IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 25, 2021, accepted November 9, 2021, date of publication November 11, 2021,
date of current version November 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3127437

Multi-Branch Neural Architecture Search for Lightweight Image Super-Resolution

JOON YOUNG AHN[1], (Member, IEEE), AND NAM IK CHO[2], (Senior Member, IEEE)
[1]Samsung Electronics Company Ltd., Suwon, Gyeonggi-do 18448, South Korea
[2]Department of Electrical and Computer Engineering, Institute of New Media and Communications (INMC), Seoul National University, Seoul 08826, South Korea

Corresponding author: Nam Ik Cho (nicho@snu.ac.kr)

This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program, supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP), under Grant IITP-2021-2016-0-00288; and in part by Samsung Electronics Company Ltd.

ABSTRACT Deep convolutional neural networks (CNNs) are widely used to improve the performance of image restoration tasks, including single-image super-resolution (SISR). Generally, researchers are manually designing more complex and deeper CNNs to further increase the given problems' performance. Instead of this hand-crafted CNN architecture design, neural architecture search (NAS) methods have been developed to find an optimal architecture for a given task automatically. For example, NAS-based SR methods find optimized network connections and operations by reinforcement learning (RL) or evolutionary algorithms (EA). These methods enable finding an optimal system automatically, but most of them need a very long search time. In this paper, we propose a new search method for the SISR that can significantly reduce the overall design time by applying a weight-sharing scheme. We also employ a multi-branch structure to enlarge the search space for capturing multi-scale features, resulting in better reconstruction on the textured region. Experiments show that the proposed method finds an optimal SISR network about twenty times faster than the existing methods, while showing comparable performance in terms of PSNR vs. parameters. Comparison of visual quality validates that the obtained SISR network reconstructs texture areas better than the previous methods because of the enlarged search space to find multi-scale features.

INDEX TERMS Single image super-resolution, neural architecture search, image restoration.

I. INTRODUCTION

Single image super-resolution (SISR) is a task that restores a high-resolution (HR) image from a single low-resolution (LR) observation. It is widely used as a preprocessing step of various tasks such as medical image analysis [1], security image processing [2], satellite image recognition [3], etc. Most of the recent researches adopt learning-based methods that use LR-HR image pairs for training [4]–[14], which generally show better performance than the classic interpolation-based [15] or reconstruction-based [16] methods.

Most earlier learning-based SR methods used single-branch neural networks for their simplicity and straightforwardness. However, when the single branch is deepened to increase the performance, there can be a gradient vanishing problem, and the resulting network needs too many parameters. Thus, instead of using the single branch network, some methods exploited multi-branch networks for extracting multi-scale features from the LR input [8], [14], [17], [18], thereby achieving better performances with fewer parameters. But due to the increased complexity of the network structure, it needs many trials and errors to find the optimal connection between the elements manually. Based on this, in this paper, we employ multi-branch architecture and propose an automated multi-branch SISR network design based on the neural architecture search (NAS) scheme [19], unlike the conventional manual design of single-branch or multi-branch networks. We include the multi-branch networks to expand the search space and propose a new NAS-based SISR network design while existing search methods attempted to find the optimal connection within the single-branch networks.

Neural architecture search (NAS) algorithm has been developed for the purpose of reducing the effort put into designing the neural architecture of certain tasks [19]–[27]. They focus on the image classification task and try to find promising network automatically by adopting reinforcement learning, evolutionary algorithm or gradient descent method.

The associate editor coordinating the review of this manuscript and approving it for publication was Peng Liu.

Recently, researchers have expanded the NAS to other tasks such as image restoration (MoreMNAS [28], FALSR [29], HNAS [30], Improved DARTS [31], DLSR [32]), object detection [33], and other dense predictions such as segmentation, pose estimation and 3D detection by encoding multi-scale image contexts in the search space [34]. For the SISR, FALSR and MoreMNAS used a reinforced evolution algorithm and solved the image SR task as a multi-objective problem. However, the reinforced evolution method took a tremendous amount of time to derive an optimal network. Additionally, FALSR did not use a complete training scheme, but they measured the performance of the network approximately. DLSR extended the differentiable NAS [25] and its improved version MiLeNas [35] for the SISR and achieved state-of-the-art performance while requiring about ten times less design time than the FLASR based on the reinforced evolution method.

