A Survey on Computationally Efficient Neural Architecture Search

Shiqing Liu, Haoyu Zhang, and Yaochu Jin, *Fellow, IEEE*

Abstract—Neural architecture search (NAS) has become increasingly popular in the deep learning community recently, mainly because it can provide an opportunity to allow interested users without rich expertise to benefit from the success of deep neural networks (DNNs). However, NAS is still laborious and time-consuming because a large number of performance estimations are required during the search process of NAS, and training DNNs is computationally intensive. To solve this major limitation of NAS, improving the computational efficiency is essential in the design of NAS. However, a systematic overview of computationally efficient NAS (CE-NAS) methods still lacks. To fill this gap, we provide a comprehensive survey of the state-of-the-art on CE-NAS by categorizing the existing work into proxy-based and surrogate-assisted NAS methods, together with a thorough discussion of their design principles and a quantitative comparison of their performances and computational complexities. The remaining challenges and open research questions are also discussed, and promising research topics in this emerging field are suggested.

Index Terms—Neural architecture search (NAS), one-shot NAS, surrogate model, Bayesian optimization, performance predictor.

I. INTRODUCTION

Deep learning has played an important role in the area of machine learning. Currently, deep learning has been successfully applied to computer vision, including image classification [1–5], object detection [6–9], boundary detection [10], semantic segmentation [11–13], pose estimation [14], among many others. The technical design of deep learning heavily relies on DNNs because it automates the feature engineering process. The architectures of DNNs are usually developed for specific tasks and the associated weights can be obtained by a learning process. Only both are optimal at the same time can DNNs achieve promising performance. However, manually designing promising architectures of DNNs is a tedious task, mainly because it usually requires rich expertise in both deep learning and the investigated problems, and tries a great number of different hyperparameters to long-time tuning. These manually designed task-specific networks can not generalize to various application areas. For example, a network architecture designed for the image classification task may obtain inferior performance in object detection tasks.

S. Liu and Y. Jin are with the Chair of Nature Inspired Computing and Engineering, Faculty of Technology, Bielefeld University, 33619 Bielefeld, Germany. Y. Jin is also with the Department of Computer Science, University of Surrey, Guildford, GU2 7XH, United Kingdom. Email: yaochu.jin@uni-bielefeld.de.

H. Zhang is with the Engineering Research Center of Digitized Textile & Apparel Technology, Ministry of Education, College of Information Science and Technology, Donghua University, Shanghai 201620, China.

Moreover, it is sometimes needed to design network architectures under a limited computational budget (latency, memory, FLOPs, etc.) for different deployment scenarios. Handcrafted network architecture design is often inefficient to explore a large number of possibilities.

Recently, NAS appeared as a practical tool that allows engineers and researchers without expertise in deep learning to benefit from the success of DNNs. NAS focuses on searching effective task-specific network architectures on given datasets in an automatic manner. By defining a search space, which contains a large set of possible candidate network architectures, NAS can adopt the search strategies to explore extensive neural network architectures that have never been designed before. Most recently, compared with manually designed networks, NAS has obtained remarkable performance gain on representative benchmark datasets such as CIFAR10 [15], CIFAR100 [15], ImageNet [16], etc., in terms of accuracy, model size, and computational complexity [17–21].

The purpose of NAS is to search for a network that minimizes some performance measures, such as error rates on unseen data (validation dataset). To guide the search process of NAS, early work [22–25] usually adopt the simplest way to evaluate the performance of network candidates. A large number of networks are sampled from the search space and trained on training data from scratch, before their performance is evaluated on the validation dataset. Since training DNNs is itself computationally expensive, early NAS methods suffer from a high computational burden. For example, Zoph et al. [22] use reinforcement learning (RL) to design neural networks on the CIFAR10 dataset, which consumes 28 days on 800 high-performance graphics process units (GPU) cards. Unfortunately, not every researcher has access to sufficient computing resources. Considerable computational overhead in evaluating network performance has been the bottleneck in the real-world application of NAS. To address the above issue, researchers have made efforts on speeding up performance estimation. Consequently, the search time has been reduced from many GPU days down to several GPU hours. For example, recent work [26] allows a promising network architecture that can be searched within 0.1 GPU days on CIFAR10.

Little work have been reported to systematically review this emerging field. Therefore, this paper focuses on providing an overview of the research on improving the search efficiency of NAS methodologies. Based on whether the weights of network candidates are required when these network candidates are evaluated, they can be divided into two different categories, i.e., evaluating network candidates under the proxy metrics and surrogate-assisted neural architecture search. For proxy-based

methods, the metrics still need the weights of architectures to be provided, which may introduce additional computational cost. In contrast to the proxy-based methods, the performance evaluation of surrogate-assisted NAS methods only depends on the architecture of the network itself. Surrogate-based NAS methods usually rely on surrogates (also called performance predictor) to predict the performance of candidate networks, thereby avoiding the additional computational overhead. Training surrogates efficiently remains a challenging topic in NAS.

The rest of this paper is organized as follows. Section II provides the definition and mathematical formulation of NAS as an optimization problem, along with a brief overview of the development of NAS methods. Section III gives a detailed investigation of proxy-based NAS, which covers low-fidelity estimation, one-shot NAS and network morphism. Section IV presents a systematical analysis of existing surrogate-assisted NAS methods, including Bayesian optimization based methods, surrogate-assisted evolutionary based algorithms, federated NAS, and multi-objective NAS. Section V summarizes the existing challenges and provides some insights into the future directions.

II. NAS

NAS aims to search task-specific neural network architectures with high performance for a target dataset $D = \{D_{tra}, D_{val}, D_{test}\}$ and releases engineers from the tremendous tedious network architecture designing process. NAS process can be modeled by a bilevel optimization problem, which can be formulated as follows:

$$W_A^* = \arg\min_{A \in S} \mathcal{L}_{tra}(N(A, W_A), D_{tra}), \tag{1}$$

$$A^* = \arg\min_{W} \mathcal{L}_{val}(N(A, W_A^*), D_{val}). \tag{2}$$

In general, S denotes the search space of the network architectures. $N(A, W_A)$ denotes the candidate architecture in the search space, where W_A denotes the parameters associated to the network A. The goal of NAS is to search the network $A \in S$ that can achieve the promising performance on the validation set D_{val} via minimizes the validation loss $\mathcal{L}_{val}$ according to Equation (2), and the parameters W_A^* can be obtained through training model A on the training set D_{tra} via minimizing the loss function $\mathcal{L}_{tra}$ according to Equation (1).

NAS search space is used to collect all possible candidate network architectures. Hence, the search space of NAS has a profound influence on the search efficiency and the performance of the designed models. In general, the search space can be divided into macro search space and micro search space. As shown in Fig.1(a), the macro search space is proposed in the algorithm [22] by Google, which is over the entire network architecture, such as the number of layers n, the link manners for connections (e.g. shortcut [1]), operation types (e.g. convolution and pooling), among others. As shown in Fig.1(b), the micro search space is proposed in the algorithm [23], only covers repeated blocks in the whole network architecture. These blocks are constructed by complex multi-branch operations.
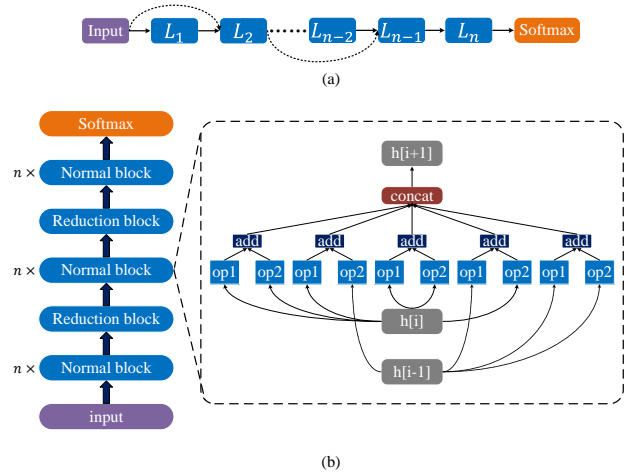

Fig. 1. (a) An example of a network architecture represented in a macro search space. (b) An example of a network architecture represented in a micro search space. A typical example of a normal block structure is shown in the dashed box. Each block obtains the outputs from the previous block h[i] and previous–previous block h[$i-1$] as its inputs. The outputs of h[i] and h[$i-1$] are connected to operations (denoted as "op").

In theory, NAS can be seen as a complex optimization problem, which faces multiple challenges such as multiple conflicting optimization objectives, complex constraints, bi-level structures, expensive computational properties, among others. Early NAS research relies on evolutionary algorithms (EAs) and reinforcement learning (RL) to design the optimal network architecture for the given data set.
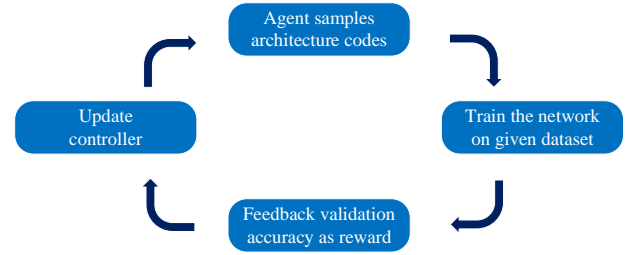
A. RL-based NAS

Fig. 2. An overall framework of RL-based NAS algorithms.

RL-based NAS methods consider the process of the design of network architecture as an agent's action and train a meta-controller as a navigating tool to guide the search process. The overall framework of RL-based NAS algorithm is shown in Fig.2. More specifically, A new candidate network is sampled by the meta-controller and trained on the given training data. The performance of the sampled network on the validation dataset is used as a reward score for updating the controller to sample better candidate networks from the search space in the next iteration.

A policy gradient method aims to approximate non-differentiable reward functions to train a model that needs parameter gradients (e.g. a network architecture). Zoph et al. [27] used a recurrent neural network (RNN) policy controller trained with policy gradients to generate a sequence of actions

to design a network architecture. The original algorithm in [22] is performed on macro search space, which designs the entire network architecture at once. Such a huge search space results in a prohibitive computational cost. For example, the algorithm [22] is performed on 800 graphics processing unit cards (GPUs) in 28 days (22400 GPU-days) on CIFAR10 dataset. To alleviate the computational burden, Zoph et al. [23] proposed micro search space and adopted proximal policy optimization to optimize the RNN controller. Since the micro search space greatly reduces the size of architecture search space, the original algorithm in [23] consumes 1800 GPU-days on CIFAR10 dataset. The Mnas algorithm [17] proposed a factorized hierarchical search space that can not only enable layer diversity but also strike balance between the size of search space and flexibility. In addition, Mnas follows the same search strategy as in [23] that automatically searches models to maximize the accuracy and minimize the real-world latency on mobile devices.

