MedNAS: Multi-Scale Training-Free Neural Architecture Search for Medical Image Analysis

Yan Wang, Liangli Zhen, Jianwei Zhang, Miqing Li, Lei Zhang, Zizhou Wang, Yangqin Feng, Yu Xue, Xiao Wang, Zheng Chen, Tao Luo, Rich Siow Mong Goh, Yong Liu

***Abstract*—Deep neural networks have demonstrated impressive results in medical image analysis, but designing suitable architectures for each specific task is expertise-dependent and time-consuming. Neural architecture search (NAS) offers an effective means of discovering architectures. It has been highly successful in numerous applications, particularly in natural image classification. Yet, medical images possess unique characteristics, such as small regions and a wide variety of lesion sizes, that differentiate them from natural images. Furthermore, most current NAS methods struggle with high computational costs, especially when dealing with high-resolution image datasets. In this paper, we present a novel evolutionary neural architecture search method called Multi-Scale Training-Free Neural Architecture Search to address these challenges. Specifically, to accommodate the broad range of lesion region sizes in disease diagnosis, we develop a new reduction cell search space that enables the search algorithm to explicitly identify the optimal scale combination for multi-scale feature extraction. To overcome the issue of high computational costs, we utilize training-free indicators as performance measures for candidate architectures, which allows us to search for the optimal architecture more efficiently. More specifically, by considering the capability and simplicity of various networks, we formulate a multi-objective optimization problem that involves two training-free indicators and model complexity for candidate architectures. Extensive experiments on a large medical image benchmark and a publicly available breast cancer detection dataset are conducted. The empirical results demonstrate that our MSTF-NAS outperforms both human-designed architectures and current state-of-the-art NAS algorithms on both datasets, indicating the effectiveness of our proposed method.**

***Index Terms*—Neural architecture search, evolutionary optimization, training-free architecture search, medical image classification**

I. Introduction

DEEP neural networks have achieved remarkable success in various tasks, including image classification [1]–[3], object detection [4], [5], and segmentation [6], [7] in both natural and medical research domains. The success of these networks can be attributed, in part, to well-designed architectures such as AlexNet [8], ResNet [9], and SENet [10], which have achieved top performance in the ImageNet Large-Scale Visual Recognition Challenge for the classification task. These human-designed methods incorporate effective structures to enhance model performance. However, in the context of medical image analysis, different architectures often yield vastly different performances on various datasets and diseases. Therefore, designing architectures tailored to specific datasets and tasks becomes crucial to account for medical image characteristics, such as small regions of interest and a wide range of lesion sizes, in order to further improve disease detection performance. Additionally, the manual design of neural networks is a trial-and-error process that requires numerous attempts and relies heavily on expert experience [11], [12].

In contrast to human-designed methods, neural architecture search (NAS) automatically identifies the optimal architecture within a given search space [13]. The search space contains all possible architecture combinations, making it challenging to find the optimal one in a limited time. For instance, the DARTS search space encompasses approximately 10^{18} candidate architectures when searching for normal and reduction cells simultaneously. Thus, an effective search algorithm is essential to expedite the process of finding the optimal architecture. Currently, NAS methods employ three categories of search algorithms: reinforcement learning [14] (RL), evolutionary algorithms [15], [16] (EA), and gradient-based methods [17]. Most of the existing NAS methods are validation-based methods. Specifically, during the search process, these methods must train each candidate model for several epochs and test it on a validation set to obtain the approximate performance of this model. Obviously, there will be hundreds of thousands of candidate architectures during the search procedure. The process of training and validating each model can be time-consuming and computationally expensive.

To mitigate the time cost of the search process, a training-free-based approach has been recently proposed [18]. Training-free-based methods utilize performance estimators to approximate performance on a specific dataset by forwarding a small batch of data or backward the gradients of weights. Performance estimators are usually computed by defining different kinds of indicators that measure the effect of different combinations of the operations of the search space after building the operation into an architecture. By replacing the training and validation processes with indicators, the time cost of the NAS search algorithm has been significantly reduced.

This work was supported by the National Research Foundation of Singapore under its AI Singapore Programme (AISG Award No.: AISG2-TC-2021-003). (Corresponding author: Liangli Zhen, e-mail: zhenll@ihpc.a-star.edu.sg).

Yan Wang, Liangli Zhen, Zizhou Wang, Yangqin Feng, Tao Luo, Rick Siow Mong Goh, and Yong Liu are with the Institute of High Performance Computing, A*STAR, Singapore 138632.

Jianwei Zhang and Lei Zhang are with the Machine Intelligence Laboratory, College of Computer Science, Sichuan University, Chengdu 610065, China.

Miqing Li is with the School of Computer Science, University of Birmingham, Birmingham B15 2TT, United Kingdom.

Yu Xue is with the School of Computer Science, Nanjing University of Information Science and Technology, Nanjing 210044, China.

Xiao Wang is with the Oak Ridge National Laboratory, Oak Ridge TN 37830, United States.

Zheng Chen is with the Graduate School of Engineering Science, Osaka University, Osaka 565-0871, Japan.

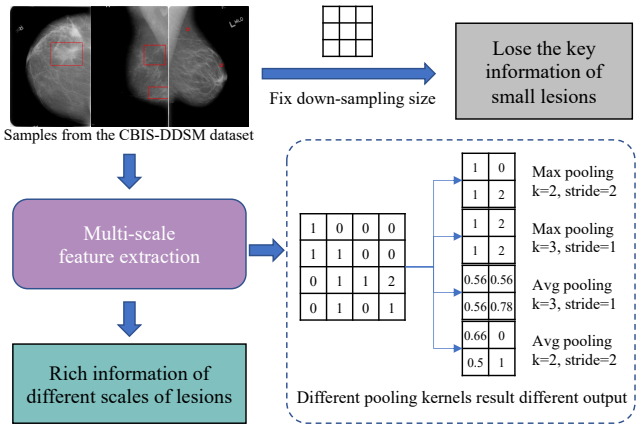

Fig. 1. Multi-scale lesion regions vs. multi-scale pooling. Several sample images from the CBIS-DDSM dataset exhibit a wide range of breast cancer lesion sizes. Different pooling operations summarize information from different regions with different perception scopes.

However, most of the existing training-free methods are designed for natural image classification. The key difference between medical images and natural images may cause the performance drop of existing methods. Specifically, medical images have a small region of interest and a wide variety of lesions.

In this paper, we present a novel evolutionary neural architecture search method called Multi-Scale Training-Free Neural Architecture Search (MSTF-NAS) to search for an architecture with a multi-scale ability that can perform well on medical image datasets with a large range of lesion region sizes. In our method, "multi-scale" refers to the utilization of multiple pooling kernel sizes designed to summarize features from feature maps while preserving essential information. To achieve this goal, we design two different search subspaces: one for normal cell searching and another for reduction cell searching. The normal cell search space is similar to existing search spaces like the commonly used DARTS search space. The reduction cell search space is designed to extract and summarize the information from different scales of input features. We employ different pooling operations with different pooling sizes to perceive input features in various scales. A toy example of different pooling operations is shown in Fig 1, from which we can see that different pooling sizes and types make a different summary of input features. It is highly advantageous to extract the features from the data which have a small region of interest and a large range of lesion sizes. Moreover, by considering the capability and simplicity of various networks, we formulate a multi-objective optimization problem that involves two training-free indicators and model complexity to search for promising candidate architectures efficiently.

We outline the novelty and main contributions of this work as follows:

- We propose a novel multi-scale training-free neural architecture search (NAS) method specifically designed for effectively searching for the optimal architecture within medical image datasets, even when dealing with high-resolution images. By adopting this approach, we are able to identify suitable architectures that effectively address the challenges posed by medical image data.
- We introduce a novel search space that incorporates various pooling operations with diverse pooling sizes, which enables the exploration of architectures that can effectively capture and retain multi-scale information. Our method addresses the challenge of small regions of interest and accommodates a wide range of lesion sizes present in medical images.
- In contrast to existing training-free NAS approaches, our methodology extends beyond solely assessing the capability of candidate architectures. We introduce a novel perspective by integrating the simplicity of architectures into the search process and formulating a new multi-objective optimization problem. By adopting this comprehensive approach, we can search for optimal architectures that not only possess superior capability but also demonstrate reasonable simplicity, ultimately aiming to achieve promising levels of diagnostic accuracy.

The remainder of this paper is organized as follows: We review related work in Section II. In Section III, we present the details of our proposed method. The experimental setup and results are reported in Section IV. Finally, Section V concludes the paper.

II. Related Work

This section provides a detailed review of the most relevant research studies related to our proposed method. Specifically, we will delve into the topics of multi-scale feature extraction, neural architecture search, and training-free indicators.