In this paper, we adopt the weight-sharing scheme of ENAS [21] as our baseline search algorithm because it is known to provide faster design time than its predecesors. As in the original ENAS for the classification problems, we configure a controller and a child network in the search process. The controller generates a sequence for a child network, and a child network is constructed by the generated controller sequence. REINFORCE algorithm is used to train the controller network to generate a better child network. For the SISR task, The reward signal in REINFORCE is the PSNR between the generated child network's output and the ground-truth. We share the parameters of each child network during the search phase. In addition, we propose a complexity-based penalty to reduce the reward from the network that needs a huge parameter. By applying the complexity-based penalty, the controller tends to recommend powerful but lightweight networks.

Image super-resolution is a kind of regression task that generally requires a more precise and complex network than a classification task. For this reason, we search for a new SR architecture on a multi-branch search space as stated above. To be specific, we develop a Multi-Branch Neural Architecture Search (MBNAS) algorithm, which tries to find optimal connections of multi-scale features. The MBNAS search space consists of partially shared nodes (PSN) for multi-scale block, local feature fusion layer, and global feature fusion layer. The PSNs share their parameters with different network branches to transmit information efficiently with fewer parameters. For simplicity, we use only 3×3 convolution and 3×3 dilated convolutions [36] as basic building blocks, and let the search algorithm find optimal connections. Still, we obtain an efficient architecture as a result of the search algorithm, which is validated by extensive experiments. The experimental results show that our network obtained by the MBNAS, named as MBNASNet, performs comparably to human-crafted networks and the existing NAS-based SR networks [28]–[30].

Our main contributions are summarized as follows:

1) New NAS-based SR: We propose a new NAS-based SR network design method, named MBNAS, which searches for networks with higher performance by combining multi-scale information efficiently. The resulting SR network is the MBNASNet.
2) Complexity-based penalty: We propose a complexity-based penalty and add it to the reward signal of the REINFORCE algorithm. This enables us to search for an efficient network that has high performance with a lightweight structure.
3) Multi-scale feature extraction: We construct the network with a multi-branch structure, which has been used in existing lightweight SR network design [8], [14], [17], [18].
4) Partially shared node (PSN): We partially share the parameters of branches to connect each other's information and construct a lightweight structure. The partially shared structure efficiently reduces the searched network's parameter without performance degradation.

We presented a preliminary work of NAS-based image super-resolution with a single-branch network in [37], called DeCoNASNet. The major difference of this work from our previous version is that we propose an expanded search space for NAS to capture multi-scale information, which brings a significant performance gain with reduced parameters. For this, we modify the algorithm to include the multi-braches into the search space. Also, we provide detailed analysis and explanations of the search process and results, and exhibit more experimental results, including the results on higher rate SR.

The rest of this paper is organized as follows. Section 2 summarizes related works on the single image super-resolution and neural architecture search methods. In section 3, We explain our proposed search method for SISR. Section 4 includes the details about our implementation settings and dataset configurations, followed by experiment results. We discuss our main contributions and conduct ablation experiments in section 5. Finally, We provide a summary and concluding remarks in section 6.

II. RELATED WORK

A. SINGLE IMAGE SUPER-RESOLUTION

A number of methods have been proposed for learning the mapping function from LR images to the appropriate HR counterparts [4]–[14]. Dong *et al.* proposed SRCNN [4], which is the first deep learning structure for the SISR. It used three layers of convolutional neural networks (CNNs) and outperformed non-learning-based conventional methods by a large margin. FRCNN [5] and ESPCN [6] used specific structures to reduce the computational cost of deep neural networks in the SISR networks. They proposed deconvolution layers and sub-pixel convolution layers to upsample LR features to an HR image. VDSR [7] used residual learning and gradient clipping strategy to increase

the depth and thus the performance. Lim *et al.* [8] introduced residual blocks with extensive features (EDSR) and multi-scale structure (MDSR) to improve the performance further. MemNet [10], MSRN [14], and DenseSR [9] proposed memory block, multi-scale residual block, and dense block, respectively, for a better SR restoration. SelNet [38] improved the performance by replacing the Relu operation with the selection unit. Zhang *et al.* proposed residual dense block and dense feature fusion algorithm in RDN [11] to extract abundant information from the input image. RCAN [39] proposed a channel attention scheme that improved the representational ability of the neural network.