Q-learning [28] is another class of popular RL-based NAS methods. The MetaQNN algorithm [22] adopted Q-learning to optimize a policy that can sequentially select a type of layer's operation and hyperparameters of the network architecture. Zhong et al. [29, 30] leveraged Q-learning with the epsilon-greedy strategy to search architectural building blocks. The building block is repeated several times and stacked sequentially to generate deeper network architectures for evaluation.
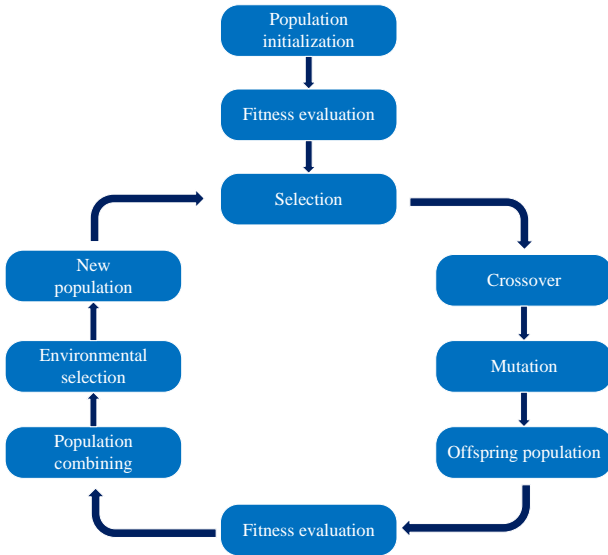
B. EA-based NAS

Fig. 3. An overall framework of EA-based NAS algorithm.

EA-based NAS is another stream of NAS methods. EA is a type of population-based heuristic computational paradigm [31]. The individual inside the population represents a candidate solution to the investigated problem. In EA-based NAS, the evolutionary algorithm is adopted as the search strategy to search for the best network architecture. Hence, each individual in the population should represent a candidate network architecture. With the evolution of the population, the performance of the network architecture is getting better and better on the given datasets. The overall framework of EA-based NAS algorithm is shown in Fig.3.

The workflow of EA-based NAS follows several steps: firstly, a population requires to be randomly initialized with the pre-defined population size by the associated phenotype-to-genotype mapping strategy. Each individual is decoded to a neural network and iteratively trained on the given train dataset for several epochs. The fitness value of the individual is the accuracy of the validation dataset. All individuals will participate in the evolutionary process. Secondly, the selection strategy (e.g. tournament selection [32]) is adopted to choose the parent individuals according to the fitness value, and the crossover and mutation operators are applied to the chosen parents to generate the new offspring until achieving the pre-defined size. Thirdly, the environmental selection is performed on the combined population to select the better individuals surviving into the next generation. This evolutionary process is repeated until the pre-defined termination condition is satisfied.

Traditional EA-based NAS methods, from the 1980s, used EAs to optimize the topology, weights, and hyperparameters of artificial neural networks, which is called neuroevolution [33–36]. Since the architecture of DNNs is more complex and has a large number of hyperparameters and connection weights, such traditional methods are not well suited for optimizing DNNs. As a result, recently, EA-based NAS methods only focus on optimizing the architecture of DNNs. The optimal weights for each candidate network are usually obtained by the gradient-based optimization algorithm [37].

Early work on optimizing CNNs, i.e. Genetic CNN [38], CoDeepNEAT [39, 40], CGP-CNN [41], and CNF [42], have shown powerful performance. Researchers found that EAs can generate high-quality optimal solutions by using biologically inspired operators, including selection, crossover and mutation [43]. Real et al. [24] adopted a variable-length encoding strategy to represent the architecture of CNNs and proposed a novel and intuitive mutation operators to explore the search space. EvoCNN algorithm [44] used EAs to evolve the network architectures and corresponding initial connection weights at the same time. More effective initial connection weights can avoid neural networks falling into local optimal solutions. Sun et al. [45, 46] proposed a variable-length encoding strategy to search the optimal depth of the CNN architecture through the basic genetic algorithm (GA). Damien et al. [47] designed a new EA-based NAS framework, Matrix Evolution for High Dimensional Skip-connection Structures (ME-HDSS), to automatically remove the skip-connection structures in the DenseNet [2] model to further reduce the trainable weights and increase the performance of the model. Zhang et al. [48] adopted I-Ching Divination Evolutionary Algorithm (IDEA) to optimize the complete network architecture, including the number of layers, the number of channels, and connections between different layers. In addition, to improve the search efficiency, the reinforced-based operator controller is developed to choose the different operators of IDEA. Liu et al. [49] introduced multiple latency constraints in the architecture search process and then proposed latency EvoNAS (LEvoNAS) for optimizing network architecture. Sun et al. [50]

adopted EAs to design unsupervised DNNs for efficiently learning meaningful representations.

In fact, real-world tasks also need to consider multiple conflicting objectives, including floating-point operations per second (FLOPs) [51–53], latency [53, 54], memory consumption [53, 55], inference time [56], among others. For example, the low power consumption and the high performance of the model in mobile applications. Compared with RL algorithm, EAs are more suited to solve multiobjective optimization problems in NAS. NEMO [56] is one of the early studies using the elitist non-dominated sorting genetic algorithm (NSGA-II) [57] to minimize the inference time of a network and maximize the classification performance. Lu et al. [51, 52] proposed NSGANetV1 method, which is formulated to design networks based on marco search space and micro search space with high classification performance and lower FLOPs using the NSGA-II algorithm. Wen et al. [58] proposed a two-stage multi-objective EA-based NAS algorithm that can optimize the network architecture in transfer learning. In addition, multiobjective EA-based NAS also can be used in the federated learning community to reduce communication costs [59, 60]. Note that federated neural architecture search is also a promising emerging research topic [61].

III. PROXY METHODS IN NAS

The search procedure itself is laborious mainly because the training and evaluation candidate networks over a large search space are time-consuming. Therefore, more recently, many algorithms have been proposed for reducing computation costs and improving the efficiency of NAS with proxy methods such as low-fidelity estimation [25, 29, 30, 62, 63], one-shot NAS [19, 60, 64], and network morphism [65, 66].

A. Low-fidelity estimation

Early methods in NAS have attempted to accelerate candidate neural network training and evaluation by low-fidelity estimation, such as adopting shorter training times (also denoted as early stopping strategy) [22, 30, 67, 68], using lower-resolution of input images [69], starting with a small-scale dataset [23], using a subset of the full training set [70–72], and downscaling the size of candidate networks (e.g., reducing the channels of candidate neural network) [25, 71, 73]. The low-fidelity estimation is adopted as surrogate measurements to guide the network architecture search process. Compared with full training optimization, low-fidelity estimation needs an order of magnitude fewer computation costs. NASNet [23] designed small top-promising building blocks on a small-scale dataset to reduce the search cost. The optimized block module usually owns competitive generalization capabilities and can be transferred between different datasets or computer vision tasks. For example, the block networks designed on the CIFAR10 dataset can also achieve competitive performance in the ImageNet classification task [16] and MS-COCO object detection task [74]. Zhong et al. [30] used an early stopping strategy to enable the search strategy with fast convergence and reduce the search costs to 20 GPU-hours on the CIFAR10 dataset.

Although such low-fidelity estimation methods can save the search time, recent studies [62, 75] indicated that they can lead to inaccurate evaluation of candidate networks, especially for complicated and large network architecture. For example, NASNet [23] added an additional reranking stage before choosing the best network architecture, which trains the top 250 promising networks for 300 epochs each. The best network was ranked the 70th among 250 top promising neural networks according to performance ranking at low-fidelity estimation. Hence, simple low-fidelity estimation methods may result in low correlation in the prediction of the performance of the network. Zhou et al. [62] also discussed this phenomenon and studied the impact of different low-fidelity estimation methods on the performance ranking of neural network architectures. Damien et al. [76] provided an analysis of the correlation between different lower fidelity estimation methods and final test performance.

B. One-shot NAS
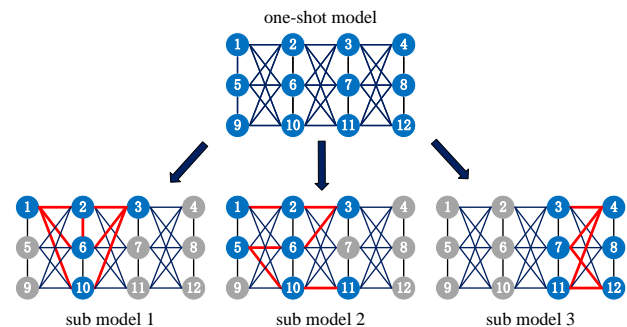

Fig. 4. An example of the search space based on the one-shot model. Blue nodes represent active nodes. Grey nodes represent inactive nodes. The red lines denote the active paths in the one-shot model.

Apart from assessing network candidates under low-fidelity estimations, one-shot NAS methods, also called weight sharing methodologies, have become more and more popular recently. One-shot NAS aims to construct a single large neural network (also denoted as the one-shot model) to emulate any network in the search space. As is shown in Fig.4, the one-shot model can be viewed as a DAG, all possible architectures are different sub-graphs of the one-shot model. The node in the DAG represents a layer (e.g. convolution layer) in the neural network, and directed edges stand for the flow of information (e.g. feature maps) from one node to another. Once the one-shot model training is complete, all network candidates in the search space can directly inherit weights from the one-shot model for evaluating performance on the validation dataset rather than training thousands of separate sub-models from scratch. The workflow of one-shot NAS follows four sequential steps:

- Design a one-shot model as the search space that contains all possible network architectures.
- Train the one-shot model to convergence by a network sampling controller.
- Adopt a search strategy (e.g. an evolutionary algorithm, reinforcement learning, and the gradient-based method)

to find the best sub-model based on the pre-trained one-shot model.

- Fully train the best submodel from scratch and evaluate its performance on the test dataset.

