks to be generated from eight kinds of numbers, this algorithm is termed NATS [35]. Cai et al. designed the ResNet50 search space based on the hyper-parameters used by ResNet during the design process, by which hyper-parameters such as the size of the input image, the depth of each block and the dilation rate are encoded [36]. They also encoded the network for MoblieNetV3, in which hyper-parameters such as the size of the input image, the size of the convolution kernel and the depth of the convolution block are encoded [36]. With the great success of vision transformer in the computer vision field, for the first time, Chen et al. constructed a benchmark dataset with the search space in the form of Transformer; the encoding of each network includes the depth of the network, the number of neurons in the hidden layer and the number of heads of the multi-head attention mechanism [37].

B. Surrogate-Assisted Neural Architecture Search

To reduce the massive time consumption required in the search process of neural network architectures, many surrogate approaches have been put forwards over the past years. This paper proposes the following categorisation of modern surrogate-assisted NAS approaches: *1) Training curve prediction:* For a given architecture, this method predicts its training curve and, thus, its performance. For example, Klein et al. used a Bayesian neural network to predict the training curves of a convolutional neural network [38]. Baker et al. used a similar approach, with the difference being that they built a regression model to obtain the corresponding features from the network structure, hyper-parameters and early learning curves [39]. These features, along with the training curves, were used to predict the final performance. *2) Accuracy/error rate prediction:* These methods directly predict the performance (accuracy/error rate) of a candidate architecture, building up a mapping between the NAS search space and the performance. Previously, Dudziak et al. used a graph convolutional network (GCN) to predict accuracy. They noted that it was difficult to directly learn the accuracy patterns, so they adopted the approach of first learning a latency predictor based on the network architecture and then transferring the latency predictor to the accuracy prediction task [40]. Wen et al. used a GCN to predict accuracy by further trading off the number of samples and the number of retentions, thus improving efficiency and effectiveness [41]. Finally, Sun et al. used random forests for end-to-end performance prediction via integrating a large number of decision trees and thus obtaining good prediction accuracy [23]. *3) Comparative relationship prediction:* For a pair of candidate architectures, this method predicts which of the two has higher performance (i.e., it predicts the outcome of a pairwise comparison). In their work, Chen et al. trained a neural architecture comparator using comparative learning and explored the possible accuracy based on the results, thus reducing the dependence on training data [42]. Wang et al. used SVM to predict the possible comparative relationship between two networks and later used the PSO method to explore the search space and filter out architectures during the search process to reduce the search time [24]. However, such approaches may introduce the problem of circular comparisons in the prediction phase. *4) Rough range prediction:* This method uses predictors with low accuracy (and fast training) to select potentially good solutions from a large set of candidate architectures. Wu et al. attempted to use random forests to find roughly good networks by continuously narrowing down the range and thus obtaining the final searched architecture [43]. However, it is impossible to know whether the architecture designed via this method is satisfactory, meaning further evaluation of the last predicted network is needed. *5) Ranking predictionL* This method predicts a score that, although not representing a performance estimate, preserves the order relation of the performance and thus allows for the ranking of a set of solutions. Zheng et al. discovered that models that perform well in the pre-training period tend to attain a good result after convergence. They obtained relative performance ranking via the early performance of candidate network architectures to speed up the search process [44]. Xu et al. proposed adding RankLoss to the prediction as an auxiliary function. However, this method relies greatly on the amount of data available [45]. Guo et al. modified this method by adding tire prediction to compensate for the imprecise prediction of superior networks in the case of insufficient data [46]. Hu et al. employ a pairwise labelling approach to train their ranking model, which enables them to derive a scoring function that assigns scores to all architectures [47].

The proposed SPNAS belongs to the latter category. As mentioned above, surrogate-assisted methods with rank-based score prediction are especially useful in ENAS approaches that routinely require population ranking, such as GA frameworks. Since this paper's SPNAS relies on a GA structure, it is used within the ranking prediction category. However, although inspired by existing research, this SPNAS is characterised by a radically different algorithmic approach. While all existing methods perform ranking prediction using specific pieces of information about the accuracy/error rate of trained architectures at some point, the current SPNAS, in a holistic surrogate spirit, builds an alternative objective function of the candidate architecture vectors and focuses only on the rank of the ground truth instead of on values, without taking intermediate steps to partly estimate the performance of the candidate architecture.

III. Score Predictor-Assisted Evolutionary Neural Architecture Search

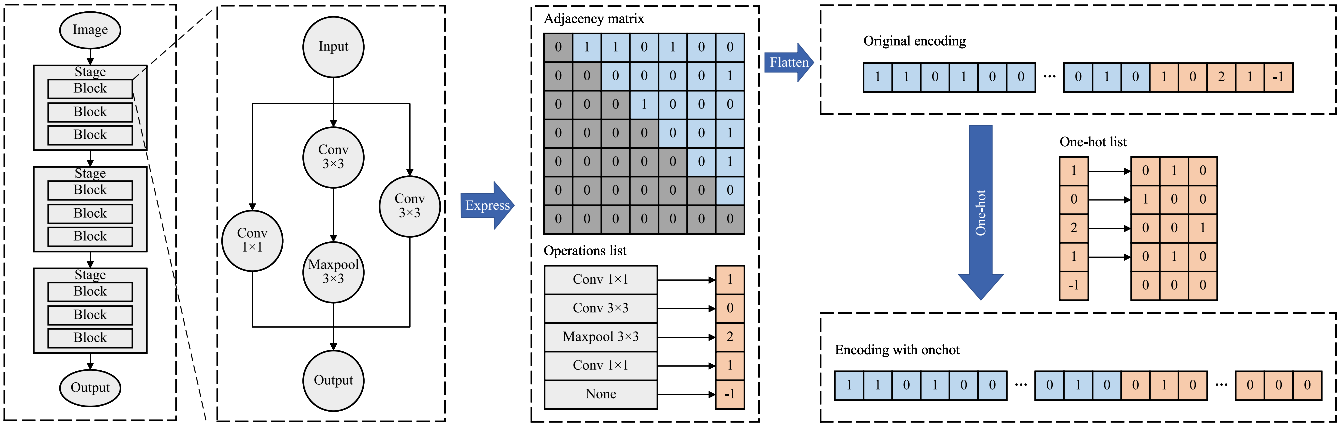
A. Encoding

In this paper, the encoding of candidate architectures adopts and modifies the method introduced in EvoXBench [30]. As an example of the current paper's encoding strategy, the encoding strategy for one CNN block in NASBench-101 is schematically represented in Fig. 1. As a CNN is composed of nine identical blocks, the encoding in Fig. 1 univocally identifies the entire candidate architecture. The candidate architectures corresponding to a CNN block in NASBench-101 are encoded as a 26-bit binary vector. The block's topology is described via an adjacency matrix. Due to the symmetry of the representation, the upper triangular matrix, which is 21 bits, already contains all relevant information. In each block, there are five operation nodes (circles in Fig. 1), one input node and one output node. Each operation can be one of three types, represented as $\{0, 1, 2\}$. Furthermore, the absence of an operation node is indicated with the null flag -1. Fig. 1 displays a null operation as 'None'. Thus, in the encoding proposed in EvoXBench, one block, like the one in Fig. 1, will have $21 + 5 = 26$ bits, where some are binary numbers and some are integers. Different from the other benchmark search spaces, the architectures in NASBench-101 contains useless nodes which do not have any input or output. In order to reduce the impact of unnecessary nodes on the training of the surrogate model, we completely remove the redundant nodes and use '0' to pad.

In addition to the original encoding from EvoXBench, this work also handles the fully binary versions of the vectors. To obtain fully binary vectors, each operation module is converted from integer to three-bit binary vector via the one-hot transformation, as depicted on the right-hand side of Fig. 1. Then, the 'None' operation is represented as a vector of zeros. Thus, in the score predictor of SPNAS, a candidate architecture in NASBench-101 is encoded as a binary vector of $21 + 3 \times 5 = 36$ bits. The encoding for the other spaces is analogous, and the one-hot encoding is used to represent all non-binary parts as binary vectors.

The original architecture representations reported in [30] are used for the GA search of candidate architectures. The modified fully binary vectors are used for training the surrogate model, as described in Section III-B.

Table I lists the vector length in the original and one-hot encoding of each candidate architecture block in each benchmark considered here. In the table, 'Dataset' indicates that the network in this search space is designed and trained based on the corresponding image dataset, and 'Size' indicates the number of candidate architectures in the search space.



0	1	1	0	1	0	0
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Fig. 1. Encoding example on NASBench-101 search space.

TABLE I
AN OVERVIEW OF THE SEARCH SPACE IN EVOXBENCH

Benchmark	Type	Dataset	Size	Dimension	
				Original	One-hot
NASBench-101	micro	CIFAR-10	423K	26	36
NASBench-201	micro	CIFAR-10 CIFAR-100 Imagenet-16-120	15.6K	6	30
NATS	macro	CIFAR-10 CIFAR-100 Imagenet-16-120	32.8K	5	40
DARTS	micro	CIFAR-10	$\approx 10^{21}$	32	168
ResNet50	macro	ImageNet-1K	$\approx 10^{14}$	25	104
Transformer	macro	ImageNet-1K	$\approx 10^{14}$	34	86
MoblieNetV3	macro	ImageNet-1K	$\approx 10^{20}$	21	205

B. Score Predictor

Since a large amount of data with which to train a surrogate model is usually not available, this paper uses an MLP surrogate model as the score predictor of the proposed SPNAS. While this MLP always contains three hidden layers, the number of neurons contained in each hidden layer is dynamically adapted according to the length of the encoding represented in the search space used. In the first two hidden layers, the number of neurons grows to twice the dimension of the input data. In the last hidden layer, the number of neurons is equal to the dimension of the input data. As such, satisfactory training with a relatively small amount of data is possible in all scenarios. All trained data are in the fully binary form described in Section III-A. The MLP input is a fully binary vector, similar to the 36-bit vector used for NASBench-101, while the output is the score associated with the network, or scalar.