B. NEURAL ARCHITECTURE SEARCH

In designing a deep network, we should select a considerable number of network configurations such as connection, operation type, the number of feature channels, depth, etc. Researchers have designed their structures through a large number of trials to achieve a competent performance. However, it is a tedious task and difficult to find an optimal system for a given task. The NAS algorithms have been proposed to alleviate this burden, especially in the case of image classification researches [19]–[26].

As the first study of NAS, Zoph *et al.* [19] proposed a reinforcement learning (RL) based algorithm. They configured a controller network to generate a child network and trained it by REINFORCE [40], which is a kind of policy gradient algorithm. The performance of the child network was used as a reward signal of the controller network, where the child network was trained from scratch. Therefore, it took a huge amount of time to get a reward signal from the child network. To reduce the time to measure the performance, PNAS by Liu *et al.* [20] used the sequential model-based optimization (SMBO) with a surrogate model which predicts its performance instantly. On the other hand, Pham *et al.* [21] proposed ENAS that constructs a weight sharing child network to reduce the reward calculation time. This method configured a large graph and regarded each child network as a sub-graph. The parameters of the child network were shared in the search phase by storing their weights in the main graph.

Evolutionary methods [22]–[24] are another trend of the NAS algorithm. They pick a population of architectures randomly at first and then encode these networks as binary codes. Genetic modifications such as crossover or mutations are applied to the sequence, suggesting a better structure. Lu *et al.* [23] proposed another method that takes advantage of search history by using a Bayesian optimization algorithm. AmoebaNet [24] applies an aging evolution method to NAS to discard the earliest trained network.

DARTS [25], SGAS [41], NAO [26] and CSA-NAS [27] proposed different approaches from RL and evolutionary methods. Specifically, DARTS applies continuous relaxation to the neural architecture's connections for optimizing the connections and parameters simultaneously. SGAS applies a greedy operation selection method to the DARTS and obtains the best architecture without retraining. NAO projects the encoded sequence to the learnable embedding space of structures and recommends the best architecture as a result. CSA-NAS adopts a binary crow search algorithm to find the optimal architecture. More recently, HR-NAS [34] was proposed to exploit multiscale features by adopting a multi-branch architecture. As a result, they could effectively learn high-resolution representations and showed improved performance in several dense prediction tasks, as well as in image classification.

Regarding the search space design, neural architecture search methods can be categorized into two groups: methods dealing with (1) flat search space or (2) cell-based search space. The methods with flat search space [19], [21]–[23] aim to find the optimal setting for the number of channels (width), number of layers (depth), types of operations (convolution or max pooling) for the whole structure, while cell-based algorithms [20], [21], [23]–[26] try to find a structure of the cell before stacking them to form the final architecture. The cell-based search space design is inspired by the split-transform-merge strategy used in Inception block [42], hence it can approximate the optimal solution for a given task.

Unlike the above algorithms, CSNAS [43], UnNAS [44], and SSNAS [45] discard supervised settings which suffer from the high cost of data labeling. CSNAS and SSNAS adopt a self-supervised setting, and UnNAS applies unsupervised learning to search for promising architectures with unlabeled data. Recently, researchers are also trying to overcome the reproduction challenge and fairly compare search methods by proposing benchmarks for the NAS and providing some important principles for scientific research in the community [46]–[48].

There have been many NAS methods as stated above, among which we choose ENAS as our SR design baseline for its fast design time and also for including the network complexity in the design constraints. Regarding the design time, DARTS [25], FBNet [49], and FBNetV2 [50] also provide fast design time for practical use. But, we choose ENAS as our SR design baseline because we can easily include the complexity constraint into consideration within the ENAS framework. Specifically, as the ENAS is based on the REINFORCE, we modify the reward signal of the REINFORCE to consider the network complexity as well as the SR performance.

C. IMAGE SUPER-RESOLUTION WITH NEURAL ARCHITECTURE SEARCH

Some researchers recently adopted NAS methods to design image super-resolution CNNs [28]–[30], [32]. MoreMNAS [28] adopted multi-objective genetic algorithm NSGA-II [51] for the model generation and proposed a reinforced mutation method. FALSR [29] used a hybrid controller instead of a reinforced controller and proposed an elastic search space for macro and micro search. The search