The source of "weight sharing" is first proposed from the ENAS [77], which used the RL-based method as the search strategy and forced all sub-models to share parameters from the one-shot model to avoid training each sub-model from scratch. Compared with NASNet [23], ENAS reduces the search cost from 1800 GPU-days down to 16 GPU-hours on CIFAR10 classicification task. Luo et al. [78] proposed the NAO algorithm, which replaces the RL-based search strategy of ENAS [77] with a GD-based auto-encoder that directly exploits weight sharing. Gabriel et al. [79] analyzed the role of weight sharing in ENAS. Zhang et al. [80] proposed SI-EvoNAS algorithm, an EA-based one-shot NAS framework, which jointly optimizes the network architecture and associated weights. In SI-EvoNAS, a sampling training strategy is proposed to train the parent individuals. A node inheritance strategy is proposed to generate the offspring individuals, which can force the offspring individuals to inherit the weights from their parents, thereby avoiding training the offspring individual from scratch.
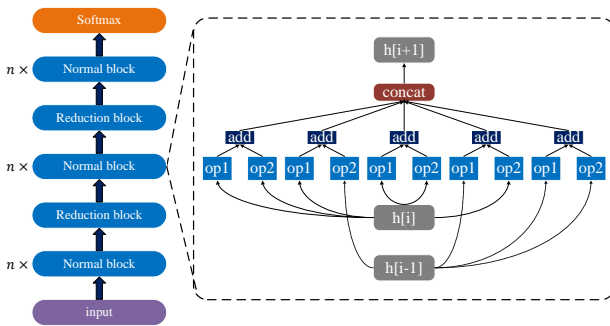

Fig. 5. An exemplar relaxation trick based on micro search space.

Yassine et al. [81] discovered the "multi-model forgetting" phenomenon in the one-shot NAS. Specifically, since the shared weights of the nodes are overwritten during the one-shot model training, the performance of the previously trained network will degrade when optimizing the network for subsequent training. As a result, the confidence of the ranking of candidate networks is no longer reliable. To solve this problem, Yassine et al. proposed a statistically-justified weight plasticity loss to regularize the one-shot model training. Bender et al. [79] proposed a "path dropout" algorithm that randomly removes some nodes of a one-shot model according to the dropout rate throughout one-shot model training. Guo et al. [54] proposed a uniform sampling strategy to ensure all nodes in the search space have equal optimization opportunities during the one-shot model training. Chu et al. [82] proposed the FairNAS algorithm, which unveils the root cause of the effectiveness of the one-shot model training under two fairness constraints including strict fairness and expectation fairness. Zhang et al. [19] formulated the training process of the one-shot model as a constrained continual learning optimization problem to ensure the current network training does not

degrade the performance of previous networks. To achieve this, Zhang et al. [19] proposed a search-based architecture selection (NSAS) loss function for one-shot model training. Ding et al. [83] designed broad scalable network architecture to avoid performance drops in ENAS during the phase of network performance estimation. Ning et al. [84] introduced an operation-level encoding scheme to change the parameter sharing scheme dynamically.

Another family of one-shot NAS algorithms adopts a continuous relaxation trick to transfer the discrete search space to be a continuous space by approximating the connectivity between different layers in the networks through real-valued variables. And then, the real-valued variables and the model weights are jointly optimized by the gradient descent (GD) procedure at the same time. Fig.5 shows an exemplar relaxation trick based on micro search space. The first work of relaxation-based NAS methods is generally viewed as the Differentiable NAS (DARTS) [73], which employed the GD method to search the best building blocks of a CNN architecture, and the experimental results on CIFAR10 and ImageNet have presented its performance. However, DARTS requires excessive GPU memory during architecture search because all layer operation candidates have to be explicitly instantiated in the memory [71]. Hence, Chen et al. [85] reduced the size of the one-shot model to solve this issue. Stochastic neural architecture search (SNAS) [86] adopted gradient information from generic differentiable loss to further accelerate the process of the network search. Cai et al. [20] proposed ProxylessNAS, which employed binary gates (1 or 0) to convert the architecture parameters of the one-shot model into binary representations. During the architecture search, ProxylessNAS only active a single path in the one-shot model by the binary gates. Therefore, the computational cost of ProxylessNAS is roughly the same as that of training a single network. Benefiting from the low computational cost, ProxylessNAS can directly search for an optimal network architecture on ImageNet. Zhou et al. [87] adopted a high-order Markov chain-based method to combine current indicators of an operation and its historical importance, which can provide more accurate decisions of the importance of the operation.

Wang et al. [88] observed that DARTS would suffer severe degradation because of the mechanisms in training and discretization. Hence, DARTS tends to keep more skip-connect operations in the final optimal model and converge to shallow network architecture. To address the issue aforementioned, Xu et al. [71, 89] proposed partially-connected DARTS (PC-DARTS) algorithm by using a channel sampling scheme that samples a subset of channels in each training step instead of activating all channels to reduce the redundant space of the one-shot model during the search process. iDARTS [88] adopted the node normalization strategy to maintain the norm balance of different nodes, thereby avoiding the degradation in DARTS. Zhou et al. [90] proposed a BDAS algorithm based on DARTS to search a domain matching diagnostic network. To solve the unfairness problem of one-shot model training in DARTS, the BDAS proposed a corresponding strategy based on path-dropout and warm-up and adopted variational Bayesian inference to estimate the uncertainty of

model matching.

Benefiting from the significant improvement in computational efficiency of one-shot NAS algorithms and significant progress in image classification tasks, the ==application== of one-shot NAS in the task of object detection has also attracted the attention of researchers. Early methods, such as NASNet [23] and RelativeNAS [91], transferred the searched network based on the classification tasks as the backbone for object detectors without further searches, which cannot guarantee ==optimal adaptation== for any detection task. With the development of NAS in automating the design of network architecture, it has also boosted the research into automatically designing the network architecture for object detectors rather than handcraft design. Inspired by one-shot NAS methods, OPANAS [92] proposed a novel search space of an FPN structure based on the one-shot model. Then, an EA-based one-shot NAS method is adopted to find the optimal path in the one-shot model to construct the FPN structure. Xiong et al. [93] proposed MobileDets for mobile or server accelerators. The authors wanted to ==replace depth-wise convolutions with regular convolutions in inverted bottlenecks== through the one-shot NAS method to improve both the accuracy and latency of models. Hit-Detector [94] adopted the gradient-based NAS method [73] to optimize the architecture of all components of the detector. The estimation quality of various one-shot and zero-shot methods was systematically investigated on ==five NAS benchmarks== [95].

C. Network morphism

Parameter remapping strategy is a class of popular methods to improve the computational efficiency in NAS. The parameter remapping strategy, also called network transformation/morphism, aims to remap the parameters of one model to each new model to speed up the training process of the new model. Chen et al. [96] tried to transfer the trained parameters from a small model to a new larger model with the help of the concept of function-preserving transformations, which effectively improves the performance of the large model on the ImageNet classification task. Following this manner, Cai et al. [97] adopted the Net2Deeper and Net2Wider operators from [96] during the architecture search. EAS [98] adopted network morphisms to grow the width of the layer and the depth of the network. Cai et al. [66] proposed Once-for-All method, which adopts a progressive shrinking algorithm to train the one-shot model. After the one-shot model has been trained, Once-for-All maps the parameters of the one-shot model to sub-models directly.

IV. SURROGATE-ASSISTED NAS

While proxy methods try to make an estimation of the network performance according to a set of sub-optimal weight matrices, the surrogate-assisted NAS approaches take advantage of those trained networks by modeling a surrogate. Generally, the surrogate models are trained by a set of training data, which consists of network encoding and the associated performance in pairs. The ==trained surrogate can evaluate the performance== of any candidate architecture in the search space,

averting the huge computational overhead of training networks with poor performance. This section begins with a brief introduction to surrogate-assisted optimization before digging into ==different types of surrogate-assisted NAS techniques==, such as Bayesian optimization-based NAS and surrogate-assisted evolutionary algorithm-based NAS. Then we give a detailed discussion about federated NAS and multi-objective optimization in surrogate-assisted NAS methods.

A. Surrogate-assisted Optimization

Surrogate-assisted evolutionary optimization derives from the practical challenges that some engineering optimization problems have no analytic objective functions, or the evaluation of a candidate solution can take hours or even a few days [99, 100]. For example, in aerodynamic design optimization, the numerical simulations could be very time-consuming [101]. When population-based optimization methods, such as EAs, are adopted to search for the optimal solution for a given task, a huge number of candidate solutions need to be evaluated. Consequently, the calculation cost under these circumstances is unaffordable. In order to cope with this, surrogate-assisted approaches investigate training surrogate models based on a ==limited== amount of data, and provide reliable evaluations for candidate solutions within optimization. The limited training data often comes from physical experiments, numerical simulations, or historical information [102–104].

The data for training surrogate models has two categories in general: direct data and indirect data [105, 106]. Direct data consists of at least two parts: decision variables and the corresponding objective or constraint values, which can be directly adopted to train a surrogate model. Most of the surrogate-assisted NAS approaches belong to this category, since the encoding of network architectures can be regarded as decision variables, and the network performance is taken as objective values. However, in other cases, it may be impossible to collect data in the form of decision variables and objective function values. For example, in trauma system design optimization, there is no mathematical formulation of the objective function, and the emergency accident records are the only information accessible [104]. Under these circumstances, the designer should calculate the objective values from the indirect data before building and training a surrogate model.

Depending on whether new sample points can be collected by making evaluations of candidate solutions with a real objective function during the optimization process, surrogate-assisted evolutionary optimization can be categorized into offline and online algorithms. In offline surrogate-assisted evolutionary optimization, a set of candidate solutions together with their ground-truth objective values are collected to train a surrogate model before the optimization starts. During the optimization process, no new data can be actively generated for real evaluation, and only the predictions provided by the trained surrogate model can guide the evolutionary search. In other words, the final performance of the algorithm is mostly determined by the distribution of samples in the offline training set as well as the accuracy of the surrogate model. In practice, it is non-trivial to collect a set of ideal offline data

in terms of quality, quantity and distribution. The data can even be noisy or incomplete in some circumstances. These challenges may hinder the offline algorithms from achieving better performance. To tackle this problem, some techniques can be adopted to improve the model performance of offline surrogates [106]. From the perspective of data, we can use data preprocessing and data mining techniques to alleviate the influence of noise and uneven distribution on data quality. From the perspective of surrogate models, we can either choose an appropriate model elaborately for a specific task, or build an ensemble of surrogate models to improve its robustness. From the perspective of tasks, multi-fidelity fitness evaluation strategies and knowledge transformation between similar tasks can also be adopted to reduce the computational overhead.