A commonly used loss function in supervised learning is the mean square error (MSE) between the predicted vectors and the corresponding true labels. However, in the current case, the use of MSE would be inappropriate, as it would not preserve the correct ranking of the solutions. For example, consider three candidate architectures whose ground-truth error rates (labels) are [0.08,0.09,0.10]. For the same three architectures, consider two sets of predicted error rates, [0.07,0.11,0.13] and [0.11,0.07,0.09]. The first prediction has the same ranking order as the ground truth, while the second does not. However, the MSE associated with both predictions is 0.0005. Thus, if the MSE were used, it would be impossible to distinguish the two scenarios.

To preserve the solution ranking, during the training process, the data are loaded in batches, thus obtaining a vector consisting of the predicted values. More formally, with N, the batch size and corresponding batch are indicated as follows:

$$(X, Y) = \{(\mathbf{X}_i, y_i)\}_N^{i=1}, \tag{2}$$

where $\mathbf{X}_i$ is the i^{th} candidate architecture of the batch encoded as a fully binary vector, as described in Section III-A, and y_i is the corresponding error rate. With $\mathcal{F}$, the function that returns the predicted error, $\hat{y}_i$, is calculated by the MLP network, associated with the candidate architecture $\mathbf{X}_i$, i.e., $\hat{y}_i = \mathcal{F}(\mathbf{X}_i)$. The set of predicted values for the entire batch is then indicated with $\hat{Y}$. The $PairLoss$ function $\mathcal{L}_1(\hat{Y}, Y)$ is introduced below:

$$\mathcal{L}_1(\hat{Y}, Y) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \phi((\hat{y}_i - \hat{y}_j) \times \text{sign}(y_i - y_j)), \tag{3}$$

where ϕ is the monotonically decreasing function $\phi(x) = \log(1 + \mathrm{e}^{-x})$ and $sign$ is the sign function. The $\mathcal{L}_1$ function amplifies the difference in performance between each pair of candidate architectures $\mathbf{X}_i$ and $\mathbf{X}_j$ contained in a batch.

However, the $PairLoss$ function $\mathcal{L}_1$ on its own is not a suitable loss function, as it performs only pairwise comparisons. As an example, consider three candidate architectures, $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$. Assume that $\mathbf{X}_1$ and $\mathbf{X}_2$ both have better performance than $\mathbf{X}_3$. If the parameters of the MLP surrogate are optimised only on the basis of $\mathcal{L}_1$, two separate updates will be observed in the directions of $\mathbf{X}_1$ and $\mathbf{X}_2$ (coming from the two comparisons $\mathbf{X}_1$ vs $\mathbf{X}_3$ and $\mathbf{X}_2$ vs $\mathbf{X}_3$, respectively). However, the magnitudes of these updates do not consider how large the $\mathbf{X}_1$ vs $\mathbf{X}_3$ improvement is with respect to $\mathbf{X}_2$ vs $\mathbf{X}_3$. In short, $PairLoss$ lacks distance perception. Correct management of this issue will enable better separation in the scores and, thus, more accurate ranking of multiple solutions.

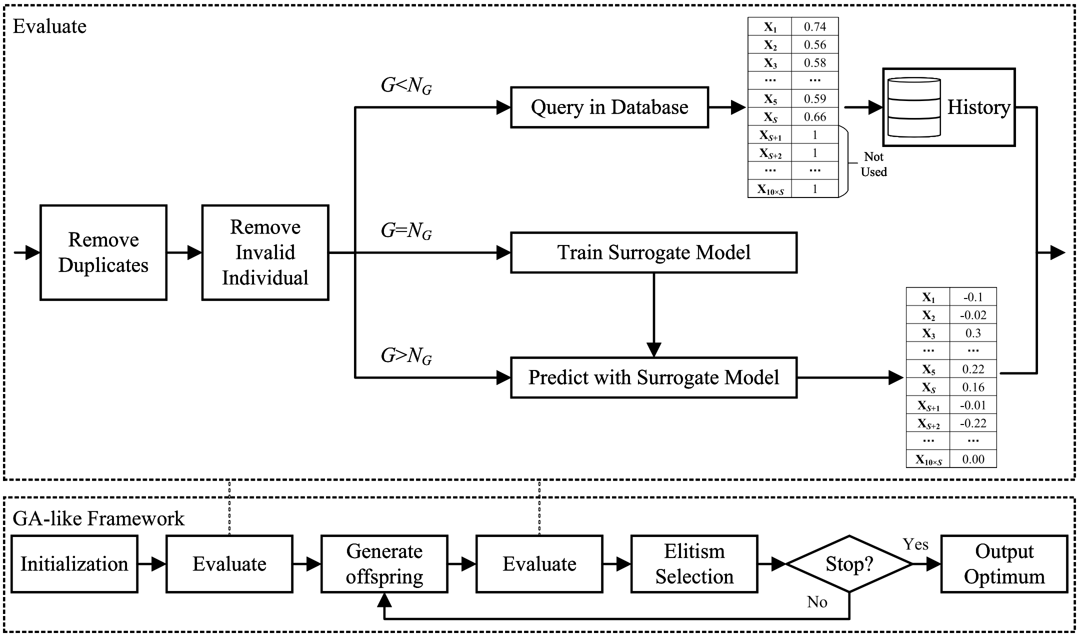


Fig. 2. The overall framework of score predictor-assisted evolutionary neural architecture.

To correct this limitation of $PairLoss$, this paper proposes a further function, $DistanceLoss$, defined as follows:

$$\mathcal{L}_2(\hat{Y}, Y) = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} \phi((\hat{d}_i - \hat{d}_j) \times \text{sign}(d_i - d_j)), \quad (4)$$

where $\mathbf{d}$ is the vector whose generic element is $|y_k - y_l|$ with $k \in \{1, 2, \ldots, N-1\}$ and $l \in \{k+1, k+2, \ldots, N\}$. The vector $\hat{\mathbf{d}}$ is analogous to $\mathbf{d}$ but refers to the predicted values. The length of vectors $\mathbf{d}$ and $\hat{\mathbf{d}}$ is $M = \frac{(N-1)\times N}{2}$.

During the training process, the loss function in (5) is thus used to train the surrogate MLP network.

$$\mathcal{L}(\hat{Y}, Y) = \mathcal{L}_1(\hat{Y}, Y) + \text{Sigmoid}(\alpha) \times \mathcal{L}_2(\hat{Y}, Y), \quad (5)$$

where α is a trainable parameter, and we use the Sigmoid function to constrain the weight of $\mathcal{L}_2(\hat{Y}, Y)$ within the range of 0 and 1. This decision is made to ensure that the training places greater emphasis on $\mathcal{L}_1$ than on $\mathcal{L}_2$. The former, which focuses on determining the winner of the pair-wise comparisons, makes a more significant contribution to accurate ranking predictions. The proposed surrogate model is the MLP network with the architecture described above and trained via minimising the loss function $\mathcal{L}$ in (5).

C. Score Predictor-Assisted Evolutionary Neural Architecture Framework

A search framework based on GA logic and embedding the elements described in Sections III-A and III-B has been designed to perform the NAS task. Fig. 2 depicts the overall framework of SPNAS. In this section, the term fitness will be used to refer to the score associated with the candidate architectures regardless of how it was calculated, e.g., via ground-truth or surrogate score. In the case of the ground-truth score (error rate of a sampled and trained architecture), a positive definite function is minimised between 0 and 1. It can be immediately observed that error rate functions always take values between 0 and 1. In the case of the surrogate score, an unbounded function is minimised, which allows for negative scores (the lower, the better). This latter function is the loss function $\mathcal{L}$ defined in (5). This difference is not an issue since SPNAS never mixes ground-truth and surrogate scores in the same generation. The SPNAS framework is divided into two levels.

The first level is a GA-like evolutionary framework whose role is to perform the search in the space of the candidate architectures, as seen in Algorithm 1. At the beginning of the evolutionary process (lines 1–4), a population of candidate architectures is randomly generated and the MLP surrogate initialised. The candidate architectures are encoded as vectors; see Section III-A. Note that while the user chooses the size of the population to undergo ground-truth fitness evaluations (training stage), a population 10 times larger is sampled for the prediction stage; see S in Algorithm 1.

In the generation cycle (lines 6–21), the offspring population Q is generated by recombining the individuals of population P via crossover and mutation operators. To start, binary tournament selection is performed to choose the parents undergoing crossover. Then, uniform crossover is performed to each pair of parents. The logic of crossover is similar to that recently discussed in [48]. A randomly chosen mutation is used in this GA-like evolutionary framework, which changes elements with a probability and randomly changes to another value between the board.