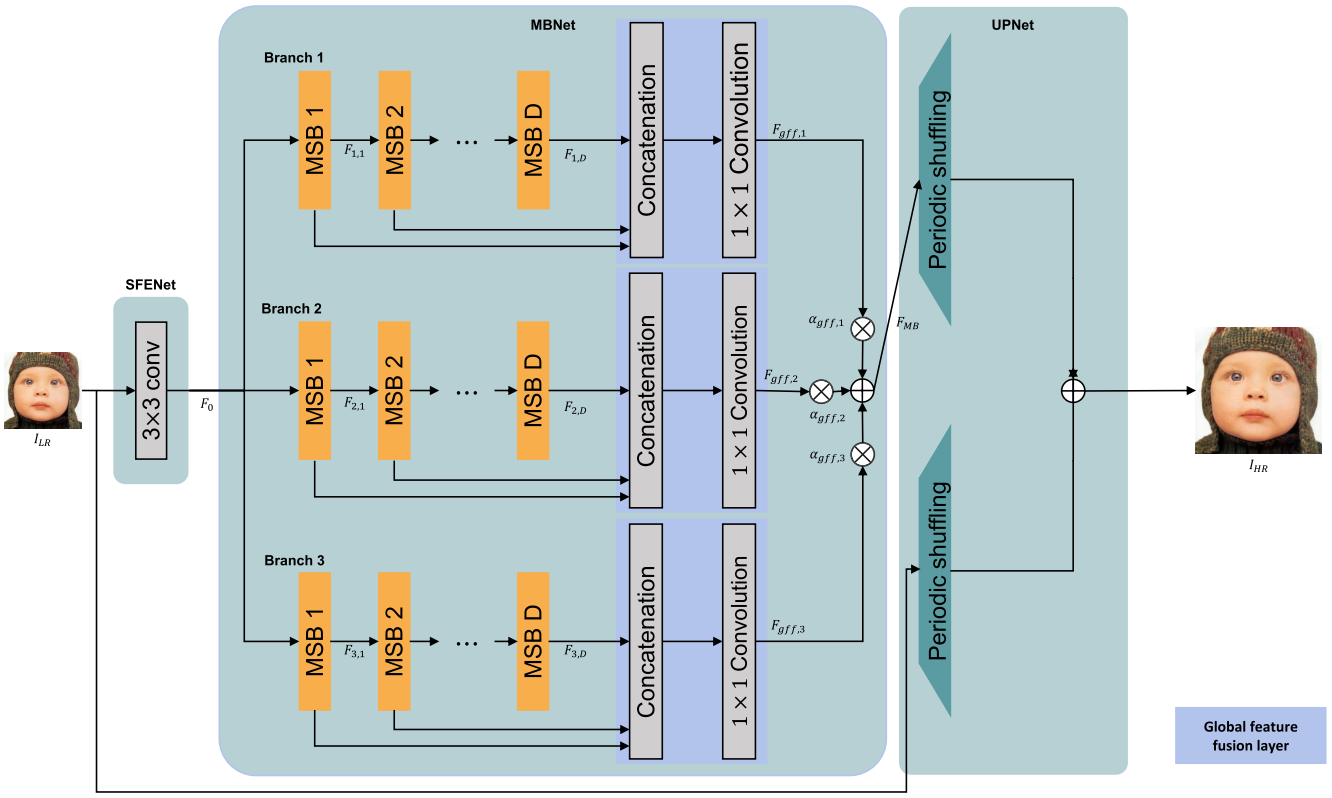

FIGURE 1. The overall structure of MBNASNet. It consists of a shallow feature extraction network (SFENet), a multi-branch network (MBNet), and an upscaling network (UPNet). The result of each branch is combined and upsampled by the periodic shuffling layer. The MSB (multi-scale block) is a basic building block, detailed in Fig. 2(a).

space complexity of both methods is 9.6×10^{15}. HNAS [30] adopted a hierarchical search algorithm with reinforcement learning to simultaneously find promising cell structure and upsampling layer positions. They also considered the computational cost (FLOPS) to meet the requirements about resources constraint. More recently, DLSR [32] adopted DARTS for SR network search, which is shown to require less design time than the preceding design methods. They also showed that SISR models could be searched on both the cell-level and network-level by their method and reported the state-of-the-art models.

Regarding the architecture and the search space thereof, these previous NAS-based methods prepare basic building blocks, which consist of convolutional layers, ReLu, etc., in cascade. Then, they let the NAS algorithm determine the number of layers and connections inside the cells. Meanwhile, we prepare a sophisticated architecture to have expanded search space, *i.e.*, a structure with more different functional elements to connect. Specifically, we prepare several branches of building blocks, consisting of multi-rate dilated convolutions, ReLu, and attention, and let the NAS algorithm find the connections among the various-scale convolutions. By expanding search space through the multi-branch of dilated convolutions, we can exploit multi-scale features for better SR reconstruction than conventional single-branch architecture.