By contrast, in online surrogate-assisted evolutionary optimization, there are no more restrictions on training surrogate models with limited offline data. During the evolutionary search, new data points can be actively sampled and evaluated by the objective function to get ground-truth labels. Consequently, we can enrich the quantity and quality of the training set of the surrogate model, thus improving its prediction accuracy. In addition to the techniques adopted in offline algorithms, we also need model management concerned with the sampling number, frequency and selection criteria to strike a balance between exploration and exploitation.

Model management is a key factor in online surrogate-assisted optimization, since it enables more efficient evaluations of the objective function as well as a more accurate surrogate approximation [105]. In surrogate-assisted evolutionary optimization, fitness evaluations are provided by surrogate models instead of the real objective functions, contributing to a significant reduction in computational overhead. However, it is often infeasible to rely solely on surrogates for the fitness approximation. With a high-dimensional objective function and limited training data, a surrogate model without much a priori knowledge of the problem itself could have a bad performance. Model management, based on individuals or generations, investigates how to use the original fitness function efficiently in a surrogate-assisted optimization process. For individual-based strategies, the best individual or randomly selected individuals in each generation will be evaluated by the original function, while others will be evaluated by the surrogate model. For generation-based strategies, the real evaluation of all individuals in the population will be carried out every few generations.

B. Bayesian Optimization in NAS

Bayesian optimization (BO) has emerged as a powerful tool for solving expensive black-box optimization problems, which has been widely applied to hyperparameter tuning of various machine learning scenarios such as recommendation systems, natural language processing, robotics and reinforcement learning [107, 108].

As a sequential strategy for derivative-free global optimization, the general principle of BO is to build a probabilistic model for the objective of interest, which can be updated using the collected data. During the optimization loop, the model provides an informative posterior distribution to guide the optimization directions and strike a balance between exploration and exploitation.

The framework of BO has two key components: a probabilistic surrogate model and an associated acquisition function. The surrogate model contains our assumptions about the priori information of the unknown objective function. Gaussian Processes (GPs) are the most common choice for surrogate models. It is assumed that the distribution of the unknown objective function follows a Gaussian process, which means the function should be smooth and the deviations are Gaussian noises. The acquisition function is introduced to determine which is the next sample point to be evaluated. Since BO aims at finding the global optimum with fewer function evaluations, the optimization of the acquisition function is expected to compromise between exploration and exploitation. Specifically, those samples with a larger degree of uncertainty (exploration) or a higher predicted value (exploitation) are preferred to be evaluated in sequence for a maximization problem. Different acquisition functions have been designed, such as probability of improvement, expected improvement [109], the Gaussian process upper confidence bound [110] and entropy search [111]. It is worth noting that the acquisition function should be computationally much cheaper (but not necessarily easier) to optimize than the objective function, since the original black-box function is time-consuming and computationally intensive.

When searching for optimal network architectures from the Bayesian optimization perspective, we consider NAS as an expensive black-box optimization problem. Before the search process, a search space $\mathcal{A}$ with related to a specific task/dataset is defined in advance, containing all possible solutions $\alpha \in \mathcal{A}$ for the optimal architecture. Actually, it is non-trivial to fully explore the entire space $\mathcal{A}$ due to the insupportable overhead. An objective function $f(\alpha)$ indicates the performance metric of neural networks, e.g., the validation accuracy of a given architecture. It may take several hours to fully train a network for evaluating $f(\alpha)$. A lot of work has been dedicated to Bayesian optimization-based NAS approaches recently.

Auto-Keras [112] is an open-source AutoML system. It's developed from a NAS framework based on Bayesian Optimization, which is designed to operate on a network morphism. In this approach, a Gaussian Process model is built and trained with the existing architectures and their performance. To cope with the challenge that the original network architectures are not in Euclidean space, a neural network kernel is proposed based on the edit-distance for morphing one architecture to another. The optimization of the acquisition function is also re-designed for the tree-structured search space in network morphism. NASBOT [113] is a Gaussian process based BO framework for neural architecture search. A distance metric named OTMANN (Optimal Transport Metrics for Architectures of Neural Networks) is developed to quantify the similarity between two networks in the search space, and the acquisition function is optimized by EA approaches. A main challenge for BO in a graph-like search space is how to capture the topological structures of neural networks. NAS-

BOWL [114] combines Weisfeiler-Lehman graph kernel with a Gaussian process, enabling the surrogate model to be directly defined in a graph-based search space. However, the above NAS methods are full-fledged, and we cannot tell which component makes the most contribution. White et al. [115] made a systematic analysis of the "BO + performance predictor" framework with five separate components: the network encoding, the performance predictor, the uncertainty calibration, the acquisition function and the acquisition function optimization. Then a BO-based algorithm named BANANAS was proposed based on the analysis. Each network architecture in the search space is represented as a labeled DAG, and path encoding is developed to improve the predictor performance.

While the BO-based NAS algorithms in [112–114] focus on designing neural network kernels for Gaussian Process, other works introduced different kind of surrogate models. BONAS [116] combines Graph Convolutional Network (GCN) [117] and Bayesian sigmoid regressor as the surrogate model instead of GP. The individuals with the top-k UCB scores will be selected and evaluated by weight-sharing in order to update the surrogate model. [118] developed a graph Bayesian optimization framework with a Bayesian graph neural network as surrogate model. As the first Bayesian approach for one-shot NAS, BayesNAS [119] uses a hierarchical automated relevance determination prior to model architecture parameters, alleviating the inappropriate operation over $zero$ operations in most one-shot methods. One of the challenges for Bayesian Optimization in NAS is the high-dimensional and discrete decision space. To counter this problem, Neural Architecture Generator Operation (NAGO) [120] considered NAS as a search for the best network generator, and built a novel graph-based hierarchical search space which can cover a wide range of network architectures with only a few hyperparameters. Consequently, the problem dimensionality was greatly reduced, allowing Bayesian Optimisation to be used effectively. GP-NAS [121] investigated the correlation among different architectures as well as the performance from a Bayesian perspective, where a kernel function for NAS is specially designed by categorizing operations into multiple groups.

In addition to direct search of network architectures, BO can also be integrated with other proxy approaches to improve search efficiency, such as knowledge distillation [122] and surrogate models [123, 124].

C. Surrogate-assisted Evolutionary NAS

As a class of population-based optimization algorithm, EAs maintain a population consisting of feasible solutions, and generate offspring with progressively better performance, enabling the algorithm to converge towards the optimal solution. In EA-based NAS approaches, individuals in the population are considered as candidate architectures defined in the search space, and the genotypes are determined by the network encoding [125]. The fitness function reflects the network performance, which can be the validation accuracy or other evaluation factors. Due to the superior performance of EAs in solving black-box optimization problems, much work has focused on designing EA-based NAS approaches.

In 2017, Google proposed the LargeEvo algorithm [24], which uses a genetic algorithm to search for well-performed CNN architectures on CIFAR-10 and CIFAR-100 datasets. This is commonly thought to be the first evolutionary-based NAS algorithm. Since then, a lot of work has been dedicated to searching for optimal network architectures by EA approaches [38, 44, 52, 55, 126–128].

However, it is non-trivial to directly use evolutionary algorithms to search for the optimal network architectures for a given task. First of all, the fitness evaluation of one candidate solution takes a considerable amount of training time. In order to get the performance evaluation of a candidate architecture, one should initialize the weights of the given network, and then train it on the training dataset by using the gradient descent approach over a large number of epochs before convergence. As the training set and network size grow, this procedure might take hours or even days. On the other hand, the EA approach always needs to evaluate a large number of candidate solutions at a time due to its population-based properties. As the task difficulty and the population size increase, this time-consuming process will be a challenge to limited computing resources [129].
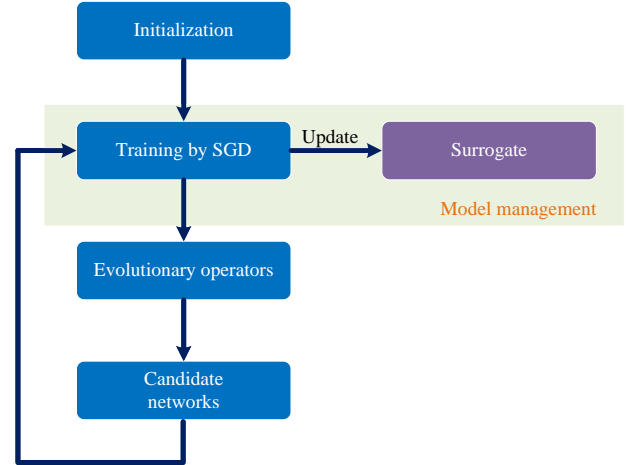
Fig. 6. A basic framework of surrogate-assisted evolutionary NAS algorithms

To address this problem, a performance predictor (also known as a surrogate model) is introduced [99]. The surrogate model aims to estimate network performance without training the network from scratch. The basic steps of surrogate-assisted evolutionary NAS approaches are:

- Sample a set of network architectures from the search space, and train them from scratch to get the ground truth labels. Store the samples in an archive A.
- Use the archive A to construct a training dataset D_{tr}. Build and train a surrogate model M by using D_{tr}.
- Perform neural architecture search by EAs, with the network performance predicted by the surrogate model M.
- Select candidate architectures for real evaluation according to the model management strategy. Update the archive A and the training dataset D_{tr}.

- Train and evaluated the selected architectures. Update the archive A and the training dataset D_{tr}.
- Update the surrogate model M for the next iteration.

According to [130] and [131], there are mainly four categories of network performance predictors: learning curve-based predictors [132–135], weight sharing-based predictors [77], shallow training-based predictors [44] and end-to-end performance predictors [130, 136, 137]. The learning curve-based approaches require a partial training of the neural network, and then extrapolate the upcoming trends of the curve based on the observed part. An example for learning curve-based models is presented in [135], where LSTM is adopted as a seq2seq model to predict the network performance based on the learning curve of the first few epochs. Domhan et al. [132] built a set of parametric functions as a probabilistic model, and used it to extrapolate from the initial part of the learning curve to a future point. A run will be terminated if its performance prediction on validation set seems unlikely to achieve the best model performance so far. Based on this work, Klein et al. [133] developed a Bayesian neural network as the probabilistic model, and improved its prediction by a specially-designed learning curve layer. Baker et al. [134] trained sequential regression models instead of a single Bayesian model to estimate the validation accuracy of candidate networks. However, the learning curve-based methods still rely on the training process of the candidate networks, making the prediction time-consuming. Similarly, the shallow training-based and weight sharing-based predictors (as mentioned in Section III-B) may also suffer from poor generalization ability due to insufficient training and weight dependency assumption.