A. Multi-Scale Feature Extraction

Multi-scale feature extraction methods aim to extract features from input images with different scales of perception. Various methods involve resizing the input image into different sizes and using a backbone to extract the different scale features. For instance, Chen *et al.* proposed a scale-aware attention-based model that improves segmentation performance. This method resizes the input image into different sizes and employs a shared backbone to extract features from the different-sized images. Then, an attention mechanism is used to softly fuse the different scales of features [19]. Similarly, Fang *et al.* proposed a multi-scale feature abstraction method with a pyramid convolutional structure [20]. By resizing the input image into different sizes, the model can learn different scale global features, which has improved the organ segmentation performance. Besides resizing the input image to obtain the multi-scale features, MIMS-CNN is a method that resizes the feature maps of the backbone into different sizes, then uses a convolutional layer to learn the multi-scale features [21]. Unlike resizing either input image or feature maps, Su *et al.* proposed a multi-scale U-Net for medical image segmentation by employing multiple kernel sizes for extraction of semantic features from images. Multi-scale features also make features diverse [22].

Unlike these methods, we utilize different pooling kernels to construct a reduction cell to summarize the multi-scale feature. Furthermore, the reduction cell is searched automatically with our proposed NAS algorithm.

B. Neural Architecture Search

NAS is an approach that discovers the optimal neural architecture automatically from a predefined search space, including various operations [14], [15], [17], [23]–[26], such as 3×3 convolutional layers, pooling layers, and dense connections. Most NAS methods are based on the gradient optimization strategy, which treats the architecture search as an optimization problem. This method relaxes the discrete and non-differentiable architecture representation into continuous and differentiable [17], [27], [28]. Besides, some NAS methods use EA to generate the optimal architecture by the iterative crossovers and mutations of the population [15], [29]–[32] or use the RL algorithm as the controller to build the optimal architecture by exploring and exploiting the performance of different architectures [14], [33], [34]. To enable the candidate architectures to have multi-scale feature learning ability, Yan *et al.* proposed multi-scale NAS (MS-NAS) for medical image segmentation. In MS-NAS, a multi-scale framework with a multi-scale cell search space and multi-scale feature fusion is defined. Similarly, CLEARER is proposed to search the multi-scale architecture for image restoration [35]. NAS-count proposes a way to search a series of cascaded pooling layers and constructs an SPPLoss to resolve the scale variation problem in counting-by-density [36]. Wong *et al.* propose a NAS method to discover the optimal combinations of 3D pooling operations to reduce computational resource consumption, and each reduction layer contains one of the pooling operations [37]. Furthermore, only a limited number of training-free indicator-based NAS methods have been introduced. FreeREA leverages an optimized combination of training-free metrics to rank architectures during the search process [38]. Yang *et al.* introduce a lightweight training-based metric for searching high-performance architectures with fewer parameters by integrating training-free indicators [39]. Additionally, Do *et al.* propose a multi-objective NAS approach that employs the neural tangent kernel, number of linear regions, and FLOPs as three distinct objectives [40]. However, these NAS methods are designed for either natural image or medical image segmentation, and few works have studied the NAS application in automated disease diagnosis. Bae *et al.* propose a new search space containing different 3D pooling operations with different kernel sizes and multiple 3D convolutional operations and try to reduce the search time and computation resources. Kwasigroch *et al.* introduced NAS with the hill-climbing search strategy to search for the specific architecture for malignant melanoma detection with less computational cost on a real-world dataset. Zhang *et al.* proposed a multi-objective evolutionary zero-shot NAS framework for medical image classification [41].

Different from these multi-scale NAS methods and NAS for medical image classification, we present a new multi-scale search space for reduction cell search and aim to search for a model that can effectively handle the large range of lesion region sizes.

C. Training-Free Indicators

For NAS, a key challenge is the high computational costs, as candidate architectures need to be trained and validated to evaluate their performance. To tackle this challenge, one way is to build a performance estimator that predicts the performance of candidate architectures [24], [42], [43]. However, extra training is required to train the predictor to predict the model's performance accurately. Recently, training-free or training-free proxies NAS have been studied to reduce computational costs. They typically define an indicator or measurement to estimate the performance of candidate architectures instead of training and validating them on a specific dataset. For example, Chen *et al.* proposed using the number of the neural tangent kernel (NTK) and the number of linear regions to estimate the architecture performance [44]. Then, using a pruning-by-importance strategy to find the optimal architecture. Mellor *et al.* estimated the architecture's performance at the initial state by computing the Hamming distance between the binary codes of two inputs [45]. By combining the Hamming distance, which can reflect how dissimilar the two inputs are, with a simple search strategy, the architecture's performance is estimated by computing a single forward pass with a batch of data. Abdelfatta *et al.* adopted a series of pruning indicators that use just a batch of training data to estimate the performance of candidate architectures. They then used RL, EA, and predictor-based search algorithms to search for the architecture. The indicators include snip [46], grasp [47], synflow [48], fisher [49], and jacobian covariance [45].

Unlike existing training-free methods, our approach involves designing a new multi-scale search space. We carefully analyze both the features of the search space and the medical images to determine the appropriate indicators for searching the optimal architecture. Besides, we consider both the simplicity and the capability of the candidate architectures to search for the optimal architecture.

III. Our Proposed Method

Given a dataset $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{valid} \cup \mathcal{D}_{test}$, we aim to search the Pareto-optimal architectures by solving the following bi-level optimization problem:

$$\begin{aligned} &\min_{\alpha \in \Omega} \boldsymbol{F}(\alpha) = (f_1(\alpha; \boldsymbol{\theta}^*_\alpha), f_2(\alpha; \boldsymbol{\theta}^*_\alpha), \cdots, f_m(\alpha; \boldsymbol{\theta}^*_\alpha), \\ &\text{s.t.} \quad \boldsymbol{\theta}^*_\alpha = \arg\min_{\boldsymbol{\theta} \in \Theta} \mathcal{L}(\boldsymbol{\theta}; \alpha), \end{aligned} \tag{1}$$

where the upper-level problem defines an architecture candidate α in the search space Ω and minimizes m desired objective functions $f_1, f_2, \cdots, f_m$ simultaneously. The lower-level problem is to search for the optimal weight parameters for α via the loss function $\mathcal{L}(\boldsymbol{\theta}; \alpha)$ on the dataset $\mathcal{D}$.

To solve the above problem, we propose a novel method called MSTF-NAS, as illustrated in Fig. 2. Our approach utilizes an evolutionary search strategy framework, employing EA as the search algorithm. It consists of three essential elements: the search space, the formulation of a multi-objective

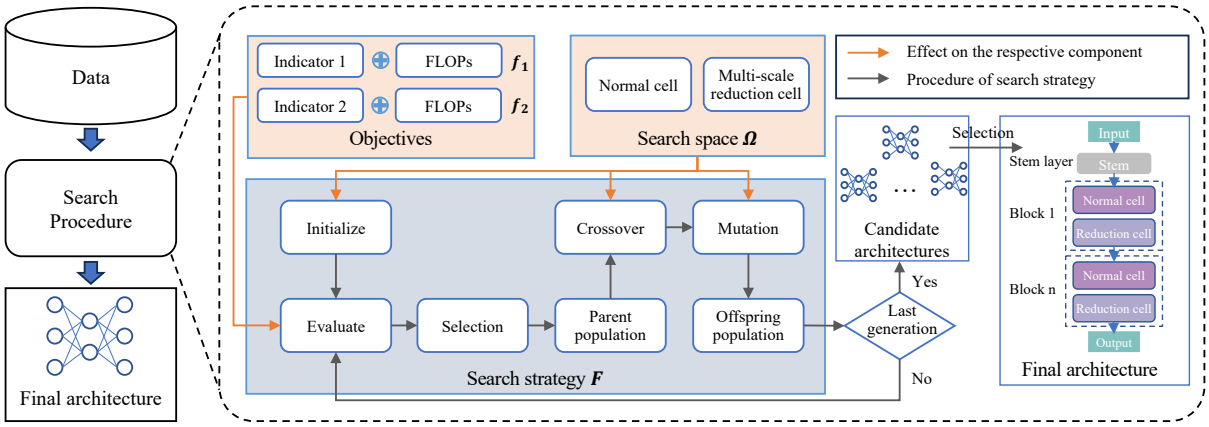

Fig. 2. The framework of our proposed MSTF-NAS algorithm. We adopt an evolutionary search strategy, which involves several key steps: randomly sampling initial architectures from the search space, evaluating the candidate architectures' performance by predefined objectives, generating candidate architectures by EA procedure, and looping the steps until the last generation. By following this framework, we can efficiently explore the space of possible architectures and identify those that best meet our desired objectives.

problem, and the evolutionary multi-objective architecture search. To begin, we establish the search space and proceed with the evolutionary search process. Within this process, we generate initial architectures by sampling from the search space. EA then generates the next generation by applying mutation, crossover, and selection, where the selection is guided by our designated objectives. We repeat the first two steps until the maximum generation is reached.

A. Search Space

The search space encompasses all the operations utilized in constructing neural architectures. An ideal search space should encompass a wide range of potential high-performance architectures while maintaining a manageable number of candidate architectures. This enables the search algorithm to discover the optimal architecture efficiently. In many previous studies, the search process simultaneously explores the normal cell and reduction cell within the search space [15], [17].In this study, we utilize the normal cell for feature extraction and the reduction cell for dimensionality reduction of feature maps. To effectively accomplish the objective of searching for a multi-scale feature extraction cell, we divide the search space into two distinct sub-spaces. In the normal cell search space, we define $o_N = 8$ operations, including 3×3, 5×5, and 7×7 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 7×1 followed by 1×7 convolutions, Identity, and *zero*. In the reduction cell search space, $o_R = 6$ different pooling operations with different pooling sizes are defined, which include 2×2 max and average pooling, 3×3 max and average pooling, and 4×4 max and average pooling. All the pooling operations have a stride of two. By including different pooling operations, we can construct a reduction cell in our reduction search space that can summarize feature maps on a multi-scale, thereby enhancing the diversity of features to cover various sizes of lesions.