III. METHOD

A. OVERVIEW OF THE PROPOSED MBNAS

Our MBNASNet (a child network) is shown in Fig. 1, whose components (MSBs) are designed by a controller in Fig. 2, according to the MBNAS algorithm of Fig. 3. The automated design cycle in Fig. 3 illustrates that the controller is trained to generate a potent network, and the child network is trained to get the performance, which is used to calculate the reward signal.

Fig. 1 shows the overview of MBNASNet, which consists of a shallow feature extraction network (SFENet), an upscaling network (UPNet), and a multi-branch network (MBNet). The MBNet is designed by the NAS, which consists of several branches. The MSB (multi-scale block) in the figure is the basic building block detailed in Fig. 2. We extract a shallow feature by the SFENet that is fed to each branch. The partially shared parameters in each branch extract the multi-scale features with different receptive fields. Results from each branch are combined and upsampled by pixelshuffle layers [6] to create HR residual information. Finally, the residual information is added to the upsampled LR input to make the final HR result.

Fig. 2 shows the details of MSB and illustrates their internal connections according to sequences from the controller. In each of Fig. 2(a) and (b), the upper part shows a branch of MBNASNet in Fig. 1, where three consecutive MSBs are shown. The central part details the structure of

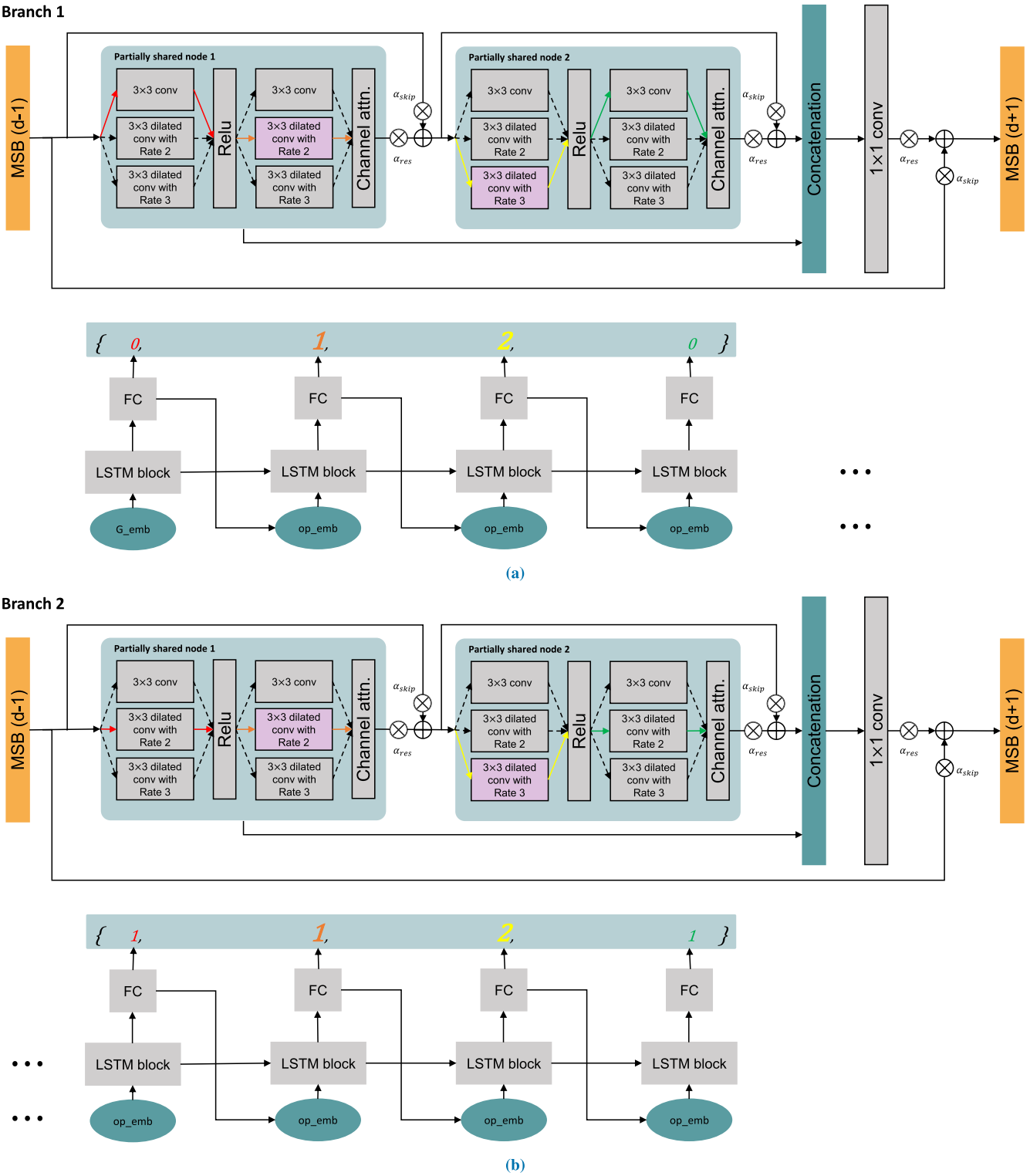

FIGURE 2. The upper part of (a) shows details of our MSB, and the lower part is illustrating that a controller determines the connections inside the MSBs of branch 1 according to the controller sequence (outputs of FC layers), with an example that there are two partially shared nodes (PSNs) ($M = 2$) and two branches ($B = 2$). (b) shows the example for branch 2, where the elements inside the MSB are differently connected than the above case according to the corresponding controller. Two branches share the parameters of the light purple box. The dashed arrows and colored arrows mean that these connections are to be searched.

the d-th MSB, and the left and right are the $(d-1)$-th and $(d+1)$-th MSBs. The lower part shows the controller that outputs a sequence to determine the internal connections of the MSB. Fig. 2(a) and (b) show different examples of the output sequences from the controller and the corresponding connections inside the MSBs.

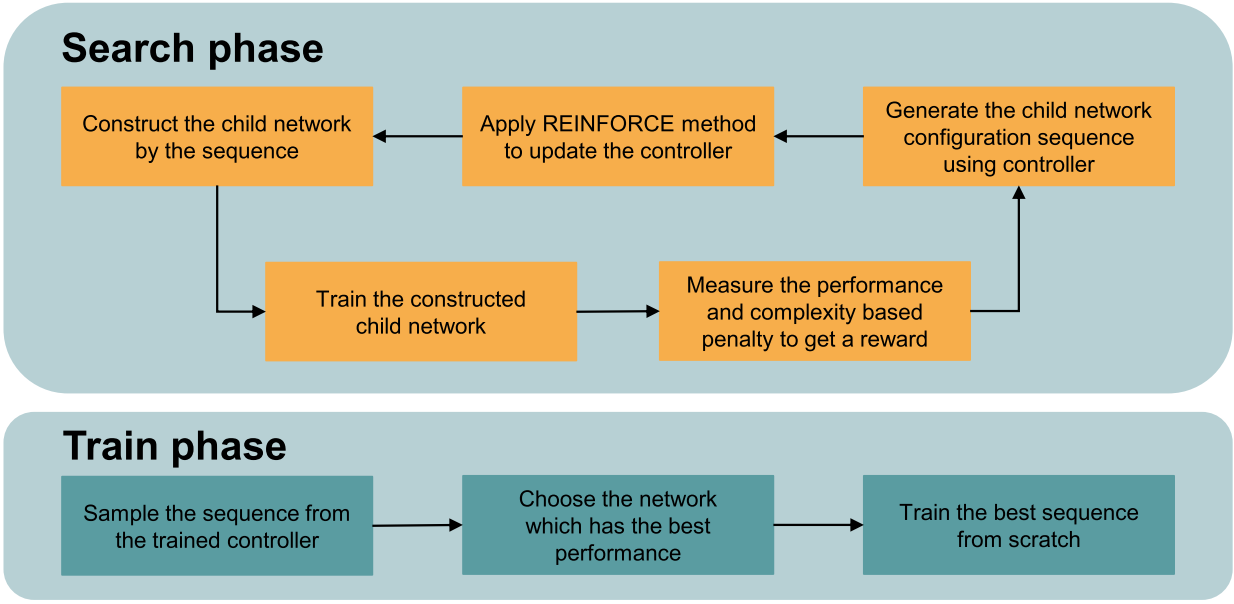

FIGURE 3. The overview of the search cycle and training. In the search phase, the controller and constructed child network are trained alternatively. In the training phase, the searched final architecture is trained from scratch.