Another option is to build and train an end-to-end predictor, which directly takes a network architecture as input, and predicts the network performance as its output. Deng et al. developed an end-to-end training approach called *Peephole* [136]. It uses LSTM to encode the individual layers as well as the number of epochs into a structural embedding, and feds it into a multi-layer perceptron to predict the validation accuracy after the given epoch. In contrast to the learning curve-based methods, *Peephole* can predict the entire learning curve without knowing the initial part of it. Although this end-to-end training can make the prediction more efficient, training such a surrogate model requires a large number of pre-trained neural networks, which introduces additional computing overhead.

To further reduce the computational overhead, E2EPP [130] is one of the representative end-to-end performance predictor models in surrogate-assisted NAS, which can achieve good performance with limited training data. In E2EPP, a random forest [138] is adopted as a surrogate model for predicting the network performance by given the network encoding. The proposed performance predictor is then embedded with AE-CNN [139], an evolutionary deep learning method to further verify its effectiveness. The advantage of adopting a random forest as a proxy model instead of other models like neural networks is that it can directly accept discrete data as input, without the need for a large amount of training data. As an ensemble learning method, a random forest model consists of a set of decision trees. Each decision tree will select a subset of features from the decision variables randomly. During the training process, each decision tree learns a mapping from the features to the targets. During the prediction process, the outputs of selected decision trees are averaged as the final prediction of the random forest. NPENAS [123] is another neural predictor guided approach for evolutionary neural architecture search. In NPENAS, multiple offspring are generated from each parent instead of one, in order to enhance the exploration ability of the search algorithm. To alleviate the additional computational cost caused by extra candidate architectures, a neural predictor is introduced to rank the offspring from the same parent, and only the candidate with the highest predicted performance will be selected for real evaluation. Two different kinds of neural predictors are proposed in NPENAS. The first one is an acquisition function defined by a graph-based uncertainty estimation network. Inspired by BO-based NAS algorithms, the authors assume that the distribution of network architectures in the search space is independent and identical, thus the performance prediction function can be regarded as a Gaussian Process, which is defined by its mean and standard deviation functions. A graph-based uncertainty estimation network is trained as a surrogate model to provide the mean and standard deviation values for a given architecture. Specifically, the model uses GINs [140] and MLPs for architecture embedding, followed by fully-connected layers to predict the mean and standard deviation values. The second neural predictor has a similar structure, while it predicts the architecture performance directly. The algorithms embedded with these two neural predictor are named NPENAS-BO and NPENAS-NP respectively, and they both showed promising performance on NASBench datasets [141, 142].

Wen et al. [143] proposed a simple yet effective method with only three steps for surrogate-assisted NAS. It first trains a set of randomly sampled network architectures to get the validation accuracies. Then the set of architectures are used to train a regression model (which is GCN in this paper). Finally, the trained model gives prediction on a large number of random architectures, and only the top-K candidates will be trained from scratch to find the optimal architecture. A recent work proposed by Greenwood et al. [137] augments the original DeepNEAT [39] by introducing a surrogate model and two-phase active learning paradigm. During the initialization phase, networks are trained and evaluated by the standard DeepNEAT. During the active learning phase, the surrogate model is used to make predictions for the population rather than training and evaluating them directly. PRE-NAS [144] proposes a representative selection scheme which enables it to train a well-performed performance predictor within an extremely limited number of training samples. B2EA [124] introduces two BO as surrogates for an EA-based NAS approach. The first BO controls the search space attentively, and the second BO predicts performance for network architectures without training process. MORAS-SH [145] introduces an online surrogate model to predict the high-fidelity performance of architectures as a helper-objective for adversarial robustness search. DAU-NAS [146] adopts a random forest as a surrogate model to directly predict the performance of each subnet in addition to the weight sharing strategy.

Graph Neural Network (GNN) is another type of effective

surrogate model in evolutionary NAS [147]. GNN takes a graph as its input and updates the graph attributes using message-passing techniques such as graph convolution layers. The output of a GNN is a graph with the same connectivity, while the graph attributes are updated and each node embedding has already incorporated the information from its neighborhood. Because the structures of most neural networks can be naturally encoded as DAGs [143], an increasing amount of work is dedicated to using GNNs as surrogate models in NAS. Lukasik et al. [148] used a GNN as a graph encoder to map networks from a discrete graph space to a continuous vector space. The model is first trained on CIFAR-10 dataset in a supervised learning method, and then its prediction ability is evaluated on zero shot scenarios with unseen architectures. Different from the traditional fully supervised way of training surrogates, Tang et al. [149] proposed a semi-supervised performance predictor based on GNN. Firstly, both labeled and unlabeled data are fed into an auto-encoder to get the meaningful embedding. Then a relation graph is constructed based on the embedding, revealing intrinsic connections among similar architectures. Finally, taking both the embedding and the relation graph as the inputs, a GCN is trained to predict the network performance of these unlabeled samples. Ning et al. [150] proposed a general graph-based architecture encoder called GATES. It regards neural networks as data processing graphs, and different operations are modeled as information transformations. By encoding cell architectures into embedding vectors, GATES can improve the performance predictors for surrogate-assisted NAS methods in various search spaces. Kyriakides et al. [151] proposed an evolutionary-based method to search for GCNs as performance predictors to evaluate the relative ranking of various network architectures.

While most of the surrogate-assisted NAS approaches share similar protocols of training a surrogate model by mean square error criterion (MSE), Sun et al. [131] proposed a pairwise ranking indicator (PRI) for surrogate training in NAS. In this protocol, a PRI is adopted to train the surrogate model instead of the traditional MSE function. Concretely, given any two sample architectures, the ranking information between the two samples will be used to train a regression model. The trained model can be easily integrated into evolutionary NAS algorithms, since its predictions reflect the real ranking among candidate architectures and are consistent with EAs' selection criterion. Another similar work is ReNAS [152], where the authors defined a pairwise ranking-based loss function for training the surrogate model instead of the traditional element-wise loss functions such as MSE. The implicit idea is that predicting the relative ranking between two architectures is more important in evolutionary search than predicting the true performance values. Similarly, HW-PR-NAS [153] introduces a novel loss function to rank the architectures based on their dominant relations, which avoids using multiple surrogates to estimate different objectives. Arch-Graph [154] formulates NAS as an architecture relation graph prediction problem, and trains a pairwise relation predictor to give architecture relation on any given task embeddings.

In addition to a single type of predictor model, White et al. [155] demonstrated that the surrogate performance can be significantly improved by combining different categories of neural predictors.

D. Federated NAS

Federated Learning (FL) [156] is an emerging technique in machine learning, where a global model is trained using distributed datasets stored locally on various clients, without the requirement to transform the privacy data to a centralized server or a third party [157, 158]. With the surge of interest in privacy preserving, there is an increasing demand for search for specific network architectures under the federated learning framework [59, 60, 159–163]. The challenges of federated NAS compared to centralized NAS may come from huge communication cost, unbalanced data distribution as well as real-time deployment.

In terms of whether the architecture search and deployment processes are coupled, federated NAS can be categorized into offline and online algorithms [159]. There are two separate parts in offline federated NAS: the first stage is searching for an optimal network architecture, and the second stage is training and deploying this searched network to all clients. One example for offline federated NAS is [59], where the NSGA-II [57] is adopted to search for a set of Pareto optimal solutions for a given task. Offline NAS indicates that only the final optimized architecture will be employed, and the performance of the candidate solutions generated during the search process do not matter much. However, in some practical scenarios of federated learning, the optimization and deployment of neural networks should be performed simultaneously. In other words, online federated NAS requires that not only the final optimized model should work well, but also the architectures generated during the search process should have acceptable performances. Zhu et al. [60] proposed a real-time federated evolutionary NAS algorithm, where a double-sampling technique is adopted to reduce the huge computational and communication cost in online federated NAS. Specifically, only a randomly selected subset of clients participate in the training of a candidate network, and only sub-models of the supernet are sampled and trained and in the search process. Liu et al. [164] developed a multi-objective convolutional interval type-2 fuzzy model for federated NAS to ensure medical data security.
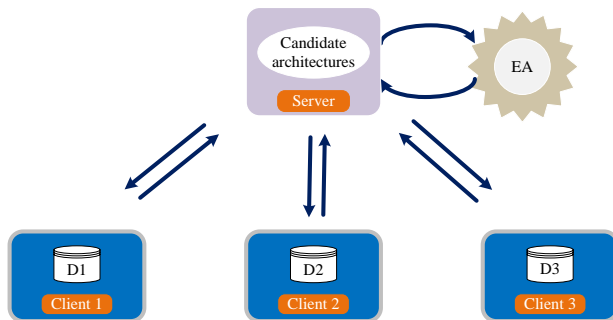

Fig. 7. A basic framework for federated evolutionary NAS algorithms

Another consideration for federated NAS is non-IID (non-independently and identically distributed) data on different

clients [158]. Due to the weight divergence of local models [165], non-IID data can lead to significant performance degradation when training a neural network under federated learning settings. To counter this problem, He et al. [160] proposed FedNAS, a distributed NAS algorithm to search for optimal architectures in federated learning with non-IID data distribution.

In federated NAS approaches, multiple clients collaboratively search for a well-performed network model without uploading their private data to a server. By keeping the data local, the security of client privacy is guaranteed to a certain extent. However, the gradient information of model parameters, which is exchanged during the training process, can still reveal privacy implicitly [166]. In other words, avoiding data disclosure in federated NAS may be not enough in terms of privacy protection. To fill this gap, Singh et al. [162] proposed DP-FNAS, which uses differential privacy [167] to further improve security by adding Gaussian noise to the gradient values before being sent to the server for aggregation.