The structure of the normal cells is the same as previous methods [15], [17], where the number of nodes in a normal cell is $n_N = 5$. The operation of each node is chosen from the normal cell search space by the search algorithm. The input of the normal cell is the feature maps from the last layer. The search algorithm aims to find the optimal combination of operations in the normal search space that allows the normal cell to extract distinctive features.

The structure of a single reduction cell is shown in Figure 3. Each reduction cell includes four nodes, i.e., $n_R = 4$, and each node is a searched pooling operation. The inputs of the reduction cell are the output feature maps of the previous layer, and its output consists of more feature maps that reduce dimensionality. The search algorithm aims to find the optimal pooling operation for each node. To maintain the same channel dimensionality for both input and output feature maps, each pooling operation includes two layers. The first layer is a standard 2D pooling layer, and the second layer is a 2D convolutional layer with a 1×1 kernel size and $\frac{1}{4}$ the number of input channel numbers.

During the architecture search process, two search strategies are commonly employed: micro-search and macro-search. Micro-search is a cell-based strategy that searches for one normal cell, replaces several layers, and inserts the pooling layer at predefined layers to construct the final architecture. On the other hand, the macro-search strategy searches for the entire architecture at once. Our approach is a macro-based search method where all the blocks of the architecture are individually distinct and undergo a search process. Each block consists of a normal cell and a reduction cell. Assuming there are n_N nodes for each normal cell and n_R nodes for each reduction cell, the number of potential architectures for a block, within our defined search space, can be calculated as $(o_N)^{n_N} \times n_N! \times (o_R)^{n_R}$, where o_N and o_R represent the total number of operations in the normal and reduction search spaces, respectively.

B. Formulation of Multi-Objective Problem

An optimal neural architecture should strike a balance between effectively representing nonlinear complexity and maintaining a reasonable simplicity. The network's capability

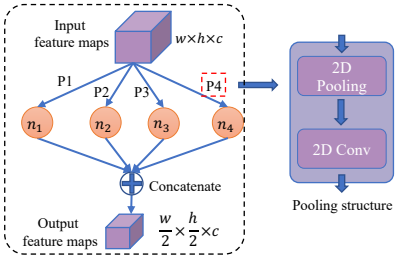

Fig. 3. The detailed illustration of the connections by constructing from our reduction search space.

describes its ability to represent nonlinear complexity, while simplicity implies that the network should achieve comparable performance with fewer trainable parameters, thereby avoiding the issue of over-parameterization. This is particularly crucial for medical datasets, which often have limited samples [50]. The capability of neural architectures can be reliably estimated using training-free indicators, as demonstrated by Abdelfattah *et al*. [18]. The simplicity of a neural architecture is influenced by the complexity of an architecture such as the number of layers, parameters, and the choice of activation functions. By considering the number of floating-point operations (FLOPs) required by architecture as an accurate and reliable proxy for network complexity [15], in our study, we utilize FLOPs of architecture as an indicator to approximate the simplicity of the network. During the architecture search process, we aim to optimize the trade-off between the capability and simplicity of architectures.

Specifically, we can adopt several types of training-free indicators [18], such as *grad_norm*, *snip*, *grasp*, synaptic flow (*synflow*), *fisher*, and Jacobian covariance (*jacob_cov*), to measure the capability of an architecture. In addition, the indicator of *grad_norm* measures the architecture performance by summing the Euclidean norm of the gradients computed at initialization using a single minibatch of data. The indicators of *grasp*, synaptic flow (*synflow*), *fisher* are defined as follows:

$$\begin{aligned}
&\text{snip: } \mathcal{S}_p = \left|\frac{\partial \mathcal{L}}{\partial \theta} \odot \theta\right|, \\
&\text{grasp: } \mathcal{S}_p(\theta) = -(H\frac{\partial \mathcal{L}}{\partial \theta}) \odot \theta, \\
&\text{synflow: } \mathcal{S}_p(\theta) = \frac{\partial L}{\partial \theta} \odot \theta, \\
&\text{fisher: } \mathcal{S}_z(z) = (\frac{\partial \mathcal{L}}{\partial z})^2, \mathcal{S}_n = \sum_{i=1}^{M} \mathcal{S}_z(z_i),
\end{aligned} \tag{2}$$

where $\mathcal{L}$, θ, and z denote the loss function, weights of architecture, and output feature maps from the last convolutional layer of architecture, respectively. H is the Hessian, and $\odot$ is the Hadamard product. It is worth noting that $\mathcal{S}_p$ calculated the saliency according to each weight parameter. Abdelfattah*et al.* used the sum of $\mathcal{S}_p$ over all N weights as the indicator $\mathcal{S}_z(z) = \sum_i^N \mathcal{S}_p(\theta)_i$. Similarly, the fisher uses the sum of $\mathcal{S}_z$ in terms of all the feature maps z_i for $i \in [1, 2, \ldots, M]$ as the indicator, where M is the number of feature maps. Lastly, the Jacobian covariance is computed as follows [45],

$$\text{jacob\_cov: } s = \log\left|\boldsymbol{K}_H\right|, \tag{3}$$

and

$$\boldsymbol{K}_H = \begin{pmatrix} N_A - d_H(\boldsymbol{c}_1, \boldsymbol{c}_1) & \cdots & N_A - d_H(\boldsymbol{c}_1, \boldsymbol{c}_N) \\ \vdots & \ddots & \vdots \\ N_A - d_H(\boldsymbol{c}_N, \boldsymbol{c}_1) & \cdots & N_A - d_H(\boldsymbol{c}_N, \boldsymbol{c}_N) \end{pmatrix}, \tag{4}$$

where N_A is the number of rectified linear units, $d_H(\boldsymbol{c}_i, \boldsymbol{c}_j)$ is the Hamming distance between two binary codes, which are computed from two different inputs. The Jacobian covariance represents the connection of activations in a network when exposed to various inputs. All six indicators demonstrate a positive relationship with the desired outcome, where larger values indicate better performance.

From the definitions of these indicators, we find that *grad_norm*, *snip*, *grasp*, and *synflow* are directly related to the weights of the architecture, with no explicit connections to the feature maps. Considering our goal is to learn features with multi-scale information, we propose to include at least one indicator associated with the saliency of feature maps like *fisher* or *jacob_cov*. In this work, we retain *fisher*, *grad_norm*, *synflow*, and *jacob_cov* as potential indicators when constructing the objective functions for architecture search since some previous studies [18] have verified that *grasp* and *snip* maybe fail to estimate the performance of architectures.

It is important to note that optimizing the multiobjective problem becomes increasingly challenging as the number of objectives involved increases. There are two main reasons for this phenomenon. Firstly, as the number of objectives grows, the selection pressure towards the Pareto front during the evolution process may diminish, leading to suboptimal solutions. Secondly, the solutions become increasingly sparse across the entire objective space, making it harder to find a well-distributed set of optimal solutions. Based on this consideration, we construct a problem that involves two training-free performance indicators and model complexity estimator (FLOPs) but with only two objectives defined as follows:

$$\begin{cases} f_1 = -p_1 - f_{flops}, \\ f_2 = -p_2 + f_{flops}, \end{cases} \tag{5}$$

where p_1 and p_2 are two performance indicators, and f_{flops} denotes the value of FLOPs for the architecture candidate.

As shown in Eq. (5), both model performance and complexity have been taken into account during the search procedure. We have verified that selecting two indicators is more precise than the previous single indicator-based methods [18] in Section IV-E. Moreover, we incorporate FLOPs into the objectives and enforce the two objectives to be conflicting, making the formulated problem suitable to be optimized with dominance-based EAs. Since the values of indicators and FLOPs are on different scales, we normalize the values of indicators and FLOPs from the same generation into the range $[0, 1]$ to ensure an equal contribution from both indicators and FLOPs.

C. Search Algorithm

Operation encoding: MSTF-NAS is based on the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [51], which is a multi-objective evolutionary optimization algorithm

that has been successfully utilized for handling various multi-objective problems [52], [53]. To leverage NSGA-II, we first encode the operation into a genotype, which is how NSGA-II searches for architectures. Since our method employs a macro search-based approach, each architecture starts with a predefined stem block [15]. Subsequently, it comprises three blocks, which are determined through the macro search strategy, with each block consisting of one normal cell and one reduction cell. Finally, a global average pooling is utilized to transfer the feature maps into a feature vector. The encoding of each block is in the same manner and the length of the encoded genotype of the entire architecture is three times of each block. Each block is encoded by a $(2 \times n_N + n_R)$-dimensional vector. Each block includes a normal cell and a reduction cell, so the genotype also encompasses these two parts. An example illustration of one normal cell and a reduction cell is shown at the top of Fig. 4. For genotype encoding, each node is represented with two dimensions in the normal cell: one denotes the operation, and another one denotes which node this operation connects to. For the reduction cell, only one dimension is used, and it represents the operation. Since all the operations are used to reduce the spatial dimensions, we concatenate all the nodes as the final output of the reduction cell. As mentioned in Section III-A, there are $o_N = 8$ operations in the normal cell and $o_R = 6$ operations in the reduction cell. Thus, we use 0 to 7 to represent the operations of the normal cell and 0 to 5 to represent the operations of the reduction cell. In our method, we set the number of nodes in the normal cell and reduction cell to be $n_N = 5$ and $n_R = 4$, respectively. Therefore, the genotype of our method is a 14-dimensional vector.