We use Long Short Term Memory (LSTM) [52] to create the controller, where the parameters are updated by REINFORCE algorithm. While conventional RL methods calculate the reward signal of REINFORCE as the performance of validation sets, we consider both performance and network complexity. For this, we design a complexity-based penalty and add it to the reward signal to find a more efficient architecture. The details of the controller, MBNASNet, and design procedure are explained in the rest of this Section.

B. CONTROLLER AND COMPLEXITY-BASED PENALTY

1) CONTROLLER CONFIGURATION

We use a two-layer LSTM as our controller as shown in the lower part of Fig. 2. It generates a sequence for creating a child network at the end of the fully connected layer (FC). The output sequence $\mathbf{S}_c$ for a child network c is defined as

$$\begin{aligned} \mathbf{S}_c &= \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_b, \ldots, \mathbf{s}_B\}, \\ \mathbf{s}_b &= \{(s_b)_{m,n}\}, 0 < m \leq M,\ 0 \leq n < N, \end{aligned} \tag{1}$$

in the case that the child network consists of B branches, M PSNs in one multi-scale block (MSB), and each node has N layers. $\mathbf{S}_c$ consist of B sequences, and each $\mathbf{s}_b$ denotes the sequence of the b-th branch structure. We need N sequences to create the m-th PSN for one branch. As a result, our controller consists of $M \times N \times B$ LSTM blocks, where each block is followed by an FC layer. The FC layer has K outputs, where K is the candidate operations of our network. The example sequence and the constructed block architecture are shown in Fig. 2, which generate eight outputs for a two-branch structure ($B = 2$) with two PSNs ($M = 2$) that have two layers ($N = 2$).

In our search space, the total number of possible directed acyclic graphs (DAGs) is $|K|^{B\times M\times N}$. The set of all possible neural architecture is enormously expanded by a factor of $|K|^{M\times N}$ when increasing the number of branches. The search space is also expanded if we increase the number of PSNs or their layers. Hence, to limit the number of possible architectures to a manageable size, we choose $B = 3$, $M = 2$ and $N = 2$ in our MBNASNet. Because we have three candidate operations ($|K| = 3$), as will be addressed in Sec. III-D, the possible set of the architecture is 5.3×10^5. Finally, to ensure that the number of parameters is less than 2M, we construct our MBNASNet with four multi-scale blocks ($D = 4$).

2) COMPLEXITY-BASED PENALTY

The REINFORCE algorithm uses a reward signal to train the parameters of the controller. While ENAS uses only a task performance as the reward signal, we modify the reward signal to find a more powerful and lightweight architecture, as stated in overview section. Specifically, we propose a complexity-based penalty to penalize a structure with large parameters, and define a reward signal R as

$$R = p(c; \mathbf{w}) - \lambda \times cb(c), \tag{2}$$

where $p(c; \mathbf{w})$ is the PSNR of model c and $\mathbf{w}$ is the parameters of a child network. The complexity-based penalty, $cb(c)$ is defined as

$$cb(c) = \frac{n_c}{n_{max}}, \tag{3}$$

where n_{max} denotes the number of the model's parameters, which uses all candidates in the search space, and n_c is the number of parameters of the designed child network. To set

a trade-off between the parameters and the performance, we multiply λ to the complexity-based penalty.

C. MBNASNet

As shown in Fig. 1, we first extract a shallow feature F_0 from an input low-resolution image (I_{LR}) by the SFENet (3×3 convolution layers). The F_0 is then fed to the first MSB of each branch. Formally, the F_0 is expressed as

$$F_0 = H_3(I_{LR}) \tag{4}$$

where $H_3(\cdot)$ denotes the 3×3 convolution operation.

The MBNet is constructed to have B branches, where each branch is a cascade of MSBs followed by their outputs' concatenation and 1×1 convolution to make a feature map. The searched MSBs in each branch have different receptive fields, and thus each branch learns multi-scale characteristics for image super-resolution. We multiply an independent scalar weight to the outputs of each node and block to adjust the gradient magnitude in back-propagation. A similar technique was used in [14]. We name these weights as gradient flow control weights and denote them as α, as illustrated in the last part of the MBNet block in Fig. 1.