Most of the existing federated NAS algorithms focus on searching network architectures in horizontal federated learning scenarios, where each participant has access to the same features and labels with high quality. This kind of scenarios are defined as horizontal federated learning (HFL). Contrary to this, vertical federated learning (VFL) means all the participants share the same sample ID space, but with different feature subsets. In most of the VFL cases, only one participant holds the label set. In order to collaboratively search for an optimal neural network with feature-partitioned local datasets, Liang et al. [161] proposed a self-supervised federated neural architecture search (SS-VFNAS) algorithm under cross-silo settings. First, each client uses a self-supervised NAS approach to find a local optimal network with its own data, and then all clients perform a supervised NAS to enhance the local optimal model in the VFL framework.

Federated neural architecture search is an emerging topic in automated machine (AutoML) learning research, and there are still some open issues to be investigated, such as introducing surrogate models to improve search efficiency, and designing robust NAS approaches to defend adversarial attacks.

E. Multi-objective NAS

In most practical scenarios of NAS, the model performance (validation accuracy) is not the only thing that matters. Several hardware constraints also need to be considered when deploying a deep learning model, such as computing power, memory usage, inference time and communication cost. The generic aim of multi-objective neural architecture search is to find a well-performed network architecture with minimal model size, computational complexity or inference time.

However, it's non-trivial to find such an acceptable network architecture, since the multiple objectives to be optimized simultaneously are always in conflict with each other. One method is to convert the multi-objective optimization into a single objective by introducing weight hyperparameters. Another idea is to search for a set of feasible solutions, where various architectures maintain a trade-off among conflicting

objectives. Consequently, population-based evolutionary algorithms seem to be a natural choice for multi-objective NAS, and surrogate models are also adopted to improve the search efficiency [52, 168–171]. Dpp-net [169] trains a surrogate function to predict the classification accuracy for different architectures in a population, and only the Pareto-based top-k candidates are trained and evaluated. Instead of training a united surrogate model for all architectures, NSGANetV1 [52] down-scales the candidate networks and trains these proxy models to get the validation accuracy and FLOPs for non-dominated ranking. Following this work, NSGANetV2 [170] treats NAS as a bi-level optimization task and uses surrogates at both upper and lower levels. For the upper level (architecture) optimization, four different kinds of surrogate models are trained at each iteration, and then Adaptive Switching is used to choose the best model for performance prediction. For the lower level (weights) optimization, a supernet is trained once before the search process, and candidate architectures inherit weights from the supernet as a warm-start for the real evaluation.

To conveniently investigate the comparisons, Table I compares the performance of various computationally efficient NAS methods in terms of the classification accuracy and consumed GPU days on three popular datasets for image classification, namely CIFAR-10, CIFAR-100 and ImageNet.

V. CHALLENGES AND FUTURE DIRECTIONS

Although a lot of effective NAS approaches have been proposed and achieved compelling performance, there are still many open challenges and future directions for efficient neural architecture search approaches.

A. Sampling efficiency

The original purpose of introducing a surrogate model as a performance predictor is to reduce the computational cost for evaluating a network (mostly training from scratch) during the searching process. However, in order to build such a surrogate with good predictive performance, it is usually necessary to construct a training set first. Building a training set itself also requires sampling and training a large number of network models by gradient-based methods. In surrogate-assisted NAS, a desired surrogate model should be not only well-performed but also sampling efficient (i.e., the number of network architectures to be trained for constructing the surrogate is as small as possible). Furthermore, the computational overhead associated with training surrogate models cannot be overlooked. For example, the computational complexity of training a Gaussian Process model is $O(N^3)$, where N denotes the number of training samples. When the performance of a single surrogate is undesirable, an ensemble of different types of surrogate models needs to be build, which further increases the training cost.

B. Model management

In offline surrogate-assisted NAS algorithms, the surrogate model is only trained once before the optimization process,

TABLE I
CLASSIFICATION ACCURACY OF VARIOUS COMPUTATIONALLY EFFICIENT NAS APPROACHES ON CIFAR-10, CIFAR-100 AND IMAGENET

Algorithm	Accuracy (%)			GPU-days	Search method	Proxy method / Surrogate
	CIFAR-10	CIFAR-100	ImageNet			
NAS v3 [27]	95.53	–	–	22400	RL	–
Genetic-CNN [38]	92.9	–	72.13	17	EA	–
AE-CNN [45]	95.3	77.6	–	27	EA	–
CNN-GA [46]	96.78	79.47	–	35	EA	–
Hier-EA [172]	96.37	–	–	300	EA	–
large-scale Evo [24]	94.60	77.00	–	2750	EA	–
LEMONADE [55]	97.42	–	–	90	EA	–
CGP-CNN [41]	97.25	–	–	227	EA	–
Amoebanet-A [126]	96.66	81.07	–	3150	EA	Low-fidelity estimation
MFENAS [173]	97.61	–	73.94	0.6	EA	Low-fidelity estimation
Block-QNN-S [29]	96.70	82.95	–	90	RL	Low-fidelity estimation
MetaQNN (top model) [22]	93.08	72.86	–	90	RL	Low-fidelity estimation
PNAS [174]	96.37	80.47	74.2	225	SMBO	Low-fidelity estimation
NASNet [23]	97.35	82.19	74.0	1800	RL	Low-fidelity estimation
ME-HDSS [47]	93.65	72.89	–	–	EA	Low-fidelity estimation
EoiNAS [175]	97.5	–	–	0.6	GD	Low-fidelity estimation/ One-shot
ModuleNet[176]	97.33	82.01	78.69	–	EA	Low-fidelity estimation
ENAS [77]	97.06	–	–	0.5	RL	One-shot
SI-EvoNAS [80]	97.31	84.30	75.8	0.458	EA	One-shot
Evo-OSNAS [64]	97.44	84.16	77.48	0.5	EA	One-shot
DARTS [73]	97.18	82.46	73.3	1	GD	One-shot
SNAS [86]	97.15	79.91	72.7	1.5	GD	One-shot
Proxyless NAS [20]	97.92	–	75.1	–	GD	One-shot
WPL [81]	96.19	–	–	–	RL	One-shot
BNAS [177]	97.03	–	74.3	0.19	RL	One-shot
PDARTS [85]	97.50	84.08	75.6	0.3	GD	One-shot
PC-DARTS [71]	97.43	82.89	74.9	0.3	GD	One-shot
BayesNAS [90]	96.98	–	73.5	0.2	GD	One-shot
GDAS [178]	97.25	81.98	74.1	0.4	GD	One-shot
RandomNAS-NSAS [179]	97.41	82.44	74.5	0.7	GD	One-shot
NAO+WS [180]	96.47	–	74.3	0.3	GD	One-shot
iDARTS [88]	97.65	–	75.3	1.9	GD	One-shot
NM(ensemble across runs) [181]	95.6	80.4	–	4	RL	Network morphism
Net2Net [96]	–	–	78.5	18	RL	Network morphism
EAS [98]	95.77	–	–	10	RL	Network morphism
Once for all [66]	–	–	80	1.7	EA	Network morphism
AK-DP [112]	96.4	–	–	–	BO	GP
NAS-BOWL [114]	97.39	–	–	3	BO	GP
GP-NAS [121]	96.21	–	73.4	0.9	BO	GP
BANANAS [115]	97.36	–	–	–	BO	Neural predictor
BayesNAS [119]	97.19	–	73.5	0.2	BO	One-shot
NAGO [120]	96.6	79.3	76.8	–	BO	BNN
BONAS [116]	97.57	–	75.2	10	BO	GCN
SSA-NAS [149]	94.01	78.64	–	–	EA	GCN
E2EPP [130]	94.70	77.98	–	8.5	EA	Random forest
PRE-NAS [144]	97.51	–	76.0	0.6	EA	Random forest
NPENAS [123]	97.46	–	–	1.8	EA	GIN+MLP
NSGANetV1 [52]	97.98	85.62	76.2	27	EA	Down-scale
NSGANetV2 [170]	98.4	–	80.4	–	EA	Ensemble
GATES [150]	97.42	–	–	–	EA	GATES+MLP
ReNAS [152]	93.99	78.56	–	–	EA	LeNet-5

while in online approaches, newly sampled networks are generated and added to the training set, and the surrogate is updated in an online manner. Considering the large size of a search space and the computational overhead for training a surrogate, it is impossible for a surrogate model to cover the entire search space. Under this circumstance, a properly designed model management strategy can improve the predictive accuracy of a surrogate. For example, in multi-objective neural architecture search, we are concerned more about the model performance near the Pareto front than those dominated regions.

Therefore, the selection of surrogate models, the construction of the initial training dataset, and the model management strategies are key factors in surrogate-assisted NAS that deserve more attention.

C. Federated Learning

Despite the increasing application of federated learning and AutoML, only a handful of work has attempted to design effective network architectures under FL scenarios. The main challenge for training a neural network model in FL is the distributed (and even non-IID) data allocation due to privacy concerns. This becomes more notable for federated NAS since a huge amount of candidate architectures need to be trained and evaluated during the optimization process. Here are some future directions suggested for federated NAS:

- In the cases where data security is the primary concern for federated neural architecture search, privacy-preserving approaches such as differential privacy [167], secure aggregation [182] and homomorphic encryption [183] could be aggregated into current NAS algorithms to further enhance the security protection.
- A significant difference between federated NAS and centralized NAS is various hardware constraints among different clients. In a centralized environment, we simply need to consider hardware limitations of the target device, and convert it into a multi-objective optimization problem. However, in federated NAS, it is more common for different participants to have various hardware constraints or even different data distribution, so it may be not reasonable to deploy a uniform global model to all clients.
- The communication cost in federated learning is a key obstacle that hinders the algorithm from better scalability. A single network trained by FedAvg [156] requires many rounds of communication to transmit the updated information between clients and server. As one can imagine, the communication overhead of an evolutionary NAS approach will increase proportionally with the population size. Therefore, it is worth investigating how to reduce the communication and computational costs of population-based NAS in a federated learning environment.
- Finally, the effectiveness of surrogate models in federated NAS remains to be explored. For example, one method is to train a global surrogate by aggregating local information from the clients, another method is to maintain and update a local surrogate on each client separately.

D. Green AI

Generally, there is a considerable computational overhead when searching for an optimal network architecture for a given task. Although various proxy methods have been developed, it should be noted that the training process of the surrogate itself will also consume computing resources. Consequently, hardware requirements and expensive computational cost have been the bottlenecks in the real-world application of neural architecture search. It is worth investigating how to adapt existing surrogate-assisted NAS algorithms and deploy them on resource-limited edge devices, such as mobile phones, IoT devices, and embedded systems [184]. In fact, the majority of current NAS approaches rely on direct encoding strategies, which limits the diversity of the candidate network architectures. One promising direction is to develop indirect or generative encoding strategies with scalability, in order to enhance the flexibility for deployment on resource-constrained edge platforms.