The performance of offspring population Q is evaluated. Then, in a fully elitist fashion, the best solutions from both P and Q ($P.fitness$ and $Q.fitness$ indicate the list of fitness values associated with populations P and Q, respectively) are saved

Algorithm 1: GA-like Evolutionary Framework.

```
Input: size of population to true evaluate $S$, number of
    generations $T$, number of generations to true evaluate
    $N_G$.
Output: Best individual $\mathbf{X}_{best}$.
 1: Initialise $10 \times S$ individuals as the initial population
    $P$.
 2: Initialise a MLP as the score predictor $M$.
 3: $G \leftarrow 0$.
 4: $History \leftarrow \{\}$.
 5: Evaluate($G$, $N_G$, None, $P$, $History$, $M$) according to
    Algorithm 2.
 6: while $G < T$ do
 7:    Tournament parent selection with size 2.
 8:    Crossover and mutation to generate offspring $Q$.
 9:    Evaluate($G$, $N_G$, $P$, $Q$, $History$, $M$) according to
       Algorithm 2.
10:    Elitist selection from $P.fitness$ and $Q.fitness$ to
       form the new population $P$.
11:    $G \leftarrow G + 1$.
12: end while
13: Select the best individual $\mathbf{X}_{best}$ based on the best $L$
    validation error rate.
14: Get true error rate of $\mathbf{X}_{best}$.
```

Algorithm 2: Evaluate(G, N_G, P, Q, $History$, M).

```
Input: current generation $G$, parent population $P$, offspring
    population $Q$, fitness records $History$, score prediction
    $M$, number of generations to true evaluate $N_G$.
Output: The fitness of $Q$.
 1: Check duplicates in $Q$ and marked.
 2: Check invalid individuals from $Q$ and marked.
 3: if $G < N_G$ then
 4:    Query true fitness $Q.fitness[1{:}S]$ for the first $S$
       unmarked individuals in $Q$.
 5:    Set fitness $Q.fitness[S+1{:}10 \times S] \leftarrow 1$ for the
       other $9 \times S$ individuals in $Q$.
 6:    Record $Q$ in $History$.
 7: else if $G = N_G$ then
 8:    Convert all non-binary regions in encoding of each
       record in $History$ to one-hot format.
 9:    Train $M$ with $History$.
10:    $Q.fitness \leftarrow M(Q)$
11:    $P.fitness \leftarrow M(P)$
12: else
13:    $Q.fitness \leftarrow M(Q)$
14: end if
```

into P for the following generation. After the last generation, the ground-truth error rate of final population P is calculated and the top L architectures are achieved. Then, the validation error rate of the top L architectures are evaluated and the best architecture, $\mathbf{X}_{best}$, is selected as the output of the ENAS process.

The second level is the evaluation, or the calculation of the Evaluate(G, N_G, P, Q, $History$, M) function to obtain the fitness of the offspring population Q outlined in Algorithm 2. To start, the offspring population is scrolled to identify and mark duplicate and invalid solutions, i.e., those solutions that do not correspond to a meaningful network architecture. Eliminating duplicates is important in the training stage, as it ensures that the training data for the surrogate model are diverse.

If the algorithm is in the training stage (lines 3–11), then S valid and unique architectures are trained to calculate their error rates (fitness values). The data associated with this training are saved in the $History$ dataset. The fitness of all remaining $9 \times S$ candidate architectures is set to 1. The latter operation effectively excludes these $9 \times S$ solutions from the evolution. The handling of a population of size $10 \times S$ when S solutions are needed is chosen here to ensure that, at each generation, S valid and unique offspring architectures are generated. This pragmatic algorithmic solution is effective since the proposed SPNAS has always generated enough valid architectures throughout all experiments in this study.

When the data collection is complete (line 7), all solutions in the $History$ dataset are converted into fully binary vectors by applying the one-hot transformation and are then used to train the MLP network to build a surrogate model. Next, the performance of the entire population composed of $10 \times S$ individuals is assessed via the newly formed surrogate. From this point on, only the surrogate fitness is used. In the following generations, i.e., the prediction stage (lines 12–14), the score calculated by the surrogate model is used in lieu of the error rate to rank and select the solutions.

The scheme of the overall SPNAS framework depicted in Fig. 2 graphically represents the GA-line search framework and the Evaluate function, including the training and prediction stages. Note that for simplicity, the 'Generate offspring' block embeds the parent selection and generation of offspring population Q.

The proposed surrogate management enables a natural integration within any population-based metaheuristic. Surrogate-assisted algorithms that combine ground-truth and surrogate models to assess the fitness on individuals of the same population implicitly include noise related to the approximation error [49]. This noise affects the selection operations, such as the ranking, and may further mislead the search (when comparing a ground-truth fitness against a surrogate fitness). In SPNAS, within a generation, either only the error rate or only the surrogate model is used to assess the quality of the entire population. This makes the algorithm more stable, as it eliminates the above-mentioned noise through a simple yet effective strategy.

IV. Experiments

To validate the effectiveness of the proposed SPNAS, this paper's approach was thoroughly tested and compared against state-of-the-art modern NAS algorithms. Section IV-A introduces the SPNAS hyper-parameters. Section IV-B details the results on EvoXBench, while Section IV-C compares the SPNAS algorithm's performance against that of other modern NAS algorithms. Finally, Section IV-E experimentally illustrates some of the functioning principles of SPNAS.

TABLE II
EXPERIMENTAL RESULTS OF ERROR RATES (%) ACROSS ALL SEARCH SPACES AND BENCHMARK DATASETS ON EVOXBENCH

	Validation			Test		
	mean±std	Best	Optimal	mean±std	Best	Optimal
NB101	5.10±0.20	4.94	5.50*	5.84±0.10	5.77	5.68
NB201 (CIFAR-10)	8.43±0.09	8.32	8.39*	5.63±0.00	5.63	5.63
NB201 (CIFAR-100)	26.40±0.11	26.28	26.51*	26.65±0.23	26.49	26.49
NB201 (ImageNet-16-120)	53.58±0.32	53.13	53.23*	53.48±0.32	52.92	52.69
NATS (CIFAR-10)	9.55±0.07	9.50	9.29*	6.67±0.18	6.35	6.35
NATS (CIFAR-100)	29.57±0.21	29.42	29.08*	29.88±0.34	29.28	28.66
NATS (ImageNet-16-120)	54.65±0.68	53.73	53.27*	53.47±0.44	52.83	52.60
DARTS	5.10±0.08	4.99	-	5.64±0.18	5.53	5.27^{+}
MobileNetV3	16.62±0.07	16.55	-	20.86±0.10	20.78	19.87^{+}
ResNet50	16.05±0.05	16.00	-	19.82±0.04	19.75	17.65^{+}
Transformer	17.66±0.01	17.64	-	17.69±0.01	17.67	20.92^{+}

A. Hyper-Parameter Settings and Datasets

Here, SPNAS was run for total number of generations $T = 50$, where the first $N_G = 10$ generations were used for the training stage and the remaining 40 generations for score prediction via the surrogate model. The small population size of the candidate architecture whose ground truth value was calculated was $S = 40$. Thus, in the prediction stage, SPNAS can handle a large population of $10 \times 40 = 400$ individuals. When applying the uniform crossover rate, each bit had a 0.5 probability of being selected from either parent. The mutation occurred on each bit with a 0.05 probability. To train the surrogate model, stochastic gradient descent with a learning rate of 5×10^{-4} and a batch size of 200 were used.

All seven search spaces in EvoXBench were used, where both NASBench-201 and NATS contain benchmark data on three image datasets.

B. Experimental Results on EvoXBench

Eleven sets of experiments were completed on all datasets in EvoXBench, with each experiment comprising five independent runs. The experimental results are shown in Table II, reported in terms of the error rate. In the table, the columns with ‘Optimal’ indicate the validation and test accuracy of the individual that performed best on the test set in the search space. The first two columns indicate the error rate of the best architecture that SPNAS could find on the validation set. The mean, standard deviation and minimum are given for five runs on the validation set. Based on the results of each search on the validation set, the encoding of the best architecture was obtained, and it was evaluated on the test set. The mean, standard deviation and minimum values were reported similarly, recorded in columns 4 and 5. It must be noted that ‘Optimal’ results on the validation set are marked with ‘*’ to indicate that they refer to the performance on the validation set of the architecture known to be optimal on the test set. It follows that the error performance of these architectures appears to be optimal on the test set but not on the validation set. Additionally, it is worth noting that the optimal records with ‘$^{+}$’ on the test set in DARTS, MobileNetV3, ResNet50, and Transformer are not actual evaluation results but rather estimates provided by the original authors. In some datasets, SPNAS was able to find the optimum, and in other datasets, it found a value close to the optimum on the test set.