Search procedure: To efficiently and effectively search the architecture from the vast search space, we adopt a similar search procedure as NSGA-NET [15]. It is an iterative procedure that generatively produces architectures from the initial generation to the maximum number of generations. At the beginning of the search procedure, an initial population is randomly initialized. One sample of the population is one architecture. The population members compete with each other to survive and produce the next generation. For each iteration, some members are selected as parents to generate a certain number of offspring, which are the new architectures that comprise the new population. When generating the offspring, crossover and mutation are employed to ensure a large diversity among the offspring members. Crossover and mutation are utilized to explore the search space. Simultaneously, an exploitation procedure is used to exploit the existing well-constructed structures in a block.

Detailed architecture construction: Figure 4 illustrates an example of how an MSTF-NAS block is constructed, from the encoding of operations to the final architecture. First, it obtains the encoding genotype and then translates the genotype into the phenotype according to the operations and connections among different nodes. Next, it selects the nodes that need to be concatenated. Finally, using the phenotype information, it builds the architecture shown at the bottom of Fig. 4.

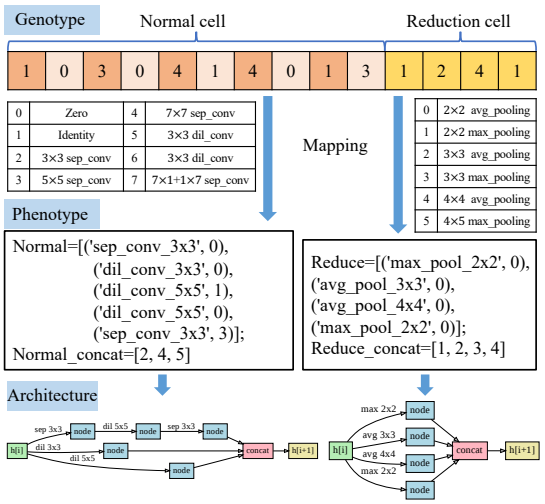

Fig. 4. Encoding of MSTF-NAS: The original architectures are represented by a 14-dimensional vector genotype, with the first 10 dimensions corresponding to the normal cell and the last 4 dimensions representing the reduction cell. The genotype is then transformed into the phenotype according to the search space mapping. Finally, the architecture is constructed based on the phenotype, as shown in the bottom architectures.

TABLE I
THE DETAILED DATASET INFORMATION AND THE NUMBER OF SAMPLES [54]. TR., VAL. AND TE. DENOTE THE TRAINING AND VALIDATION AND TEST SETS, RESPECTIVELY.

MedMNIST	Tasks (# Classes)	# Samples	# Tr. / Val. / Te.
Path	Multi-Class (9)	107180	89996 / 10004 / 7180
Chest	Multi-Label (14)	112120	78468 / 1219 / 22433
Derma	Multi-Class (7)	10015	7007 / 1003 / 2005
OCT	Multi-Class (4)	109309	97477 / 10832 / 1000
Pneumonia	Binary-Class (2)	5856	4708 / 524 / 624
Retina	Ordinal Regression (5)	1600	1080 / 120 / 400
Breast	Binary-Class (2)	780	546 / 78 / 156
Blood	Multi-Class (8)	17092	11959 / 1712 / 3421
Tissue	Multi-Class (8)	236386	165466 / 23640 / 47280
OrganA	Multi-Class (11)	58850	34581 / 6491 / 17778
OrganC	Multi-Class (11)	23660	13000 / 239 / 8268
OrganS	Multi-Class (11)	25221	13940 / 2452 / 8829

IV. EXPERIMENTAL STUDY

A. Datasets

To evaluate the effectiveness of our proposed method, we employ a large medical benchmark dataset, MEDMNIST V2 [54], and a mammography dataset, Curated Breast Imaging Subset of the Digital Database for Screening Mammography (CBIS-DDSM) [55]. MEDMNIST is the first large-scale medical image benchmark for evaluating computer vision tasks, including auto-machine learning algorithms. It contains twelve pre-processed 2D datasets and six 3D datasets which include diverse data modalities (such as X-Ray, OCT, Ultrasound, CT and *et al.*), diverse classification tasks (binary/multiclass classification, multi-label and ordinal regression) and different data scales (from hundreds to hundreds of thousands) for developing 2D and 3D neural network, respectively [54]. In our experiments, we leverage all the 2D image datasets to evaluate our method. The detailed dataset information and the number of samples are shown in Table I. All the tasks of MEDMNIST are classification tasks, including the ChestMNIST dataset, which is a multi-label binary classification task. The size of the images in this benchmark is pre-processed

to 28×28 pixels for standardizing the benchmark officially. The CBIS-DDSM dataset is a standard version of the DDSM dataset. There are 891 mass cases and 753 calcification cases. Each case may have one or two mammography images, so there are 3071 images in the CBIS-DDSM dataset in total, with 1353 malignant images and 1718 benign/normal images. The original images are stored in DICOM format, and the resolution is around 3000×4000 pixels. The CBIS-DDSM dataset has provided the official training (2489 images) and test (645 images) sets. To search the CBIS-DDSM, we first transform the DICOM image into JPEG format, then resize the original image to 224×224 pixels to search the architectures. For evaluation, we resize the original image to 800×800 pixels, following the preprocessing method used by Shu *et al.* [12].

B. Experimental Settings

1) Implementation Details: In searching for the optimal network for each dataset from MEDMNIST and CBIS-DDSM, we set the depth of the blocks to three and four, respectively, since the images in the CBIS-DDSM dataset have high resolutions. The NSGA-II settings are the same as those in NSGA-Net [15]. After the search procedure, we select the architecture with the highest value of the sum of the two training-free indicators from the last generation as the optimal architecture for further evaluation of classification performance. In the evaluation procedure, we follow the official guidelines of MEDMNIST [54] to train and evaluate the model's performance. For the evaluation of CBIS-DDSM, we employ the same strategy as Shu *et al.* [12]. The initial channel number for all architectures in our method search is set to 128, and the channel number will double with each subsequent block. Both the search and evaluation procedures are conducted on a single DGX A100 GPU card with 40 GB memory, using the PyTorch framework. For the training-free indicators in the objectives, they are selected by employing an ablation study as mentioned in section IV-E2. Finally, unless explicitly stated otherwise, we utilize *jacob_cov* and *synflow* to construct the objectives to search the optimal architectures.

2) Evaluation Metrics: To evaluate the searched architecture's performance, we use the metrics provided by the official guideline [54] and then most popular used metrics on CBIS-DDSM [11], [12], i.e., accuracy (ACC) and the area under the receiver operating characteristic curve (AUC) for both the MEDMNIST and CBIS-DDMS datasets. The values of ACC and AUC can be computed as follows:

$$\begin{cases} ACC = \dfrac{TP + TN}{TP + FP + TN + FP} \\ AUC = \displaystyle\int_0^1 \mathrm{ROC}(r)\, d_r \end{cases}, \tag{6}$$

where TP, TN, FP, and FN denote the true positive, true negative, false positive, and false negative, respectively. Moreover, $\mathrm{ROC}(r)$ represents the ROC curve (the true positive rate against the false positive rate) as a function of the false positive rate, and d_r denotes the differential element of the false positive rate.

C. Evaluation on MEDMNIST

To evaluate the effectiveness and efficiency of our proposed method, we compare it with two human-designed architectures, three auto-machine learning methods, four NAS methods, and the random search strategy. Specifically, we employ ResNet-18 and ResNet-50 [9] as the human-designed models. These two models are evaluated on two different sizes, i.e., 28×28 and 224×224. For the auto-machine learning methods, Auto-sklearn, AutoKeras, and Google AutoML Vision are employed. Auto-sklearn is an auto-search method for searching optimal classifiers, feature preprocessing methods, and data preprocessing methods [56]. AutoKeras is a Bayesian optimization-based NAS method for searching both neural architectures and hyperparameters [57]. Google AutoML Vision is a commercial AutoML tool. All the results of these mentioned models are provided by the MEDMNIST benchmark [54]. For NAS methods, we compare with DARTS [17], SNAS [58], NSGA-Net [15], and HOPNAS [59]. DARTS, SNAS, and HOPNAS are one-shot-based NAS methods that search the architectures from a pre-trained supernet. DARTS and NSGA-Net are designed for searching architectures for natural images. HOPNAS searches architectures by reducing the supernet dynamically for medical images. Additionally, the random search strategy is a commonly used NAS baseline that randomly selects architectures from the search space. Of all these methods, NSGA-Net is the most related to our work. However, the difference is that NSGA-Net is a validation-based NAS method, which requires training and validating each architecture on a specific dataset. In contrast, our method only needs to forward compute a batch of data.