VI. CONCLUSION

This survey conducts a systematic overview and detailed analysis of computationally efficient methods for performance prediction in neural architecture search algorithms. We first give a brief overview of existing NAS methods, mainly based on reinforcement learning and evolutionary algorithms. We categorize the computationally efficient NAS approaches into proxy-based methods and surrogate-assisted methods according to whether the weight values are needed for the prediction. The proxy-based methods evaluate the network performance using proxy metrics, including low-fidelity estimation, one-shot NAS and network morphism. We present a summary of representative literature on each type of strategy with an analysis of their characteristics. In contrast to other NAS surveys, we further concentrate on surrogate-assisted NAS methods, where surrogate models are trained and updated to evaluate the performance of unseen architectures in an end-to-end manner. Then, a detailed description and performance analysis of different types of surrogate models with corresponding milestone work. Finally, we discuss the existing challenges and future directions for performance prediction in NAS optimization, especially under the privacy-preserving federated learning framework. We hope this survey will provide some insights into more efficient performance prediction in NAS algorithms, thereby promoting research in AutoML and security machine learning.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[6] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[10] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1395–1403.

[11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[14] A. Toshev and C. Szegedy, "Human pose estimation via deep neural networks'," *CVPR.(Columbus, Ohio, 2014)*, pp. 1653–1660, 2014.

[15] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Master's thesis, University of Tront*, 2009.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[17] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[18] X. Dong and Y. Yang, "Searching for a robust neural architecture in four gpu hours," in *Proceedings of the IEEE Conference on computer vision and pattern recognition*, 2019, pp. 1761–1770.

[19] M. Zhang, H. Li, S. Pan, X. Chang, C. Zhou, Z. Ge, and S. Su, "One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2921–2935, 2021.

[20] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2019.

[21] 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, 2021.

[22] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[23] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[24] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2902–2911.

[25] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Aging evolution for image classifier architecture search," in *AAAI conference on artificial intelligence*, vol. 2, 2019.

[26] Y. Xu, L. Xie, W. Dai, X. Zhang, X. Chen, G.-J. Qi, H. Xiong, and Q. Tian, "Partially-connected neural architecture search for reduced computational redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2953–2970, 2021.

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

[28] C. J. C. H. Watkins, "Learning from delayed rewards," *King's College, Cambridge United Kingdom*, 1989.

[29] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2423–2432.

[30] Z. Zhong, Z. Yang, B. Deng, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Blockqnn: Efficient block-wise neural network architecture generation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 7, pp. 2314–2328, 2020.

[31] L. M. Schmitt, "Theory of genetic algorithms," *Theoretical Computer Science*, vol. 259, no. 1-2, pp. 1–61, 2001.

[32] B. L. Miller, D. E. Goldberg *et al.*, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[33] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*. IEEE, 1992, pp. 1–37.

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

[35] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[36] B. Inden, Y. Jin, R. Haschke, and H. Ritter, "Evolving neural fields for problems with large input and output spaces," *Neural Networks*, vol. 28, pp. 24–39, 2012.

[37] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.

[38] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1379–1388.

[39] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, "Evolving deep neural networks," in *Artificial intelligence in the age of neural networks and brain computing*. Elsevier, 2019, pp.

293–312.

[40] J. Liang, E. Meyerson, and R. Miikkulainen, "Evolutionary architecture search for deep multitask networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 466–473.

[41] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the genetic and evolutionary computation conference*, 2017, pp. 497–504.

[42] S. Saxena and J. Verbeek, "Convolutional neural fabrics," *Advances in neural information processing systems*, vol. 29, 2016.

[43] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[44] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 394–407, 2019.

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

[46] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated cnn architecture design based on blocks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1242–1254, 2020.

[47] D. O'Neill, B. Xue, and M. Zhang, "Evolutionary neural architecture search for high-dimensional skip-connection structures on densenet style networks," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 6, pp. 1118–1132, 2021.

[48] T. Zhang, C. Lei, Z. Zhang, X.-B. Meng, and C. P. Chen, "As-nas: Adaptive scalable neural architecture search with reinforced evolutionary algorithm for deep learning," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 830–841, 2021.

[49] J. Liu, S. Zhou, Y. Wu, K. Chen, W. Ouyang, and D. Xu, "Block proposal neural architecture search," *IEEE Transactions on Image Processing*, vol. 30, pp. 15–25, 2020.

[50] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervised deep neural networks for learning meaningful representations," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 89–103, 2018.

[51] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.

[52] Z. Lu, I. Whalen, Y. Dhebar, K. Deb, E. D. Goodman, W. Banzhaf, and V. N. Boddeti, "Multiobjective evolutionary design of deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 277–291, 2020.

[53] Z. Lu, G. Sreekumar, E. Goodman, W. Banzhaf, K. Deb, and V. N. Boddeti, "Neural architecture transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2971–2989, 2021.

[54] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *European Conference on Computer Vision*. Springer, 2020, pp. 544–560.

[55] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," *arXiv preprint arXiv:1804.09081*, 2018.

[56] Y.-H. Kim, B. Reddy, S. Yun, and C. Seo, "Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy," in *ICML 2017 AutoML workshop*, 2017.

[57] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[58] Y.-W. Wen, S.-H. Peng, and C.-K. Ting, "Two-stage evolutionary neural architecture search for transfer learning," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 928–940, 2021.

[59] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1310–1322, 2019.

[60] ——, "Real-time federated evolutionary neural architecture search," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 2, pp. 364–378, 2021.

[61] H. Zhu, H. Zhang, and Y. Jin, "From federated learning to federated neural architecture search: a survey," *Complex & Intelligent Systems*, vol. 7, no. 2, pp. 639–657, 2021.

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

[63] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: Bandit-based configuration evaluation for hyperparameter optimization," in *ICLR (Poster)*, 2017.

[64] H. Zhang, Y. Jin, and K. Hao, "Evolutionary search for complete neural network architectures with partial weight sharing," *IEEE Transactions on Evolutionary Computation*, 2022.

[65] J. Fang, Y. Sun, Q. Zhang, K. Peng, Y. Li, W. Liu, and X. Wang, "Fna++: Fast network adaptation via parameter remapping and architecture search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2990–3004, 2020.

[66] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations*, 2020.

[67] A. Zela, A. Klein, S. Falkner, and F. Hutter, "Towards automated deep learning: Efficient joint neural architecture and hyperparameter search," *arXiv preprint arXiv:1807.06906*, 2018.

[68] S. Yang, Y. Tian, X. Xiang, S. Peng, and X. Zhang,

"Accelerating evolutionary neural architecture search via multi-fidelity evaluation," *IEEE Transactions on Cognitive and Developmental Systems*, 2022.

[69] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A down-sampled variant of imagenet as an alternative to the cifar datasets," *arXiv preprint arXiv:1707.08819*, 2017.

[70] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast bayesian optimization of machine learning hyper-parameters on large datasets," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 528–536.

[71] Y. Xu, L. Xie, W. Dai, X. Zhang, X. Chen, G.-J. Qi, H. Xiong, and Q. Tian, "Partially-connected neural architecture search for reduced computational redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2953–2970, 2021.

[72] B. Moser, F. Raue, J. Hees, and A. Dengel, "Less is more: Proxy datasets in nas approaches," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1953–1961.

[73] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *International Conference on Learning Representations*, 2018.

[74] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[75] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "Cars: Continuous evolution for efficient neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1829–1838.

[76] D. O'Neill, B. Xue, and M. Zhang, "Evolutionary neural architecture search for high-dimensional skip-connection structures on densenet style networks," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 6, pp. 1118–1132, 2021.

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

[78] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[79] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *International Conference on Machine Learning*. PMLR, 2018, pp. 550–559.

[80] H. Zhang, Y. Jin, R. Cheng, and K. Hao, "Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 371–385, 2020.

[81] Y. Benyahia, K. Yu, K. B. Smires, M. Jaggi, A. C. Davison, M. Salzmann, and C. Musat, "Overcoming multi-model forgetting," in *International Conference on Machine Learning*. PMLR, 2019, pp. 594–603.

[82] X. Chu, B. Zhang, and R. Xu, "Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 239–12 248.

[83] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. L. P. Chen, "Bnas: Efficient neural architecture search using broad scalable architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 5004–5018, 2022.

[84] Z. Zhou, X. Ning, Y. Cai, J. Han, Y. Deng, Y. Dong, H. Yang, and Y. Wang, "Close: Curriculum learning on the sharing extent towards better one-shot nas," *arXiv preprint arXiv:2207.07868*, 2022.

[85] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1294–1303.

[86] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," in *International Conference on Learning Representations*, 2018.

[87] Y. Zhou, X. Xie, and S.-Y. Kung, "Exploiting operation importance for differentiable neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[88] H. Wang, R. Yang, D. Huang, and Y. Wang, "idarts: Improving darts by node normalization and decorrelation discretization," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2021.

[89] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," in *International Conference on Learning Representations*, 2019.

[90] Z. Zhou, T. Li, Z. Zhang, Z. Zhao, C. Sun, R. Yan, and X. Chen, "Bayesian differentiable architecture search for efficient domain matching fault diagnosis," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.

[91] H. Tan, R. Cheng, S. Huang, C. He, C. Qiu, F. Yang, and P. Luo, "Relativenas: Relative neural architecture search via slow-fast learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[92] T. Liang, Y. Wang, Z. Tang, G. Hu, and H. Ling, "Opanas: One-shot path aggregation network architecture search for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 195–10 203.

[93] Y. Xiong, H. Liu, S. Gupta, B. Akin, G. Bender, Y. Wang, P.-J. Kindermans, M. Tan, V. Singh, and B. Chen, "Mobiledets: Searching for object detection architectures for mobile accelerators," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3825–3834.

[94] J. Guo, K. Han, Y. Wang, C. Zhang, Z. Yang, H. Wu, X. Chen, and C. Xu, "Hit-detector: Hierarchical trinity architecture search for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 405–11 414.

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

[96] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641*, 2015.

[97] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Reinforcement learning for architecture search by network transformation," *arXiv preprint arXiv:1707.04873*, vol. 4, 2017.