TABLE III
COMPARISON OF DIFFERENT METHODS REGARDING THE OBTAINED ERROR RATE (%) OF THE CIFAR-10 DATASET AND THE RANKING (%) IN THE SEARCH SPACE OF NASBENCH-101

Methods	#Queries	Accuracy (mean±std)	Accuracy (best)	Rank (%)
Peephole* [50]	1000	6.59±0.34	-	1.6387
NAO* [51]	1000	-	6.26	0.1900^{+}
DARTS* [6]	-	7.79±0.61	6.98	19.6131
ENAS* [52]	-	8.17±0.42	7.46	26.7912
FBNet* [53]	-	7.71±1.25	6.02	18.2041
Reinforce* [54]	400	-	6.26	0.1900^{+}
Bayesian Optimisation* [54]	400	-	6.21	0.1211^{+}
SPOS* [14]	-	10.15±3.80	6.16	63.6744
Random Search* [54]	1000	-	6.46	0.8177^{+}
Bayesian Optimisation* [54]	1000	-	6.28	0.2224^{+}
Regularized Evolution* [54]	1000	-	6.28	0.2224^{+}
E2EPP [23]	1000	6.23±0.13	-	0.1445
SSANA [55]	1000	5.99±0.12	-	0.0111
HAAP [56]	1000	5.91±0.11	-	0.0038
Reinforce* [54]	1000	-	6.42	0.6307^{+}
GenNAS-N [57]	500	6.08±0.004	-	0.0321
BANANAS* [25]	500	-	5.92	0.0047^{+}
ReNAS [45]	423	6.05±0.11	-	0.0222
CTNAS [42]	423	6.08±0.18	5.78	0.0321
Random Search* [42]	423	10.69±3.92	6.54	71.2759
RegressionNAS* [42]	423	10.49±4.94	6.35	68.6082
FairNAS* [11]	-	8.90±1.84	6.45	41.1148
NAR (statistics) [46]	4236	5.93±0.09	5.81	0.0054
NAR (random) [46]	4236	5.94±0.04	5.90	0.0061
RFGIAug [58]	424	-	5.77	0.0005^{+}
RFGIAug [58]	1000	-	5.80	0.0009^{+}
MbML-NAS(GB) [59]	860	6.74±0.01	-	3.0466
HybridNAS [60]	250	-	5.90	0.0061^{+}
HybridNAS [60]	400	-	5.83	0.0017^{+}
SSENAS [28]	400	6.31±0.11	6.10	0.2809
SSENAS [28]	800	6.11±0.13	5.91	0.0463
SPNAS (Ours)	410	**5.84±0.10**	**5.77**	**0.0017** (**0.0005**$^{+}$)
Oracle	N/A	N/A	5.68	N/A

However, since there were cases in the benchmark datasets in which the optimal individual on the test set performed poorly on the validation set, e.g., NASBench-101, SPNAS was misled to find the optimal solution on the validation set although this solution was ranked second on the test set.

C. Comparison Against Other Methods on NB101, NB201 and NATS

The SPNAS algorithm was further compared against other methods on the EvoXBench benchmark dataset. The comparison was conducted against the modern NAS algorithms listed in Tables III, IV, and V; the best results are highlighted in bold. For each method, the number of function calls to the true objective function (and not the surrogate model) is reported and indicated

TABLE IV
COMPARISON OF DIFFERENT METHODS REGARDING THE OBTAINED ERROR RATE (%) OF THREE DATASETS IN THE SEARCH SPACE OF NASBENCH-201

Methods	#Queries	CIFAR-10		CIFAR-100		ImageNet-16-120	
		Validation	Test	Validation	Test	Validation	Test
RSPS [61]	-	12.40±0.61	8.95±0.66	31.73±0.72	31.74±0.96	60.27±0.34	59.31±0.36
SETN [62]	-	10.00±0.97	7.28±0.73	30.81±1.42	30.64±1.72	60.23±0.33	60.49±0.33
ENAS [52]	-	9.80±0.00	6.24±0.00	29.79±0.71	29.33±0.62	59.22±0.00	58.56±0.00
FairNAS [11]	-	9.93±0.57	6.77±0.18	29.06±0.94	29.00±1.46	58.10±1.00	57.81±0.31
DARTS [63]	-	8.97±0.44	6.20±0.40	28.64±1.51	28.47±1.51	55.13±1.46	54.88±0.82
BOHB [64]	-	8.83±0.27	6.06±0.28	27.96±0.93	28.00±0.86	54.45±0.79	54.30±0.86
REINFORCE [65]	-	8.88±0.25	6.10±0.26	28.20±0.94	28.14±0.89	54.63±0.74	54.36±0.78
GDAS [66]	-	10.32±0.72	6.77±0.58	31.65±2.71	31.83±2.50	60.45±0.00	60.60±0.00
NAR [46]	1000	8.56±0.10	5.67±0.05	27.46±0.44	27.11±0.37	53.84±0.37	**53.34±0.23**
GenNAS-N [57]	1000	-	5.82±0.10	-	27.44±0.74	-	54.41±0.54
NASWOT [67]	1000	10.31±0.73	7.04±0.81	30.14±1.21	30.02±1.22	56.05±2.05	55.56±2.10
MbML-NAS (RF) [59]	860	-	6.64±0.20	-	29.67±0.85	-	57.01±4.21
MbML-NAS (GB) [59]	860	-	6.97±0.52	-	29.98±1.17	-	55.72±1.42
NASWOT [67]	100	10.45±0.89	7.19±0.99	30.65±1.70	30.52±1.70	57.19±3.05	56.90±3.16
Random Search* [68]	100	8.93±0.27	6.18±0.24	28.56±0.83	28.60±0.83	54.63±0.63	54.64±0.67
REA* [69]	100	8.63±0.25	5.94±0.29	27.21±0.69	27.28±0.72	53.85±0.45	54.01±0.51
BANANAS* [25]	100	8.50±0.15	5.77±0.30	26.73±0.57	26.75±0.63	53.55±0.25	53.69±0.31
GP bayesopt* [25]	100	8.55±0.23	5.84±0.31	26.89±0.65	26.95±0.75	53.49±0.25	53.75±0.34
DNGO* [68]	100	8.59±0.17	5.92±0.26	27.29±0.66	27.34±0.67	53.89±0.44	54.00±0.48
Bohamiann* [25]	100	8.59±0.18	5.91±0.26	27.27±0.64	27.35±0.66	53.85±0.45	53.97±0.48
GCN Predictor* [25]	100	8.98±0.32	6.26±0.33	28.57±0.72	28.57±0.77	54.53±0.72	54.64±0.78
BONAS* [68]	100	8.44±0.10	5.68±0.15	26.68±0.40	26.70±0.46	53.44±0.19	53.69±0.30
arch2vec (RL)* [54]	100	8.57±0.28	5.77±0.26	26.88±0.66	26.94±0.87	53.68±0.20	53.65±0.35
arch2vec (BO)* [54]	100	8.49±0.13	5.69±0.14	26.62±0.34	26.56±0.22	53.67±0.21	53.65±0.27
NPENAS-SSRL* [70]	100	8.44±0.14	5.68±0.19	26.53±0.22	26.53±0.30	53.47±0.33	54.17±0.60
NPENAS-CCL* [70]	100	8.43±0.13	5.68±0.19	26.52±0.15	**26.51±0.23**	**53.38±0.34**	54.39±0.41
SAENAS-NE [68]	100	**8.42±0.09**	5.66±0.12	26.54±0.18	26.54±0.20	53.41±0.14	53.64±0.26
ReNAS [45]	90	9.10±0.31	6.01±0.25	28.04±0.99	27.88±0.79	54.15±0.47	54.03±0.49
SPNAS (Ours)	110	8.48±0.06	5.65±0.05	26.72±0.34	26.69±0.38	54.25±0.58	53.79±0.51
SPNAS (Ours)	410	8.43±0.09	**5.63±0.00**	**26.40±0.11**	26.65±0.23	53.58±0.32	53.48±0.32
Oracle	N/A	8.39	5.63	26.51	26.49	53.27	52.69

TABLE V
COMPARISON OF DIFFERENT METHODS REGARDING THE OBTAINED ERROR RATE (%) OF THREE DATASETS IN THE SEARCH SPACE OF NATS

Methods	#Queries	CIFAR-10		CIFAR-100		ImageNet-16-120	
		Validation	Test	Validation	Test	Validation	Test
REA [69]	-	9.63±0.20	6.78±0.16	29.77±0.50	29.89±0.61	54.70±0.69	54.06±0.92
REINFORCE [65]	-	9.75±0.23	6.84±0.21	30.16±0.59	30.04±0.57	54.94±0.77	54.29±0.93
RANDOM	-	9.90±0.26 [35]	6.97±0.25	30.43±0.57	30.28±0.61	54.99±0.74	54.58±0.86
BOHB [64]	-	9.93±0.28	6.99±0.24	30.25±0.60	30.10±0.60	54.89±0.69	54.44±0.81
channel-wise interpolation [35]	-	**9.29±0.00**	**6.60±0.00**	29.70±0.00	29.28±0.00	55.27±0.00	52.83±0.00
masking+Gumbel-Softmax [35]	-	9.59±0.10	6.86±0.13	29.70±0.00	29.28±0.00	54.29±0.39	53.62±0.27
masking+sampling [35]	-	10.27±0.37	7.22±0.30	30.33±0.22	29.89±0.33	55.30±0.60	54.89±0.76
SPNAS	410	9.55±0.07	6.67±0.18	**29.57±0.21**	**29.88±0.34**	**54.65±0.68**	**53.47±0.44**

as #Queries. Table III displays the comparison results of SPNAS against 12 modern NAS algorithms over NASBench-101. Table III lists the mean test error ± standard deviation and the ranking within NASBench-101 of the individual detected. This search space contained 423,624 potential architectures. Note that SPNAS detected the second best solution of the space, thus outperforming all other search algorithms. Since the best ranking was assessed on the validation test, the results on the test set did not fully reflect this ranking. Nonetheless, SPNAS achieved a good performance in terms of test error as well.

To further illustrate the effectiveness of the proposed SPNAS, experiments on NASBench-201 and the search space were conducted, as displayed in Table IV. In this search space, the network was trained on CIFAR-10, CIFAR-100 and ImageNet-16-120. In this search space, SPNAS was able to search on the three benchmark datasets to consistently find the best architecture on both the validation and test sets.