Table II and III report the detailed experimental results in terms of accuracy and AUC score. The overall conclusion from the results is that our proposed method outperforms all other methods, as it achieved the highest average AUC score and the best average accuracy over 12 subsets. When comparing different types of methods, we find that human-designed architectures can achieve better performance compared to AutoML methods and can also outperform some NAS methods, such as DARTS and SNAS. The potential reason is that none of these methods are designed specifically for searching architectures for medical images. The domain gap between natural images and medical images leads to these results. Among all the NAS methods, the performances of one-shot-based methods (DARTS, SNAS, and HOPNAS) are inferior to the validation-based method (NSGA-NET) and our proposed method. The main reason for this phenomenon is that one-shot-based methods estimate candidate architectures' performance using a pre-trained supernet, which may not be well-trained. However, NSGA-NET estimates each candidate architecture by training and validating on the dataset, making it more precise, but at the cost of higher computational and time expenses. For our method, even though we estimate candidate architectures without training and validating on the dataset, we use multiple indicators to comprehensively estimate the performance of the candidate architectures. It is worth noting that the random search strategy based on our search space can achieve a similar performance to the previous SOTA method,

TABLE II
RESULTS OF DIFFERENT METHODS ON THE MEDMNIST BENCHMARK IN TERMS OF ACCURACY. AVG. DENOTES THE AVERAGE PERFORMANCE OVER THE 12 SUBSETS.

Methods	Path	Chest	Derma	OCT	Pneum	Retina	Breast	Blood	Tissue	OrganA	OrganC	OrganS	AVG.
ResNet-18 (28)	0.907	0.947	0.735	0.743	0.854	0.524	0.863	0.958	0.676	0.935	0.900	0.782	0.819
ResNet-18 (224)	0.909	0.947	0.754	0.763	0.864	0.493	0.833	0.963	0.681	0.951	0.920	0.778	0.821
ResNet-50 (28)	0.911	0.947	0.735	0.762	0.854	0.528	0.812	0.956	0.680	0.935	0.905	0.770	0.816
ResNet-50 (224)	0.892	**0.948**	0.731	0.776	0.884	0.511	0.842	0.950	0.680	0.947	0.911	0.785	0.821
auto-sklearn	0.716	0.779	0.719	0.601	0.855	0.515	0.803	0.878	0.532	0.762	0.829	0.672	0.722
AutoKeras	0.834	0.937	0.749	0.763	0.878	0.503	0.831	0.961	0.703	0.905	0.879	0.813	0.813
Google AutoML	0.728	**0.948**	0.768	0.771	**0.946**	0.531	0.861	0.966	0.673	0.886	0.877	0.749	0.809
DARTS	0.872	0.934	0.749	0.712	0.874	0.510	0.832	0.953	0.648	0.926	0.791	0.808	0.801
SNAS	0.850	0.938	0.737	0.708	0.871	0.515	0.811	0.946	0.708	0.918	0.891	0.778	0.806
HOPNAS	**0.912**	0.947	0.759	0.761	0.852	0.523	0.853	0.958	0.698	0.937	0.911	0.803	0.826
NSGA-NET	0.866	0.947	0.744	0.765	0.907	0.540	0.846	0.970	0.712	0.952	0.923	0.820	0.833
Random search	0.854	0.946	0.773	0.760	0.904	0.542	**0.897**	0.966	0.717	0.955	0.923	0.820	0.838
MSTF-NAS (Ours)	0.910	0.945	**0.774**	**0.780**	0.912	**0.550**	0.872	**0.976**	**0.740**	**0.962**	**0.936**	**0.838**	**0.850**

TABLE III
RESULTS OF DIFFERENT METHODS ON THE MEDMNIST BENCHMARK IN TERMS OF THE AUC SCORE. AVG. DENOTES THE AVERAGE PERFORMANCE OVER 12 SUBSETS.

Methods	Path	Chest	Derma	OCT	Pneum	Retina	Breast	Blood	Tissue	OrganA	OrganC	OrganS	AVG.
ResNet-18 (28)	0.983	0.768	0.917	0.943	0.944	0.717	0.901	0.998	0.930	0.997	0.992	0.972	0.922
ResNet-18 (224)	0.989	0.773	0.920	0.958	0.956	0.710	0.891	0.998	0.933	0.998	0.994	0.974	0.925
ResNet-50 (28)	**0.990**	0.769	0.913	0.952	0.948	0.726	0.857	0.997	0.931	0.997	0.992	0.972	0.920
ResNet-50 (224)	0.989	0.773	0.912	0.958	0.962	0.716	0.866	0.997	0.932	0.998	0.993	0.975	0.923
auto-sklearn	0.934	0.649	0.902	0.887	0.942	0.690	0.836	0.984	0.828	0.963	0.976	0.945	0.878
AutoKeras	0.959	0.742	0.915	0.955	0.947	0.719	0.871	0.998	0.941	0.994	0.990	0.974	0.917
Google AutoML	0.944	0.778	0.914	0.963	0.991	0.750	0.919	0.998	0.924	0.990	0.988	0.964	0.927
DARTS	0.975	0.732	0.913	0.953	0.965	0.742	0.912	0.994	0.901	0.987	0.969	0.910	0.913
SNAS	0.969	0.733	0.906	0.949	0.974	0.753	0.894	0.996	0.921	0.979	0.927	0.952	0.913
HOPNAS	0.987	0.763	0.899	0.948	0.971	**0.770**	0.907	0.996	0.913	0.995	**0.998**	0.975	0.927
NSGA-NET	0.979	0.779	0.915	0.958	0.965	0.759	0.857	**0.999**	0.942	**0.999**	0.993	0.978	0.927
Random search	0.980	0.774	0.923	0.956	**0.963**	0.750	0.921	**0.999**	0.944	**0.999**	0.994	0.982	0.932
MSTF-NAS (Ours)	**0.990**	**0.791**	**0.934**	**0.968**	**0.963**	0.755	**0.930**	**0.999**	**0.951**	**0.999**	0.996	**0.983**	**0.938**

TABLE IV
THE SEARCH TIME (MINUTES) COSTS OF DIFFERENT SEARCH ALGORITHMS ON THE MEDMNIST BENCHMARK. AVG. DENOTES THE AVERAGE TIME OVER 12 SUBSETS.

Methods	Path	Chest	Derma	OCT	Pneum	Retina	Breast	Blood	Tissue	OrganA	OrganC	OrganS	AVG.
Google AutoML	180	180	120	180	60	60	60	180	240	120	120	120	135
DARTS	1496	1184	121	1181	25	16	29	196	538	478	342	519	510
SNAS	691	585	45	443	11	7	13	474	1080	195	133	154	319
HOPNAS	465	390	38	372	**8**	**5**	**8**	55	985	152	92	112	224
NSGA-NET	2645	1067	223	3061	108	35	14	273	4147	799	280	281	1078
MSTF-NAS (Ours)	**34**	**24**	**21**	**24**	19	21	10	**20**	**31**	**23**	**20**	**16**	**22**

which shows the effectiveness of our search space. In another aspect, SNAS, DARTS, and NSGA-NET were originally designed for natural image classification and demonstrated strong performance on datasets such as Cifar10 and ImageNet. However, when tested on the 12 datasets included in the MEDMNIST benchmark, they exhibited subpar performance compared to our approach. Specifically, our method achieved improvements of 1.7% in average ACC and 1.1% in AUC. This performance disparity may be attributed to the fact that our method was purposefully tailored to handle a wide range of lesion size data, accomplished through innovative reduction cell construction and more refined search objectives during the search procedure.

Table IV demonstrates the search time of different search algorithms on the 12 datasets. We compare our method with the NAS algorithms that are mentioned in Table III and the most commonly used auto-machine learning method, i.e., Google AutoML. From Table IV, we can see that our method spends the minimum average search time across the 12 datasets. It has also been revealed that one-shot-based NAS and train-validation-based NAS are highly dependent on the dataset size. The larger the dataset size, the more time is spent searching for architectures. For example, PathMNIST has more than 100000 samples in the training set, 10 times the number of DermaMNIST, and the search time of these methods on PathMNIST is also around 10 times that of DermaMNIST. Our method has a weaker connection to the dataset size, and it spends only 64 minutes on search time, which is at least two times faster than other methods. The reduced search cost and stable search time across the 12 datasets benefit from the training-free search strategy, which only needs to feed one batch of data to estimate the performance of each candidate architecture. To further demonstrate the effectiveness of our method in searching for architectures in high-resolution imagery, we conduct an evaluation on the CBIS-DDSM dataset, setting the input image resolution to 224×224 pixels. We

TABLE V
RESULTS OF DIFFERENT BACKBONE MODELS ON CBIS-DDSM DATASET. 'W/' AND 'W/O' STAND FOR WITH AND WITHOUT, RESPECTIVELY.