[98] ——, "Efficient architecture search by network transformation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[99] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[100] ——, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft computing*, vol. 9, no. 1, pp. 3–12, 2005.

[101] J. Tao and G. Sun, "Application of deep learning based multi-fidelity surrogate model to robust aerodynamic design optimization," *Aerospace Science and Technology*, vol. 92, pp. 722–737, 2019.

[102] C. Sun, Y. Jin, and Y. Tan, "Semi-supervised learning assisted particle swarm optimization of computationally expensive problems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 45–52.

[103] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, "Transfer learning based co-surrogate assisted evolutionary bi-objective optimization for objectives with non-uniform evaluation times," *Evolutionary computation*, pp. 1–27, 2021.

[104] H. Wang, Y. Jin, and J. O. Jansen, "Data-driven surrogate-assisted multiobjective evolutionary optimization of a trauma system," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 6, pp. 939–952, 2016.

[105] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 442–458, 2018.

[106] C. Sun, H. Wang, and Y. Jin, *Data-Driven Evolutionary Optimization: Integrating Evolutionary Computation, Machine Learning and Data Science*. Springer, 2021.

[107] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[108] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[109] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of bayesian methods for seeking the extremum," *Towards global optimization*, vol. 2, no. 117-129, p. 2, 1978.

[110] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the International Conference on Machine Learning*, 2010.

[111] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization." *Journal of Machine Learning Research*, vol. 13, no. 6, 2012.

[112] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 1946–1956.

[113] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[114] B. Ru, X. Wan, X. Dong, and M. Osborne, "Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels," *arXiv preprint arXiv:2006.07556*, 2020.

[115] C. White, W. Neiswanger, and Y. Savani, "Bananas: Bayesian optimization with neural architectures for neural architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 293–10 301.

[116] H. Shi, R. Pi, H. Xu, Z. Li, J. Kwok, and T. Zhang, "Bridging the gap between sample-based and one-shot neural architecture search with bonas," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1808–1819, 2020.

[117] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[118] L. Ma, J. Cui, and B. Yang, "Deep neural architecture search with deep graph bayesian optimization," in *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 2019, pp. 500–507.

[119] H. Zhou, M. Yang, J. Wang, and W. Pan, "Bayesnas: A bayesian approach for neural architecture search," in *International conference on machine learning*. PMLR, 2019, pp. 7603–7613.

[120] R. Ru, P. Esperanca, and F. M. Carlucci, "Neural architecture generator optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 057–12 069, 2020.

[121] Z. Li, T. Xi, J. Deng, G. Zhang, S. Wen, and R. He, "Gp-nas: Gaussian process based neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 933–11 942.

[122] I. Trofimov, N. Klyuchnikov, M. Salnikov, A. Filippov, and E. Burnaev, "Multi-fidelity neural architecture search with knowledge distillation," *arXiv preprint arXiv:2006.08341*, 2020.

[123] C. Wei, C. Niu, Y. Tang, Y. Wang, H. Hu, and J. Liang, "Npenas: Neural predictor guided evolution for neural

architecture search," *arXiv preprint arXiv:2003.12857*, 2020.

[124] H. Cho, J. Shin, and W. Rhee, "B2ea: An evolutionary algorithm assisted by two bayesian optimization modules for neural architecture search," *arXiv preprint arXiv:2202.03005*, 2022.

[125] C. White, W. Neiswanger, S. Nolen, and Y. Savani, "A study on encodings for neural architecture search," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 309–20 319, 2020.

[126] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4780–4789.

[127] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," *arXiv preprint arXiv:1711.00436*, 2017.

[128] H. Zhu and Y. Jin, "Toward real-time federated evolutionary neural architecture search," in *Automated Design of Machine Learning and Search Algorithms*. Springer, 2021, pp. 133–147.

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

[130] 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 Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 350–364, 2019.

[131] Y. Sun, X. Sun, Y. Fang, G. G. Yen, and Y. Liu, "A novel training protocol for performance predictors of evolutionary neural architecture search algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 524–536, 2021.

[132] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[133] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter, "Learning curve prediction with bayesian neural networks," in *International Conference on Learning Representations*, 2016.

[134] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," *arXiv preprint arXiv:1705.10823*, 2017.

[135] A. Rawal and R. Miikkulainen, "From nodes to networks: Evolving recurrent neural networks," *arXiv preprint arXiv:1803.04439*, 2018.

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

[137] B. Greenwood and T. McDonnell, "Surrogate-assisted neuroevolution," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 1048–1056.

[138] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[139] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated cnn architecture design based on blocks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 4, pp. 1242–1254, 2019.

[140] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[141] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "Nas-bench-101: Towards reproducible neural architecture search," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7105–7114.

[142] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," *arXiv preprint arXiv:2001.00326*, 2020.

[143] W. Wen, H. Liu, Y. Chen, H. Li, G. Bender, and P.-J. Kindermans, "Neural predictor for neural architecture search," in *European Conference on Computer Vision*. Springer, 2020, pp. 660–676.

[144] Y. Peng, A. Song, V. Ciesielski, H. M. Fayek, and X. Chang, "Pre-nas: Predictor-assisted evolutionary neural architecture search," *arXiv preprint arXiv:2204.12726*, 2022.

[145] J. Liu, R. Cheng, and Y. Jin, "Bi-fidelity evolutionary multiobjective search for adversarially robust deep neural architectures," *arXiv preprint arXiv:2207.05321*, 2022.

[146] W. Ying, K. Yang, Y. Wu, J. Li, Z. Zhou, and B. Huang, "Multi-objective evolutionary architecture search of u-net with diamond atrous convolution," in *International Symposium on Intelligence Computation and Applications*. Springer, 2022, pp. 31–40.

[147] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[148] J. Lukasik, D. Friede, H. Stuckenschmidt, and M. Keuper, "Neural architecture performance prediction using graph neural networks," in *DAGM German Conference on Pattern Recognition*. Springer, 2020, pp. 188–201.

[149] Y. Tang, Y. Wang, Y. Xu, H. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "A semi-supervised assessor of neural architectures," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1810–1819.

[150] X. Ning, Y. Zheng, T. Zhao, Y. Wang, and H. Yang, "A generic graph-based neural architecture encoding scheme for predictor-based nas," in *European Conference on Computer Vision*. Springer, 2020, pp. 189–204.

[151] G. Kyriakides and K. Margaritis, "Evolving graph convolutional networks for neural architecture search," *Neural Computing and Applications*, vol. 34, no. 2, pp. 899–909, 2022.

[152] Y. Xu, Y. Wang, K. Han, Y. Tang, S. Jui, C. Xu, and C. Xu, "Renas: Relativistic evaluation of neural architecture search," in *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, 2021, pp. 4411–4420.

[153] H. Benmeziane, S. Niar, H. Ouarnoughi, and K. El Maghraoui, "Pareto rank surrogate model for hardware-aware neural architecture search," in *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2022, pp. 267–276.

[154] M. Huang, Z. Huang, C. Li, X. Chen, H. Xu, Z. Li, and X. Liang, "Arch-graph: Acyclic architecture relation predictor for task-transferable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 881–11 891.

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

[156] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[157] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[158] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[159] H. Zhu, H. Zhang, and Y. Jin, "From federated learning to federated neural architecture search: a survey," *Complex & Intelligent Systems*, vol. 7, no. 2, pp. 639–657, 2021.

[160] C. He, M. Annavaram, and S. Avestimehr, "Towards non-iid and invisible data with fednas: federated deep learning via neural architecture search," *arXiv preprint arXiv:2004.08546*, 2020.

[161] X. Liang, Y. Liu, J. Luo, Y. He, T. Chen, and Q. Yang, "Self-supervised cross-silo federated neural architecture search," *arXiv preprint arXiv:2101.11896*, 2021.

[162] I. Singh, H. Zhou, K. Yang, M. Ding, B. Lin, and P. Xie, "Differentially-private federated neural architecture search," *arXiv preprint arXiv:2006.10559*, 2020.

[163] M. Xu, Y. Zhao, K. Bian, G. Huang, Q. Mei, and X. Liu, "Federated neural architecture search," *arXiv preprint arXiv:2002.06352*, 2020.

[164] X. Liu, J. Zhao, J. Li, B. Cao, and Z. Lv, "Federated neural architecture search for medical data security," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5628–5636, 2022.

[165] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[166] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[167] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014.

[168] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[169] J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun, "Dpp-net: Device-aware progressive search for pareto-optimal neural architectures," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 517–531.

[170] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *European Conference on Computer Vision*. Springer, 2020, pp. 35–51.

[171] Z. Lu, R. Cheng, S. Huang, H. Zhang, C. Qiu, and F. Yang, "Surrogate-assisted multi-objective neural architecture search for real-time semantic segmentation," *arXiv preprint arXiv:2208.06820*, 2022.

[172] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *International Conference on Learning Representations*, 2018.

[173] L. Chen and H. Xu, "Mfenas: multifactorial evolution for neural architecture search," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 631–634.

[174] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 19–34.

[175] Y. Zhou, X. Xie, and S.-Y. Kung, "Exploiting operation importance for differentiable neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2021.

[176] Y. Chen, R. Gao, F. Liu, and D. Zhao, "Modulenet: Knowledge-inherited neural architecture search," *IEEE Transactions on Cybernetics*, pp. 1–11, 2021.

[177] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. L. P. Chen, "Bnas: Efficient neural architecture search using broad scalable architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 5004–5018, 2022.

[178] X. Dong and Y. Yang, "Searching for a robust neural architecture in four gpu hours," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2020.

[179] M. Zhang, H. Li, S. Pan, X. Chang, C. Zhou, Z. Ge, and S. Su, "One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2921–2935, 2021.

[180] Y. Guo, Y. Zheng, M. Tan, Q. Chen, Z. Li, J. Chen, P. Zhao, and J. Huang, "Towards accurate and compact architectures via neural architecture transformer," *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 2021.

[181] T. Elsken, J.-H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," *arXiv preprint arXiv:1711.04528*, 2017.

[182] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[183] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (Csur)*, vol. 51, no. 4, pp. 1–35, 2018.

[184] H. Sun, C. Wang, Z. Zhu, X. Ning, G. Dai, H. Yang, and Y. Wang, "Gibbon: efficient co-exploration of nn model and processing-in-memory architecture," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 867–872.