The proposed algorithm achieved excellent results in the NATS space as well, as shown in Table V, identifying the best architectures for CIFAR-100 and ImageNet-16-20 and the second best for CIFAR-10.

D. Transfer to MobileNetV3 With Once-for-All

To further evaluate the effectiveness of the proposed SPNAS framework on practical NAS problems, we extended our experiments to the MobileNetV3 search space. The details of the search process for MobileNetV3, previously described, are not reiterated here. In our experiments, we explored various aspects of the MobileNetV3 architecture, including image input size, convolution kernel size, expansion ratio and block depth. Unlike prior experiments that utilised benchmark results from EvoXBench, this phase involved a direct evaluation of the constructed MobileNetV3 networks. Specifically, we trained (fine-tuned) these networks on a validation dataset to determine their error rates. To optimise evaluation efficiency, we employed pre-trained supernet weights from the Once-for-All framework. Using these weights, we fine-tuned the networks for five epochs

TABLE VI
Comparison With State-of-The-Art Methods on the CIFAR-10 Dataset

Method	Accuracy (%)	Params (M)	Search Cost (GPU Days)	Search Method
NASNet-A [7]	97.35	3.2	1800	RL
ENAS [52]	97.11	4.6	0.5	RL
BlockQNN [71]	97.20	39.8	32	RL
DARTS (first) [6]	97.00	3.3	1.5	GD
DARTS (second) [6]	97.24	3.3	4	GD
PDARTS [72]	97.50	3.4	0.3	GD
PC-DARTS [73]	97.43	3.6	0.1	GD
Proxyless NAS [74]	97.92	5.7	4	GD
SNAS [75]	97.15	2.8	1.5	GD
ADARTS [76]	97.54	2.9	0.2	GD
PA-DARTS [29]	97.42	3.3	0.36	GD
SWD-NAS [77]	97.49	3.17	0.13	GD
SLE-NAS-A [16]	96.01	0.41	0.35	EA
SLE-NAS-B [16]	96.53	0.96	0.4	EA
SaMuNet [78]	96.4	1.5	23	EA
AE-CNN [8]	95.3	2.0	27	EA
AE-CNN+E2EPP [23]	94.7	4.3	7	EA
CNN-GA [9]	96.78	2.9	35	EA
Evo-OSNet [12]	97.44	3.3	0.5	EA
SI-EvoNAS [79]	97.31	1.84	0.458	EA
MSNAS [80]	97.47	3.25	0.23	EA
EffPnet [24]	96.51	2.54	< 3	EA
RelativeNAS [81]	97.66	3.93	0.4	EA
SPNAS (ours)	98.20	6.33	1.4	EA

TABLE VII
Comparison With State-of-The-Art Methods on the CIFAR-100 Dataset

Method	Accuracy (%)	Params (M)	Search Cost (GPU Days)	Search Method
NASNet-A [7]	82.19	3.2	1800	RL
ENAS [52]	81.09	4.6	0.5	RL
PDARTS [72]	84.08	3.6	0.3	GD
PC-DARTS [73]	82.89	3.6	0.1	GD
SNAS [75]	79.91	2.8	1.5	GD
ADARTS [76]	82.94	2.9	0.2	GD
PA-DARTS [29]	83.03	3.3	0.36	GD
SWD-NAS [77]	83.87	3.56	0.13	GD
AE-CNN [8]	77.60	5.4	36	EA
AE-CNN+E2EPP [23]	77.98	20.9	10	EA
SaMuNet [78]	79.8	4.6	25	EA
CNN-GA [9]	79.47	4.1	40	EA
MSNAS [80]	83.59	3.19	0.25	EA
MSNAS (transfer) [80]	82.80	3.25	0.23	EA
SLE-NAS-A [16]	78.76	0.52	0.4	EA
SLE-NAS-B [16]	81.93	0.94	0.4	EA
Evo-OSNet [12]	84.16	3.5	0.5	EA
SI-EvoNAS [79]	84.16	3.5	0.5	EA
EffPnet [24]	81.51	3.98	< 3	EA
RelativeNAS [81]	84.14	3.98	0.4	EA
SPNAS (ours)	87.26	6.74	1.6	EA

TABLE VIII
Comparison With State-of-The-Art Methods on the ImageNet Dataset

Method	Top-1 Acc (%)	Top-5 Acc (%)	Params (M)	Search Cost (GPU Days)	Search Method
NASNet-A [7]	74.0	91.6	5.3	1800	RL
NASNet-B [7]	72.8	91.3	5.3	1800	RL
NASNet-C [7]	72.5	91.0	4.9	1800	RL
SNAS [75]	72.7	90.8	4.3	1.5	GD
PDARTS [72]	75.6	92.6	4.9	0.3	GD
PDARTS [72]	75.3	92.5	5.1	0.3	GD
PC-DARTS [73]	74.9	92.2	5.3	0.1	GD
PC-DARTS [73]	75.8	92.7	5.3	3.8	GD
PA-DARTS [29]	75.3	92.25	5.2	0.4	GD
SWD-NAS [77]	75.5	92.4	6.3	0.13	GD
SLE-NAS-A [16]	74.4	91.8	3.3	3	EA
SLE-NAS-B [16]	75.7	92.5	4.5	3	EA
EPCNAS [82]	72.9	91.5	3.02	1.17	EA
MFENAS [83]	73.94	91.82	5.98	0.6	EA
Evo-OSNet [12]	77.48	93.53	5	8.6	EA
SI-EvoNAS [79]	75.8	92.59	4.7	0.458	EA
EffPnet [24]	72.99	90.75	2.54	< 3	EA
RelativeNAS [81]	75.12	92.30	5.05	0.4	EA
SPNAS (ours)	78.62	94.07	6.64	0.37	EA

TABLE IX
Kendall's Tau ($KTau$) Correlation Index Between the Ground-Truth and Surrogate Ranking and the Number of Queries in Explored Search Space to Train the Surrogate Models on the NASBench-101 Dataset

Method	$KTau$	#Queries
E2EPP [23]	0.5038	424
E2EPP [23]	0.5705±0.0082	1000
Peephole [50]	0.4556	424
Peephole [50]	0.4373±0.0112	1000
NPNAS [41]	0.6945	424
HAAP [56]	0.7010±0.0022	424
HAAP [56]	0.7126±0.0024	1000
SSANA [55]	0.6541±0.0078	1000
ReNAS [45]	0.6574	424
NAO [51]	0.6550	423
SPNAS	0.7022±0.0031	424
SPNAS	0.7502±0.0043	1000

TABLE X
Kendall's Tau ($KTau$) Correlation Coefficient Between the Ground-Truth and Surrogate Ranking for Ablation Experiments on One-Hot Encoding

		$KTau$ (Val)	$KTau$ (Test)
NB201	w/	0.7148±0.0083	0.7193±0.0093
(CIFAR-10)	w/o	0.2905±0.0083	0.2901±0.0097
NB201	w/	0.7248±0.0031	0.7341±0.0030
(CIFAR-100)	w/o	0.2699±0.0096	0.2730±0.0088
NB201	w/	0.7044±0.0067	0.7111±0.0061
(ImageNet16-120)	w/o	0.3372±0.0099	0.3374±0.0109
NATS	w/	0.8449±0.0048	0.8627±0.0026
(CIFAR-10)	w/o	0.8082±0.0093	0.8246±0.0084
NATS	w/	0.8956±0.0013	0.8942±0.0016
(CIFAR-100)	w/o	0.8604±0.0067	0.8550±0.0049
NATS	w/	0.8793±0.0023	0.8855±0.0009
(ImageNet16-120)	w/o	0.8331±0.0089	0.8387±0.0067

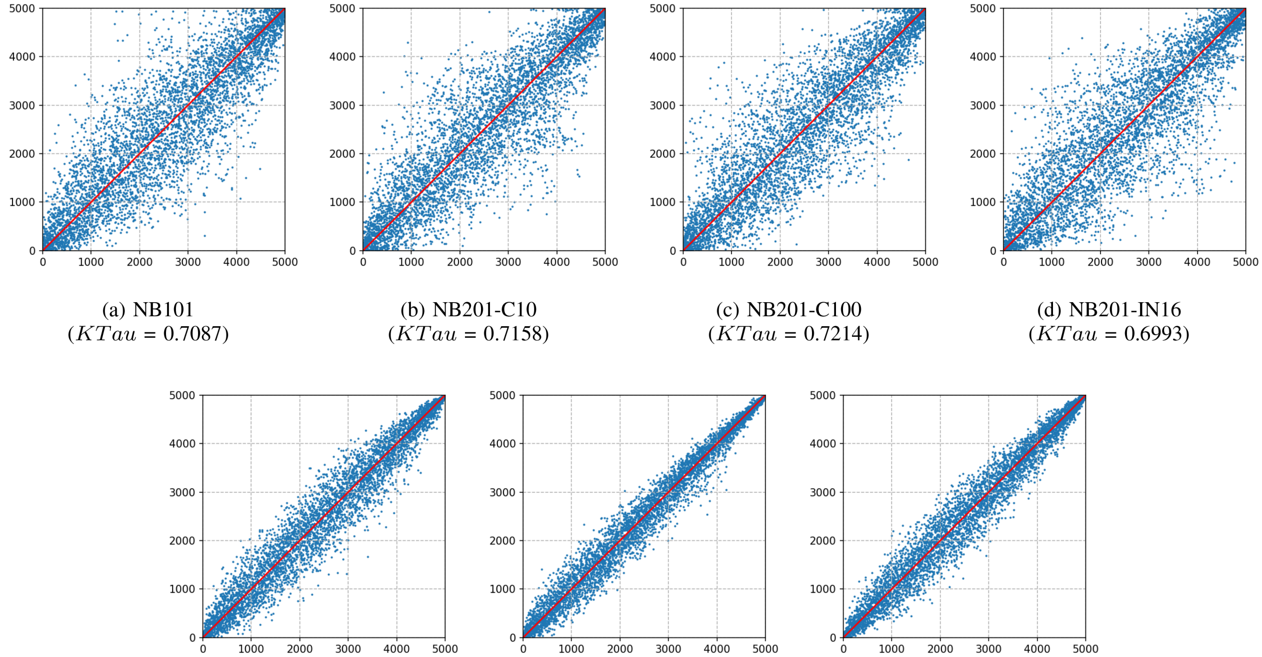
on the CIFAR datasets. For the ImageNet dataset, we leveraged the high performance of Once-for-All by directly applying the pre-trained weights for validation set classification. The hyperparameters used in this phase were consistent with those specified in Section IV-A. The results of these experiments are summarised in Tables VI, VII and VIII. As shown, SPNAS significantly improves classification accuracy for networks in the MobileNetV3 space. Additionally, SPNAS demonstrates a notable advantage in search time on ImageNet compared to other evolutionary algorithms.