Model	Setting	ACC (%)	AUC (%)
VGG-16		57.21	62.17
ResNet-18		62.17	67.46
ResNet-34		65.43	70.10
ResNet-50	w/o pre-trained	62.33	65.62
ResNet-101		62.17	67.84
DenseNet-121		62.48	68.37
ViT-B-16		59.69	62.10
ViT-L-16		61.24	63.79
MSTF-NAS (Ours)		**66.67**	**70.56**
VGG-16		69.45	75.24
ResNet-18		65.89	73.82
ResNet-34		69.15	76.37
ResNet-50	w/ pre-trained	66.98	76.27
ResNet-101		70.70	77.18
DenseNet-121		68.53	75.67
MSTF-NAS (Ours)		**71.63**	**80.22**

compare our method to our primary baseline, NSGA-NET, and report the time required for 40 generations of both methods. Notably, for NSGA-NET, we limit training to just five epochs to estimate the performance of each architecture. The results indicate a significant difference in time efficiency, with our method taking 115 minutes and NSGA-NET requiring 870 minutes. It is worth noting that the time represents training for only five epochs on a 2000 image training set. This stark contrast highlights our method's ability to accelerate the search process by approximately 8 times compared to NSGA-NET, confirming its effectiveness when dealing with high-resolution image datasets.

D. Evaluation on CBIS-DDSM

To further evaluate the effectiveness of our proposed method, we search for and evaluate the optimal architecture on the CBIS-DDSM dataset. As our MSTF-NAS is designed for searching backbones, we leverage the most commonly used backbones in breast cancer detection as our baselines. Specifically, we compare our method with VGG-16 [2], ResNet [9] series (ResNet-18, ResNet-34, ResNet-50, and ResNet-101), DenseNet-121 [1] and vision transformer (ViT) series [60] (ViT-B-16 and ViT-L-16). Since most previous studies have verified that ImageNet pre-trained weights can significantly improve the classification task on CBIS-DDSM [11], [12], we also use pre-trained models to evaluate the performance of all the models (with pre-trained). By pretraining our model on the ImageNet dataset, our model achieved 75.13% of the top 1 accuracy on the validation set. Additionally, we provide the performance without pre-trained weights (without pre-trained) to show the trainability of our searched architecture. We only compare ViT without pre-trained weights, because the ImageNet pre-trained ViTs require fixed input image size, i.e., 224×224, which is not fair compared with other models which use 800×800 image as input. The results for all the models are demonstrated in Table V. The accuracies of all the models are based on the optimal threshold, which is selected using the precision-recall curve.

From Table V, it can be observed that all models achieve higher AUC scores when using pre-trained weights. The AUC scores of all models with pre-trained weights improve by approximately 6%-13% when compared to models without pre-trained weights. More importantly, our MSTF-NAS achieves the highest ACC and AUC scores among all seven backbone models, both with and without pre-trained weights. Specifically, MSTF-NAS achieves 0.4% and 3.04% higher AUC scores than the best model of other backbones without and with pre-trained weights, respectively. In the comparison between ViTs and our approach, it becomes evident that ViTs exhibit subpar performance on this dataset. Notably, their performance surpasses that of VGG-16 only with respect to accuracy (ACC) and the area under the curve (AUC). Several potential factors contribute to this observation: 1) ViTs excel at capturing global features within input images, yet the presence of breast cancer lesions accounts for merely 1% of the image area, rendering them relatively small for effective recognition. 2) The training of a ViT model necessitates a substantial volume of images, and our dataset comprising 2489 images falls significantly short of what is required for comprehensive ViT model training [61]. On one hand, these results illustrate the effectiveness and reasonable simplicity of MSTF-NAS, where performance benefits from the well-designed search space and the effective search algorithm. On the other hand, the results reveal that backbones designed for natural images may not perform as effectively as backbones specifically tailored to the characteristics of medical data.

To assess the efficacy of classifying lesions of varying sizes, we partition the test dataset from CBIS-DDSM into three distinct subsets based on lesion size. The CBIS-DDSM dataset comprises a substantial number of samples featuring calcifications and masses, each exhibiting lesions spanning a wide range of scales, from 0.182 to 2.08×10^{-5}. This scale range signifies the proportion of lesion area relative to the entire image. Subsequently, we categorize the 645 test samples into three subsets based on lesion area proportions: large size (comprising 206 samples with proportions > 0.01), medium size (encompassing 409 samples with proportions ranging from 0.001 to 0.01), and small size (consisting of 30 samples with proportions < 0.001). We conduct evaluations on these three subsets using seven models pre-trained on ImageNet, with the AUC as the chosen evaluation metric, as it remains unaffected by sample distribution disparities. The specific classification outcomes are presented in Table VI. Our experimental findings conclusively demonstrate that our custom-designed model, MSZS-NET, achieves the highest AUC score across all three subsets, as well as an impressive average AUC score over the entire dataset. Our approach exhibits notable improvements, increasing AUC scores by 2.85%, 2.67%, and 2.88% on the large, medium, and small lesion size subsets, respectively. It has indicated that the lesion sizes are smaller, and a higher improvement of our method was achieved. These results substantiate the effectiveness of our multi-scale search strategy in enhancing performance across varying lesion sizes.

E. Ablation Study

To evaluate the effectiveness of each module of our proposed MSTF-NAS, we conduct the following ablation studies

TABLE VI
THE RESULTS OF DIFFERENT MODELS ON DIFFERENT LESION SIZE DATASETS.

Model	AUC			
	Large	Medium	Small	Average
ResNet-18	72.32	71.82	95.19	79.78
ResNet-34	77.07	73.27	90.38	80.24
ResNet-50	78.69	71.57	96.15	82.14
ResNet-101	79.18	73.65	97.12	83.32
VGG-16	74.09	73.58	84.62	77.43
DenseNet-121	74.69	72.66	87.50	78.28
MSZS-NET (Ours)	**82.03**	**76.32**	**100.00**	**86.12**

TABLE VII
RESULTS OF DIFFERENT POOLING OPERATIONS IN TERMS OF DIFFERENT LESION SIZES.

Pooling	AUC		
	Large	Medium	small
2x2 average	74.69	74.18	77.05
2x2 max	76.36	74.32	80.13
3x3 average	**80.29**	73.58	83.61
3x3 max	77.84	**76.89**	82.49
4x4 average	76.82	71.28	**83.78**
4x4 max	73.25	74.81	77.72

on one dataset of MEDMNIST and CBIS-DDSM datasets. We report the average performance over 12 subsets for the results on MEDMNIST.

1) The effect of pooling sizes on the lesion sizes: To investigate the impact of various pooling operations and different pooling sizes on lesions of varying sizes, we conduct an evaluation across multiple combinations. We utilize two distinct pooling operations, namely max pooling and average pooling, combined with three different pooling sizes: 2×2, 3×3, and 4×4. This evaluation is performed on datasets featuring lesions of different sizes, as outlined in Section IV-D. Specifically, we employ DenseNet-121 as our backbone architecture, replacing the original pooling operations with six alternatives: 2×2 average pooling and max pooling, 3×3 average pooling and max pooling, and 4×4 average pooling and max pooling. The resulting AUC scores on the CBIS-DDSM datasets for various lesion sizes are presented in Table VII. The outcomes reveal distinct trends: For the large lesion dataset, 3×3 average pooling achieves the highest performance. In the case of the medium lesion dataset, 3×3 max pooling outperforms other configurations. The small lesion dataset benefits most from 4×4 average pooling. Furthermore, it is noteworthy that the performance of different pooling operations exhibits a considerable range across the various lesion size datasets, with AUC values spanning from 73.25% to 80.29% for large lesions, 71.28% to 76.89% for medium-sized lesions, and 77.05% to 83.78% for small lesions. Moreover, 3×3 pooling size performs best on large and medium lesion size subsets and 4×4 pooling size performs best on small lesion size subset. These findings underscore the significant impact of pooling operations on feature learning across different lesion sizes.

2) Indicator Selection: To construct effective objectives for searching for the optimal architecture, we evaluate different combinations of training-free indicators and choose the best combination to construct the objectives. We examine the combinations of the *jacob_cov*, *fisher*, *synflow*, and *grad_norm* indicators, which have been demonstrated to have the strongest relationship to architecture performance [18]. The objectives for all combinations are constructed using our designed method, as described in Eq. (5). For each combination approach, we conduct searches on 12 subsets of the MEDMNIST benchmark using our method and select the best architectures from the Pareto front for each subset. Subsequently, we train and test these selected architectures on each subset individually, reporting the average testing results across all 12 subsets in terms of average accuracy, AUC and the number of parameters (# Param). The results of different combinations of these four indicators over 12 datasets of the MEDMNIST benchmark, in terms of average accuracy and AUC, are shown in Table VIII.

TABLE VIII
RESULTS OF DIFFERENT COMBINATIONS OF INDICATORS TO CONSTRUCT THE OBJECTIVES BASED ON OUR OBJECTIVE CONSTRUCTION WAY ON MEDMNIST.

Model	AUC	ACC	# Param (MB)
fisher+synflow	0.922	0.833	5.46
jacob_cov+synflow	0.938	0.850	4.47
fisher+grad_norm	0.929	0.828	5.91
jacob_cov+grad_norm	0.930	0.841	4.20
fisher+jacob_cov	0.936	0.846	5.08
grad_norm+synflow	0.920	0.834	5.25

Table VIII demonstrates that the performances of combinations vary from 0.922 to 0.935 and 0.828 to 0.846 in terms of AUC and ACC, respectively. Specifically, *jacob_cov* combined with *synflow* and fisher have achieved very similar results which outperform other combinations. In addition, the *jacob_cov* indicator combined with any other indicators can achieve good performance when compared to other combinations. Furthermore, the performance of the architectures, as determined by various indicators, does not exhibit a clear relationship with the number of parameters.

3) Effectiveness of Objective Construction: To evaluate the effectiveness of our proposed objectives construction way, we compare our objectives with the following objectives construction way: 1) using the *synflow* and *jacob_cov* indicators as two objectives directly; 2) using FLOPs and *synflow* indicator as two objectives; 3) using FLOPs and *jacob_cov* indicator as two objectives; and 4) using FLOPs as one objective and (*synflow*+*jacob_cov*) as another one objective on MEDMNIST benchmark. It is worth noting that our method introduces the Flops to constrain the model's complexity. The results of different ways to construct the objectives are shown in Table IX. We report the accuracy, AUC score, FLOPs and the number of parameters of different ways to construct objectives. From Table IX, one can see that by adding FLOPs to different indicators, the final searched architectures' FLOPs can be well-constrained. Using multiple indicators in the objective at one time, the searched architectures can achieve better average performance over 12 subsets. Also, our proposed strategy of constructing the objectives can not only achieve higher average accuracy and AUC scores over 12 subsets of MEDMNIST benchmark but also have searched the model with fewer FLOPs. The higher ACC and AUC scores benefit from using multiple indicators to estimate the performance of candidate architectures. It has been verified that using our

TABLE IX
THE RESULTS OVER 12 DATASETS OF THE MEDMNIST BENCHMARK USING DIFFERENT WAYS TO COMBINE THE INDICATORS IN TERMS OF AVERAGE ACCURACY, AUC, FLOPS AND THE NUMBER OF PARAMETERS.

Model	AUC	ACC	FLOPs (M)	# Param (MB)
jacob_cov+*synflow* (w/o FLOPs)	0.924	0.834	24.21	5.24
FLOPs+*synflow*	0.921	0.831	14.45	4.59
FLOPs+*jacob_cov*	0.931	0.832	7.90	2.70
FLOPs+(*synflow*+*jacob_cov*)	0.931	0.835	11.21	3.53
MSTF-NAS (Ours)	0.938	0.850	14.62	4.47

TABLE X
AVERAGE ACCURACY AND AUC OVER 12 DATASETS OF THE MEDMNIST BENCHMARK USING DIFFERENT SEARCH SPACES.

Search space	AUC	ACC
DARTS	0.926	0.825
NSGA-NET	0.928	0.824
MSTF-NAS (Ours)	0.932	0.838

designed objectives to search the architecture is more efficient and effective. Furthermore, the number of parameters has not demonstrated any correlation with the model's performance that is consistent with the conclusion of Section IV-E2; however, it is directly proportional to the FLOPs.

4) Effectiveness of Search Space: To evaluate the effectiveness of our designed search space, we compare our search space with the other two search spaces, i.e. DARTS search space [17] and NSGA-NET search space [15]. DARTS search space is a very commonly used search space in most tasks in both natural image and medical image classification. To make a fair comparison among all the search spaces, we use a random search strategy to candidate 100 architectures, and then select the architecture that has the best estimate performance based on Eq. (5) from different search spaces. Then we build the neural network in the same way as our method. Finally, we train and test the performance of architectures on the MEDMNIST benchmark and report the average metrics over 12 subsets. Table X lists the average performance in terms of accuracy and AUC scores, which shows our search space outperforms other search spaces by 1.3% in terms of ACC and verifies the effectiveness of our search space.

F. Visualization of Searching Process

Our method is based on the evolutionary multi-objective algorithm, NSGA-II, to search for the optimal architecture candidates from a given search space. To intuitively show the search procedure, we visualize the Pareto-front from the initial generation to the last generation every ten generations. We normalize the values of the indicators and FLOPs into $[0, 1]$ over all generations. Pareto-front is made up of several Pareto-efficient solutions. The goal of the search procedure is to search the architectures that push the front into a place that contains more non-dominant solutions. The visualization of the Pareto-front of different generations is shown in Fig. 5, from which we can see that the Pareto-front of an initial generation almost scatters in a small region which means that all the initial architectures have similar performance in terms of the objectives. Moreover, from the first generation to the last generation, the Perato-fronts move from the diagonal line to a curve that non-dominates all the rest of the architectures in terms of two objectives with more and more scatter distribution. It means that the generation becomes more diverse and has higher objective values.

Besides, we also visualize the searched architecture from the PneumoniaMNIST dataset. The entire architecture is shown in Fig. 6. From the results, for the normal cell, we notice that the shallow normal cell has five operations with multiple kernel sizes (3×3, 5×5 and 7×7) and the deep normal cell has deeper connections when compared to the shallow one. For the reduction cell, we notice that each reduction cell contains different pooling operations with different pooling sizes. It means that the multi-scale pooling search space is effective in building the reduction cell. The high performance on both the MEDMNIST and CBIS-DDSM datasets indicates the effectiveness of our constructed multi-scale search space.

V. CONCLUSION

In this paper, we presented a novel method called multi-scale training-free neural architecture search (MSTF-NAS) for the customization of neural architecture in the context of medical image data. The MSTF-NAS approach efficiently searches for architectures by taking into account the multi-scale lesion region sizes present in medical images. To enable the search process, we divided the search space into a normal cell search space and a reduction search space, and we designed a new reduction search space specifically tailored for finding the optimal multi-scale architecture for a given dataset. Furthermore, we proposed a novel strategy to construct objectives that strike a balance between the capability and trainability of candidate architectures, enabling a multi-objective evolutionary search algorithm. To reduce search costs significantly, we incorporated a training-free proxy indicator, which accurately estimated the performance of candidate architectures. Finally, the effectiveness of our proposed method was validated through extensive evaluations on a large medical image benchmark and a sizable mammography dataset. The results demonstrated the capability of MSTF-NAS in customizing neural architectures for medical image analysis, showcasing its potential for enhancing the performance of medical image analysis systems.

REFERENCES

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

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of International Conference on Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[3] A. G. Roy, N. Navab, and C. Wachinger, "Concurrent spatial and channel 'squeeze & excitation' in fully convolutional networks," in *Medical Image Computing and Computer Assisted Intervention*. Cham: Springer International Publishing, 2018, pp. 421–429.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

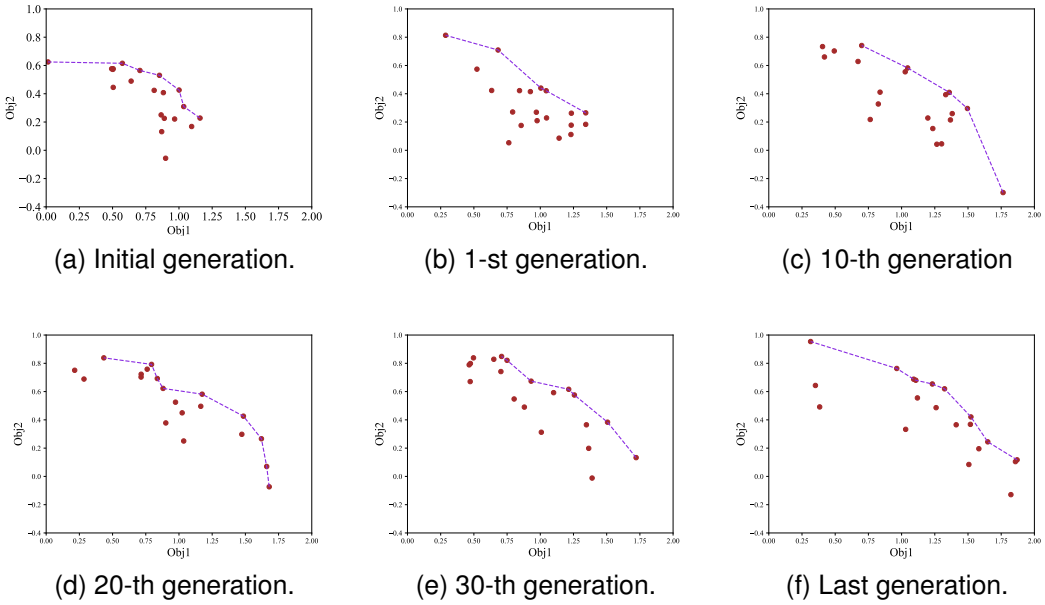

Fig. 5. Visualization of the Pareto-front of the evolutionary search procedure. The two objectives are constructed by our proposed objective construction method. The dotted line denotes the front of each generation.