E. Analysis and Ablation Study for the Score Predictor

To better understand the functioning of SPNAS, the following experimental study was designed: only random sampling was used to obtain some of the data from the benchmark dataset for training, and 5,000 non-utilised data points were sampled for validation. Using the same mechanism utilised for the score prediction during search, for the training process, only the validation dataset was used. In this scenario, for NASBench-101, Kendall's Tau ($KTau$) correlation coefficient [84] was calculated between the ranking achieved through the ground-truth calculation and that achieved through surrogate for SPNAS and 10 competitor

TABLE XI
KENDALL'S TAU ($KTau$) CORRELATION INDEX BETWEEN THE GROUND-TRUTH AND SURROGATE RANKING FOR DIFFERENT LOSS FUNCTIONS AND THE NUMBER OF TRAINING DATA (#QUERIES) AND THE PERCENTAGE ON NASBENCH-101 BENCHMARK

	212 ($\approx 0.05\%$)		424 ($\approx 0.1\%$)		4236 ($\approx 1\%$)		42362 ($\approx 10\%$)	
	Validation	Test	Validation	Test	Validation	Test	Validation	Test
MSE	0.5420±0.0419	0.5435±0.0412	0.6009±0.0167	0.6018±0.0160	0.6586±0.0135	0.6578±0.0134	0.6988±0.0094	0.6977±0.0091
$\mathcal{L}_1$	0.6138±0.0033	0.6140±0.0037	0.6986±0.0089	0.6949±0.0081	0.8330±0.0012	0.8276±0.0017	0.8801±0.0009	0.8740±0.0014
$\mathcal{L}_1 + \alpha\mathcal{L}_2$	0.6243±0.0092	0.6223±0.0088	0.7040±0.0030	0.7022±0.0031	0.8329±0.0031	0.8283±0.0036	0.8811±0.0020	0.8756±0.0017



(a) NB101
($KTau$ = 0.7087)

(b) NB201-C10
($KTau$ = 0.7158)

(c) NB201-C100
($KTau$ = 0.7214)

(d) NB201-IN16
($KTau$ = 0.6993)

(e) NATS-C10
($KTau$ = 0.8506)

(f) NATS-C100
($KTau$ = 0.8975)

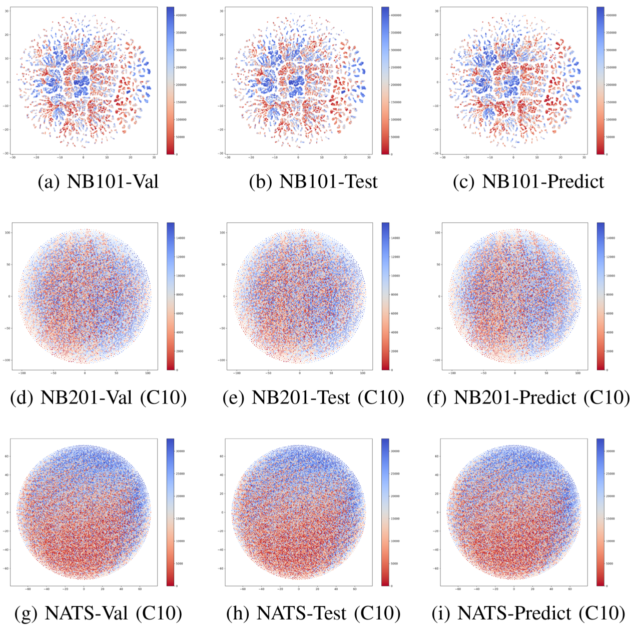
(g) NATS-IN16
($KTau$ = 0.8763)

Fig. 3. Scatterplot of the predicted ranking (x-axis) vs the actual ranking based on the error rate (y-axis). The closer the predicted point is to the red line, the closer the predicted ranking is to the true ranking. Here, 5,000 points randomly chosen from a run were plotted.

algorithms for a surrogate-assisted NAS. All experiments in this section were conducted for five runs, and we have reported the mean and standard deviation in the table. Table IX displays the $KTau$ values in the scenarios considered and the number of queries of search space explored to build the surrogate.

To evaluate the effectiveness of one-hot encoding, we conducted the following ablation study, detailed in Table X. In this study, we used MLPs of the same size to train with both one-hot encoding and standard encoding. The proposed loss function, as described in (5), was employed to train the models, and we then obtained scores from different experiments. Table X presents the $KTau$ correlation coefficient between the rankings of these scores and the ground truth validation/test rankings. The results indicate that one-hot encoding significantly affects NASBench-201, as ordinal encoding introduces incorrect magnitude relations that are unnecessary for representing the type of operation. For the NATS search space, one-hot encoding remains effective. This is because ordinal encoding in NATS represents the number of channels and involves a proportional magnitude relation. This ablation study underscores the importance of using one-hot encoding in our experiments.

The impact of the amount of data and the loss function used for surrogate model training on the prediction performance was also explored. The corresponding results are listed in Table XI. The experiments were conducted using the MSE and $\mathcal{L}_1$ alone, and the corresponding coefficient $KTau$ was calculated. Scrolling the columns, it was observed that the proposed loss function yielded higher $KTau$ values in all scenarios. Scrolling the rows, it was observed that an increase in Per corresponded to an increase in performance index $KTau$. As shown in Table XI, our surrogate model achieves good prediction accuracy with only 0.05% of the data. However, a significant improvement (close to 8%) is observed by doubling the data, so we prefer to use 0.1%. Moving to 1% of the data, representing a tenfold increase, results in only an 11% improvement, indicating that this setting may not be the best trade-off between accuracy and cost. Re-evaluating with 10% of the training data further validates that the surrogate model's accuracy can be enhanced, suggesting that the prediction accuracy does not plateau between using 0.1% and 1%. While there might be a better inflection point between 0.1% and 1%, for the sake of convenience, we have chosen 0.1%.



(a) NB101-Val (b) NB101-Test (c) NB101-Predict

(d) NB201-Val (C10) (e) NB201-Test (C10) (f) NB201-Predict (C10)

(g) NATS-Val (C10) (h) NATS-Test (C10) (i) NATS-Predict (C10)

Fig. 4. Ranking landscape analysis of NASBench-101, NASBench-201 and NATS with t-distributed stochastic neighbour embedding. The first and second columns display the ranking calculated in the validation and testing via using the actual error rate, while the third column displays the ranking data obtained using the SPNAS score predictor.

To further highlight the significance of the data in Table IX, a graphical representation of the prediction results of the models trained on seven benchmark datasets is provided in Fig. 3. What we can see in Fig. 3 is that the score predictor is able to obtain very good prediction accuracy on all 7 datasets. It is worth mentioning that the points on NATS are very close to the red line and all of them obtain a $KTau$ of more than 0.85. The $KTau$ of the validation set accuracy and the test set accuracy obtained by directly training the neural networks in NATS on the three image datasets are 0.8349, 0.9009, and 0.8757. This indicates that the use of our proposed score predictor has been able to completely replace the original training process.

Furthermore, Fig. 4 displays the prediction ability of the proposed score predictor. Here, t-distributed stochastic neighbour embedding [85] was used to reduce the dimensions of the search spaces of NASBench-101, NASBench-201 and NATS and create scatterplots to obtain the ranking landscape of these three benchmark spaces on CIFAR-10. In Fig. 4, the first column represents the validation rankings of all neural networks on CIFAR-10, the second column represents the test ranking of all neural networks, and the third column represents the ranking of all neural networks in terms of the scores obtained using the score predictor. The diagrams in the third column (obtained by the predictor) are similar to those in the other two columns (obtained using the ground-truth fitness). This experiment visually confirmed that the score predictor in SPNAS is reliable and well predicts the ranking based on the actual error rate.

F. Ablation Study on the Surrogate Logic

To demonstrate the viability of the proposed surrogate-assisted logic, we have designed the following ablation study. On NASBench-101 (and thus CIFAR-10), we have designed an algorithm containing all the elements of SPNAS but with a different logic to train and update the surrogate model. More specifically, by taking inspiration from [68], [86], the training and use of the surrogate model are divided into three phases. Initially, in the first phase, lasting only the first generation, I individuals are sampled, and their ground-truth values are obtained to perform an initial training of the surrogate model. Subsequently, in the second phase, every α generations, the best k individuals with the optimal surrogate values are selected to update the surrogate model. The training dataset associated with the surrogate update has a maximum size limit of L (i.e., the total number of queries). If this limit is reached, no further real evaluations are conducted, and in the third phase, the surrogate model is no longer updated. It must be remarked that even though the ground-truth values are determined in the first two phases to update the surrogate model, all the selection operations within the GA framework are conducted based on the values predicted by the surrogate model.

TABLE XII
Ablation Study on the Surrogate-Assisted Logic: The Error Rate and Ranking (%) of SPNAS Surrogate Update Logic are Compared With Four Scenarios Inspired by [68], [86]

	$I = 40$, $k = 10$, $\alpha = 1$, $L = 420$	$I = 40$, $k = 20$, $\alpha = 2$, $L = 420$	$I = 100$, $k = 10$, $\alpha = 1$, $L = 420$	$I = 100$, $k = 20$, $\alpha = 2$, $L = 420$	SPNAS (ours)
Validation	5.68±0.20	5.43±0.18	5.45±0.22	5.25±0.16	**5.10±0.19**
Test	6.27±0.25	6.08±0.21	6.08±0.40	5.95±0.30	**5.84±0.10**
Ranking (%)	0.2066	0.0321	0.0321	0.0071	**0.0017**

By employing this three-phase structure we have created four variants of the proposed SPNAS running in four scenarios. The results of this ablation study are presented in Table XII. It is evident that the best results inspired by the approach proposed in [68], [86] are achieved for $I = 100$, $k = 20$ and $\alpha = 2$. However, in this case, numerical results show that our SPNAS approach outperforms these results.

V. Conclusion

This paper proposed an ENAS method assisted by a surrogate scoring algorithm that allows for easy and computationally cheap population ranking. The proposed ENAS framework searches for the optimal architecture via employing a GA-like search logic. In the early stages of the evolution, it uses the ground-truth fitness function on a small population and collects the data, which are then used to train an MLP network specifically designed to assign scores to the candidate architectures that enable easy population ranking without the need for calculating or estimating the error rate. In the following part of the search, the proposed algorithm evolves a large population using only the surrogate function.

The results of the experiments conducted on all seven search spaces of the EvoXBench library illustrated that the proposed algorithm can detect excellent solutions across the entire benchmark, thus displaying versatility. A comparison against a large number of modern NAS algorithms showed that the algorithm has excellent performance and achieved the best network design

TABLE XIII
CORRELATION COEFFICIENT $KTau$ BETWEEN THE TRUE FITNESS AND PREDICTED VALUES FOR FOUR WELL-KNOWN TEST PROBLEMS, COMPARING THE PROPOSED SP, A STANDARD RBF AND A MODERN SURROGATE ENSEMBLE METHOD [87] THAT REPRESENTS THE STATE-OF-THE-ART SURROGATE OPTIMISATION

Problem	Dimension	RBF	Ensemble [87]	SP (Ours)
Griewank	40	0.8458±0.0082	0.8961±0.0046	**0.9277±0.0057**
	80	0.6345±0.0100	0.7773±0.0087	**0.8103±0.0072**
	120	0.3808±0.0522	0.5745±0.0210	**0.5876±0.0373**
Ackley	40	0.6982±0.0100	0.7229±0.0053	**0.7676±0.0874**
	80	0.4949±0.0360	0.6142±0.0203	**0.6314±0.0393**
	120	0.2962±0.0131	0.4728±0.0228	**0.4747±0.0301**
Rastrigin	40	0.7376±0.0258	0.7571±0.0139	**0.8081±0.0432**
	80	0.5137±0.0503	0.6502±0.0164	**0.6898±0.0302**
	120	0.2773±0.0231	0.4437±0.0279	**0.4515±0.0298**
Rosenbrock	40	0.7611±0.0064	0.7876±0.0037	**0.8099±0.0065**
	80	0.5630±0.0394	0.6882±0.0170	**0.6932±0.0386**
	120	0.4089±0.0400	0.5528±0.0194	**0.5706±0.0501**

in almost all cases considered. The experimental studies on the score predictor's functioning demonstrated that the proposed surrogate ranked the population by reliably simulating the ranking according to the ground-truth fitness.

Future work will further investigate search operators and domain-specific constraint handling techniques to ensure that offspring solutions are always meaningful architectures. Another direction of the future research will design the surrogate model as a multi-objective optimisation problem where accuracy and computational cost will be simultaneously addressed. Meanwhile, it would be beneficial to extend this approach to more general and complex black-box optimisation problems, including more objectives and dimensions. Furthermore, a more general way to utilize surrogate models is to adjust the surrogate model by selecting elite individuals for real evaluation based on surrogate values during the search process. Therefore, it is highly worthwhile to consider how to integrate the methods presented in this paper with this more general approach. Furthermore, few-shot learning approaches including e.g., transfer learning, metric learning, and Siamese neural networks will be explored to train the surrogate.

APPENDIX

To demonstrate the generalisability of the proposed score predictor (SP), we applied it to four commonly used benchmark problems — Griewank, Ackley, Rastrigin and Rosenbrock (corresponding to f_2–f_5 from [87]) — and conducted experiments across multiple dimensions: 40, 80 and 120. In each of the twelve scenarios, we trained the SP, the ensemble approach proposed in the state-of-the-art surrogate method from Ensemble Surrogate-based Coevolutionary Optimizer (ESCO) [87] and a standard radial basis function (RBF). Specifically, we used 400 samples (true fitness evaluations) to build each surrogate model and then used 5,000 predictions to assess the reliability of each model's ranking capability. Table XIII displays the correlation coefficient $KTau$ between the true and predicted rankings. The results show that the score predictor offers a significant advantage. However, it should be noted that as the dimensionality increases, the advantage of the score predictor becomes less pronounced.

REFERENCES

[1] J. Du, K. Guan, Y. Zhou, Y. Li, and T. Wang, "Parameter-free similarity-aware attention module for medical image classification and segmentation," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 3, pp. 845–857, Jun. 2023.

[2] A. Pramanik, S. K. Pal, J. Maiti, and P. Mitra, "Granulated RCNN and multi-class deep sort for multi-object detection and tracking," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 1, pp. 171–181, Feb. 2022.

[3] R. Cong, W. Song, J. Lei, G. Yue, Y. Zhao, and S. Kwong, "PSNet: Parallel symmetric network for video salient object detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 2, pp. 402–414, Apr. 2023.

[4] S. Wen, M. Dong, Y. Yang, P. Zhou, T. Huang, and Y. Chen, "End-to-end detection-segmentation system for face labeling," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 5, no. 3, pp. 457–467, Jun. 2021.

[5] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 550–570, Feb. 2023.

[6] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–13.

[7] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. 2018 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8710.

[8] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1242–1254, Apr. 2020.

[9] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020.

[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] X. Chu, B. Zhang, and R. Xu, "FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 12219–12228.

[12] H. Zhang, Y. Jin, and K. Hao, "Evolutionary search for complete neural network architectures with partial weight sharing," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 1072–1086, Oct. 2022.

[13] A. Brock, T. Lim, J. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–22.

[14] Z. Guo et al., "Single path one-shot neural architecture search with uniform sampling," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 544–560.

[15] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 35–51.

[16] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Split-level evolutionary neural architecture search with elite weight inheritance," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 10, pp. 13523–13537, Oct. 2024.

[17] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 550–559.

[18] P. Dong et al., "RD-NAS: Enhancing one-shot supernet ranking ability via ranking distillation from zero-cost proxies," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2023, pp. 1–5.

[19] H. Wang, C. Ge, H. Chen, and X. Sun, "PreNAS: Preferred one-shot learning towards efficient neural architecture search," in *Proc. Int. Conf. Mach. Learn.*, vol. 202, 2023, pp. 35642–35654.

[20] M. Zhang, X. Yu, H. Zhao, and L. Ou, "ShiftNAS: Improving one-shot nas via probability shift," in *Proc. 2023 IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 5896–5905.

[21] Y. Liu, J. Liu, Y. Jin, F. Li, and T. Zheng, "A surrogate-assisted two-stage differential evolution for expensive constrained optimization," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 3, pp. 715–730, Jun. 2023.

[22] S. Liu, H. Zhang, and Y. Jin, "A survey on computationally efficient neural architecture search," *J. Automat. Intell.*, vol. 1, no. 1, 2022, Art. no. 100002.

[23] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 350–364, Apr. 2020.

[24] B. Wang, B. Xue, and M. Zhang, "Surrogate-assisted particle swarm optimization for evolving variable-length transferable blocks for image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 3727–3740, Aug. 2022.

[25] C. White, W. Neiswanger, and Y. Savani, "BANANAS: Bayesian optimization with neural architectures for neural architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 10293–10301.

[26] S. Hassantabar, X. Dai, and N. K. Jha, "CURIOUS: Efficient neural architecture search based on a performance predictor and evolutionary search," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4975–4990, Nov. 2022.

[27] 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.*, vol. 4, no. 6, pp. 1602–1615, Dec. 2023.