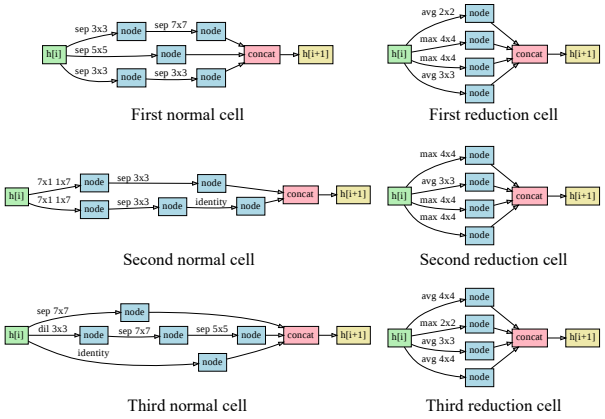

Fig. 6. Example architecture searched from the PneumoniaMNIST dataset.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015, pp. 234–241.

[7] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7262–7272.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

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

[10] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[11] Y. Wang, Z. Wang, Y. Feng, and L. Zhang, "WDCCNet: Weighted double-classifier constraint neural network for mammographic image classification," *IEEE Transactions on Medical Imaging*, vol. 41, no. 3, pp. 559–570, 2021.

[12] X. Shu, L. Zhang, Z. Wang, Q. Lv, and Z. Yi, "Deep neural networks with region-based pooling structures for mammographic image classification," *IEEE Transactions on Medical Imaging*, vol. 39, no. 6, pp. 2246–2255, 2020.

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

[14] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proceedings of the International Conference on Learning Representations*, 2017.

[15] 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.

[16] X. Xie, Y. Liu, Y. Sun, G. G. Yen, B. Xue, and M. Zhang, "BenchENAS: A benchmarking platform for evolutionary neural architecture search," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 6, pp. 1473–1485, 2022.

[17] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proceedings of the International Conference on Learning Representations*, 2019.

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

[19] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3640–3649.

[20] X. Fang and P. Yan, "Multi-organ segmentation over partially labeled datasets with multi-scale feature abstraction," *IEEE Transactions on Medical Imaging*, vol. 39, no. 11, pp. 3619–3629, 2020.

[21] S. Li, Y. Liu, X. Sui, C. Chen, G. Tjio, D. S. W. Ting, and R. S. M. Goh, "Multi-instance multi-scale CNN for medical image classification," in *Medical Image Computing and Computer Assisted Intervention*. Cham: Springer International Publishing, 2019, pp. 531–539.

[22] R. Su, D. Zhang, J. Liu, and C. Cheng, "MSU-Net: Multi-scale U-net for 2d medical image segmentation," *Frontiers in Genetics*, vol. 12, p. 639930, 2021.

[23] 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.

[24] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "NSGA-NetV2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *Proceedings of the European Conference on Computer Vision*. Springer, 2020, pp. 35–51.

[25] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Particle swarm optimization for compact neural architecture search for image classification," *IEEE Transactions on Evolutionary Computation*, 2022.

[26] Y. Sun, G. G. Yen, B. Xue, M. Zhang, and J. Lv, "Arctext: A unified

text approach to describing convolutional neural network architectures," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 4, pp. 526–540, 2021.

[27] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen *et al.*, "FBNetV2: Differentiable neural architecture search for spatial and channel dimensions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 965–12 974.

[28] Y. Yang, S. You, H. Li, F. Wang, C. Qian, and Z. Lin, "Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6667–6676.

[29] X. Chu, B. Zhang, H. Ma, R. Xu, and Q. Li, "Fast, accurate and lightweight super-resolution with neural architecture search," in *Proceedings of the International Conference on Pattern Recognition*. IEEE, 2021, pp. 59–64.

[30] X. Li, J. Zheng, M. Li, W. Ma, and Y. Hu, "One-shot neural architecture search for fault diagnosis using vibration signals," *Expert Systems with Applications*, vol. 190, p. 116027, 2022.

[31] 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.

[32] 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.

[33] C. Hsu, S. Chang, D. Juan, J. Pan, Y. Chen, W. Wei, and S. Chang, "MONAS: multi-objective neural architecture search using reinforcement learning," *CoRR*, vol. abs/1806.10332, 2018. [Online]. Available: http://arxiv.org/abs/1806.10332

[34] W. Li, S. Wen, K. Shi, Y. Yang, and T. Huang, "Neural architecture search with a lightweight transformer for text-to-image synthesis," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1567–1576, 2022.

[35] Y. Gou, B. Li, Z. Liu, S. Yang, and X. Peng, "CLEARER: Multi-scale neural architecture search for image restoration," in *Proceedings of the Advances in Neural Information Processing Systems*, 2020.

[36] Y. Hu, X. Jiang, X. Liu, B. Zhang, J. Han, X. Cao, and D. S. Doermann, "NAS-Count: Counting-by-Density with neural architecture search," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII*, ser. Lecture Notes in Computer Science, vol. 12367. Springer, 2020, pp. 747–766.

[37] W. Bae, S. Lee, Y. Lee, B. Park, M. Chung, and K.-H. Jung, "Resource optimized neural architecture search for 3d medical image segmentation," in *Medical Image Computing and Computer Assisted Intervention*. Springer, 2019, pp. 228–236.

[38] N. Cavagnero, L. Robbiano, B. Caputo, and G. Averta, "FreeREA: Training-Free evolution-based architecture search," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 1493–1502.

[39] T. Yang, L. Yang, X. Jin, and C. Chen, "Revisiting training-free nas metrics: An efficient training-based method," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 4751–4760.

[40] T. Do and N. H. Luong, "Training-free multi-objective evolutionary neural architecture search via neural tangent kernel and number of linear regions," in *Proceedings of International Conference on Neural Information Processing*, 2021, pp. 335–347.

[41] J. Zhang, L. Zhang, Y. Wang, J. Wang, X. Wei, and W. Liu, "An efficient multi-objective evolutionary zero-shot neural architecture search framework for image classification." *International Journal of Neural Systems*, pp. 2 350 016–2 350 016, 2023.

[42] R. Ru, C. Lyle, L. Schut, M. Fil, M. van der Wilk, and Y. Gal, "Speedy performance estimation for neural architecture search," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4079–4092, 2021.

[43] V. Lopes, S. Alirezazadeh, and L. A. Alexandre, "EPE-NAS: Efficient performance estimation without training for neural architecture search," in *Artificial Neural Networks and Machine Learning – ICANN 2021*. Cham: Springer International Publishing, 2021, pp. 552–563.

[44] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on imagenet in four GPU hours: A theoretically inspired perspective," *CoRR*, vol. abs/2102.11535, 2021. [Online]. Available: https://arxiv.org/abs/2102.11535

[45] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 7588–7598.

[46] N. Lee, T. Ajanthan, and P. H. S. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in *Proceedings of International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=B1VZqjAcYX

[47] C. Wang, G. Zhang, and R. B. Grosse, "Picking winning tickets before training by preserving gradient flow," in *Proceedings of International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkgsACVKPH

[48] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6377–6389, 2020.

[49] J. Turner, E. J. Crowley, M. F. P. O'Boyle, A. J. Storkey, and G. Gray, "BlockSwap: Fisher-guided block substitution for network compression on a budget," in *Proceedings of International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SklkDkSFPB

[50] Y. Balaji, M. Sajedi, N. M. Kalibhat, M. Ding, D. Stöger, M. Soltanolkotabi, and S. Feizi, "Understanding overparameterization in generative adversarial networks," *CoRR*, vol. abs/2104.05605, 2021. [Online]. Available: https://arxiv.org/abs/2104.05605

[51] 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.

[52] P. Murugan, S. Kannan, and S. Baskar, "NSGA-II algorithm for multi-objective generation expansion planning problem," *Electric Power Systems Research*, vol. 79, no. 4, pp. 622–628, 2009.

[53] J. Li, W. Zuo, E. Jiaqiang, Y. Zhang, Q. Li, K. Sun, K. Zhou, and G. Zhang, "Multi-objective optimization of mini U-channel cold plate with sio2 nanofluid by rsm and nsga-ii," *Energy*, vol. 242, p. 123039, 2022.

[54] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, "MedMNIST V2-A large-scale lightweight benchmark for 2d and 3d biomedical image classification," *Scientific Data*, vol. 10, no. 1, p. 41, 2023.

[55] R. S. Lee, F. Gimenez, A. Hoogi, K. K. Miyake, M. Gorovoy, and D. L. Rubin, "A curated mammography data set for use in computer-aided detection and diagnosis research," *Scientific Data*, vol. 4, no. 1, pp. 1–9, 2017.

[56] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, *Auto-sklearn: Efficient and robust automated machine learning*. Springer, 2019, pp. 113–134.

[57] 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*. ACM, 2019, pp. 1946–1956. [Online]. Available: https://doi.org/10.1145/3292500.3330648

[58] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," in *Proceedings of International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rylqooRqK7

[59] J. Zhang, D. Li, L. Wang, and L. Zhang, "One-shot neural architecture search by dynamically pruning supernet in hierarchical order," *International Journal of Neural Systems*, vol. 31, no. 07, p. 2150029, 2021.

[60] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proceedings of the International Conference on Learning Representations*, 2021.

[61] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your ViT? Data, augmentation, and regularization in vision transformers," *Transactions on Machine Learning Research*, vol. 2022, 2022.