[28] Y. Xue, Z. Zhang, and F. Neri, "Similarity surrogate-assisted evolutionary neural architecture search with dual encoding strategy," *Electron. Res. Arch.*, vol. 32, no. 2, pp. 1017–1043, 2024.

[29] Y. Xue, C. Lu, F. Neri, and J. Qin, "Improved differentiable architecture search with multi-stage progressive partial channel connections," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 8, no. 1, pp. 32–43, Feb. 2024.

[30] Z. Lu, R. Cheng, Y. Jin, K. C. Tan, and K. Deb, "Neural architecture search as multiobjective optimization benchmarks: Problem formulation and performance assessment," *IEEE Trans. Evol. Comput.*, vol. 28, no. 2, pp. 323–337, Apr. 2024.

[31] 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.*, vol. 97, 2019, pp. 7105–7114.

[32] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–16.

[33] C. Li et al., "HW-NAS-Bench: Hardware-aware neural architecture search benchmark," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–18.

[34] 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. Representations*, 2021, pp. 1–36.

[35] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-Bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3634–3646, Jul. 2022.

[36] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–15.

[37] M. Chen, H. Peng, J. Fu, and H. Ling, "AutoFormer: Searching transformers for visual recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 12250–12260.

[38] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter, "Learning curve prediction with Bayesian neural networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–16.

[39] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–19.

[40] L. Dudziak, T. Chau, M. Abdelfattah, R. Lee, H. Kim, and N. Lane, "BRP-NAS: Prediction-based NAS using GCNs," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 10480–10490, 2020.

[41] W. Wen, H. Liu, Y. Chen, H. Li, G. Bender, and P.-J. Kindermans, "Neural predictor for neural architecture search," in *Eur. Proc. Conf. Comput. Vis.*, 2020, pp. 660–676.

[42] Y. Chen et al., "Contrastive neural architecture search with neural architecture comparators," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 9497–9506.

[43] J. Wu et al., "Stronger NAS with weaker predictors," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 28904–28918, 2021.

[44] X. Zheng, R. Ji, L. Tang, B. Zhang, J. Liu, and Q. Tian, "Multinomial distribution learning for effective neural architecture search," in *Proc. 2019 IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1304–1313.

[45] Y. Xu et al., "ReNAS: Relativistic evaluation of neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4409–4418.

[46] B. Guo et al., "Generalized global ranking-aware neural architecture ranker for efficient image classifier search," in *Proc. ACM Int. Conf. Multimedia*, 2022, pp. 3730–3741.

[47] C. Hu et al., "RankNAS: Efficient neural architecture search by pairwise ranking," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 2469–2480.

[48] P. Jiang, Y. Xue, and F. Neri, "Convolutional neural network pruning based on multi-objective feature map selection for image classification," *Appl. Soft Comput.*, vol. 139, 2023, Art. no. 110229.

[49] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.

[50] B. Deng, J. Yan, and D. Lin, "Peephole: Predicting network performance before training," 2017, *arXiv:1712.03351*.

[51] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 7827–7838, 2018.

[52] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 4095–4104.

[53] B. Wu et al., "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10726–10734.

[54] S. Yan, Y. Zheng, W. Ao, X. Zeng, and M. Zhang, "Does unsupervised architecture representation learning help neural architecture search?," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 12486–12498, 2020.

[55] Y. Tang et al., "A semi-supervised assessor of neural architectures," in *Proc. 2020 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1807–1816.

[56] Y. Liu, Y. Tang, and Y. Sun, "Homogeneous architecture augmentation for neural predictor," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 12229–12238.

[57] Y. Li, C. Hao, P. Li, J. Xiong, and D. Chen, "Generic neural architecture search via regression," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 20476–20490, 2021.

[58] X. Xie, Y. Sun, Y. Liu, M. Zhang, and K. C. Tan, "Architecture augmentation for performance predictor via graph isomorphism," *IEEE Trans. Cybern.*, vol. 54, no. 3, pp. 1828–1840, Mar. 2024.

[59] G. T. Pereira, I. B. Santos, L. P. Garcia, T. Urruty, M. Visani, and A. C. De Carvalho, "Neural architecture search with interpretable meta-features and fast predictors," *Inf. Sci.*, vol. 649, 2023, Art. no. 119642.

[60] Z. Xun, L. Songbai, W. Ka-Chun, L. Qiuzhen, and T. Kaychen, "A hybrid search method for accelerating convolutional neural architecture search," in *Proc. Int. Conf. Mach. Learn. Comput.*, 2023, pp. 177–182.

[61] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. Uncertainty Artif. Intell. Conf.*, 2020, pp. 367–377.

[62] X. Dong and Y. Yang, "One-shot neural architecture search via self-evaluated template network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3680–3689.

[63] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, and J. Yan, "DARTS-: Robustly stepping out of performance collapse without indicators," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–22.

[64] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 1437–1446.

[65] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–16.

[66] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1761–1770.

[67] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, 2021, pp. 7588–7598.

[68] L. Fan and H. Wang, "Surrogate-assisted evolutionary neural architecture search with network embedding," *Complex Intell. Syst.*, vol. 9, no. 3, pp. 3313–3331, 2023.

[69] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 4780–4789.

[70] C. Wei, Y. Tang, C. N. Chuang Niu, H. Hu, Y. Wang, and J. Liang, "Self-supervised representation learning for evolutionary neural architecture search," *IEEE Comput. Intell. Mag.*, vol. 16, no. 3, pp. 33–49, Aug. 2021.

[71] Z. Zhong et al., "BlockQNN: Efficient block-wise neural network architecture generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 7, pp. 2314–2328, Jul. 2021.

[72] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. 2019 IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1294–1303.

[73] Y. Xu et al., "PC-DARTS: Partial channel connections for memory-efficient architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–13.

[74] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–13.

[75] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–17.
[76] Y. Xue and J. Qin, "Partial connection based on channel attention for differentiable neural architecture search," *IEEE Trans. Ind. Inform.*, vol. 19, no. 5, pp. 6804–6813, May 2023.
[77] Y. Xue, X. Han, and Z. Wang, "Self-adaptive weight based on dual-attention for differentiable neural architecture search," *IEEE Trans. Ind. Inform.*, vol. 20, no. 4, pp. 6394–6403, Apr. 2024.
[78] Y. Xue, Y. Wang, J. Liang, and A. Slowik, "A self-adaptive mutation neural architecture search algorithm based on blocks," *IEEE Comput. Intell. Mag.*, vol. 16, no. 3, pp. 67–78, Aug. 2021.
[79] H. Zhang, Y. Jin, R. Cheng, and K. Hao, "Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance," *IEEE Trans. Evol. Comput.*, vol. 25, no. 2, pp. 371–385, Apr. 2021.
[80] J. Dong, B. Hou, L. Feng, H. Tang, K. C. Tan, and Y.-S. Ong, "A cell-based fast memetic algorithm for automated convolutional neural architecture design," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 9040–9053, Nov. 2023.
[81] H. Tan et al., "RelativeNAS: Relative neural architecture search via slow-fast learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 475–489, Jan. 2023.
[82] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Particle swarm optimization for compact neural architecture search for image classification," *IEEE Trans. Evol. Comput.*, vol. 27, no. 5, pp. 1298–1312, Oct. 2023.
[83] S. Yang, Y. Tian, X. Xiang, S. Peng, and X. Zhang, "Accelerating evolutionary neural architecture search via multifidelity evaluation," *IEEE Trans. Cogn. Devel. Syst.*, vol. 14, no. 4, pp. 1778–1792, Dec. 2022.
[84] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–89, 1938.
[85] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.
[86] C. Wei, C. Niu, Y. Tang, Y. Wang, H. Hu, and J. Liang, "NPENAS: Neural predictor guided evolution for neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 8441–8455, Nov. 2023.
[87] X. Wu, Q. Lin, J. Li, K. C. Tan, and V. C. M. Leung, "An ensemble surrogate-based coevolutionary algorithm for solving large-scale expensive optimization problems," *IEEE Trans. Cybern.*, vol. 53, no. 9, pp. 5854–5866, Sep. 2023.

Pengcheng Jiang received the B.E. degree in 2020 from the Nanjing University of Information Science and Technology, Nanjing, China, where he is currently working toward the Ph.D. degree with the School of Software. His research interests include feature selection, neural architecture search, model compression, and large machine learning models.

Yu Xue (Senior Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2013. From 2016 to 2017, he was a Visiting Scholar with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. From 2017 to 2018, he was a Research Scholar with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA. He is currently a Professor with the School of Software, Nanjing University of Information Science and Technology, Nanjing. His research interests include deep learning, evolutionary computation, machine learning, computer vision, and feature map selection.

Ferrante Neri (Senior Member, IEEE) received Laurea and Ph.D. degrees in electrical engineering from the Technical University of Bari, Bari, Italy, in 2002 and 2007, respectively, the second Ph.D. degree in scientific computing and optimisation and the D.Sc. degree in computational intelligence from the University of Jyväskylä, Jyväskylä, Finland, in 2007 and 2010, respectively. He is currently a Full Professor of machine learning and artificial intelligence with the University of Surrey, Guildford, U.K. He is also an Associate Dean (International) with the Faculty of Engineering and Physical Sciences, University of Surrey. He is also a Jiangsu Distinguished Professor in China. His research interests include metaheuristic optimisation with applications in machine learning.