Formally, the output of the d-th MSB in the b-th branch, $F_{b,d}$ is

$$\begin{aligned} F_{b,d} &= (\alpha_{skip})_{b,d} \times F_{b,d-1} \\ &\quad + (\alpha_{res})_{b,d} \times H_1(concat(F_{b,d,1}, \ldots, F_{b,d,M})), \end{aligned} \tag{5}$$

where $F_{b,d,m}$ denotes the output of the m-th PSN of the d-th multi-scale block (MSB) in the b-th branch, and $H_1(\cdot)$ denotes the 1×1 convolution operation for the local feature fusion layer. Also, α_{skip} and α_{res} are the gradient flow control weights for residual feature and skip connection, respectively. $F_{b,d,m}$ will be detailed in the following subsection, with Fig. 2 and Eq. 9.

Then, the output of the MBNet is a weighted sum of all the branch outputs:

$$F_{MB} = \sum_{b=1}^{B} (\alpha_{gff})_b \times (F_{gff})_b \tag{6}$$

where

$$(F_{gff})_b = H_1(concat(F_{b,1}, \ldots, F_{b,D})), \tag{7}$$

and $(\alpha_{gff})_b$ is a gradient flow control weights for global feature fusion layer. Also, $(F_{gff})_b$ is the output of global feature fusion layer of the b-th branch.

Finally, we obtain the reconstructed high-resolution image I_{HR} by combining the up-sampled low-resolution image I_{LR} and residual information in the UPNet F_{MB}. Formally, the I_{HR} is computed as

$$I_{HR} = H_{ps}(I_{LR}) + H_{ps}(F_{MB}), \tag{8}$$

where $H_{ps}(\cdot)$ denotes 3×3 convolution and periodic shuffling layer as in ESPCN [6]. We fix the structure of SFENet and UPNet while searching the connection of MBNet.

D. MULTI-SCALE BLOCK WITH PARTIALLY SHARED NODES

We apply a cell structure for the MSB, which means that all MSBs in the same branch have the same connection and operation. Each MSB consists of M PSNs as shown in the upper part of Fig. 2 (a) and (b). The dashed arrows and colored arrows in Fig. 2 mean that these connections are to be searched. The candidate operations of the PSN are

1) 3×3 convolution,
2) 3×3 dilated convolution with rate two,
3) 3×3 dilated convolution with rate three.

Following the signal flow in Fig. 2, $F_{b,d,m}$ in Eq. 5 is calculated as

$$\begin{aligned} F_{b,d,m} &= (\alpha_{skip})_{b,d,m} \times F_{b,d,m-1} \\ &\quad + (\alpha_{res})_{b,d,m} \times (H_b)_{PSN,m}(F_{b,d,m-1}), \end{aligned} \tag{9}$$

where $(H_b)_{PSN,m}(\cdot)$ denotes the operation of the m-th PSN in the b-th branch. The $(H_b)_{PSN,m}(\cdot)$ can be expressed as

$$(H_b)_{PSN}(\cdot) = CA(H_{(s_b)_{m,2}}(Relu(H_{(s_b)_{m,1}}(\cdot)))), \tag{10}$$

where $H_{(s_b)_{m,n}}(\cdot)$ denotes the k-th operation among K candidates, which is chosen by the configuration sequence $(s_b)_{m,n}$. We construct the PSN with two operations and one Relu activation as shown in Eq. 10. $CA(\cdot)$ denotes channel attention layer of RCAN [39].

To reduce the number of network parameters and spread the information through the branches, the parameters of PSNs have common weights if the configuration sequence of different branches activates an identical position in their sequence. For example, if two branches' configuration sequences are '001' and '011,' the operation corresponding to the first and the third digit share their weights. In Fig. 2, we emphasize the shared positions in the controller sequence (FC outputs) by big bold digits.

E. MBNAS

Like conventional RL-based NAS methods [19], [21], our algorithm has $\boldsymbol{\theta}$ and $\mathbf{w}$, which represents the parameter of the controller and the child network, respectively. In the search phase, $\boldsymbol{\theta}$ and $\mathbf{w}$ are trained alternately for each epoch. After the search phase is finished, we sample the sequences by the trained controller. Then, the best sequence among the sampled ones is chosen and trained from scratch.

1) TRAINING THE CHILD NETWORK

We first train the parameters of a child network to calculate the reward signal of the controller. The problem is formulated as

$$\min_{\mathbf{w}} \mathbb{E}_{c \sim \pi(c;\boldsymbol{\theta})}[L(c; \mathbf{w})], \tag{11}$$

where $L(\cdot)$ denotes the loss function for the task which is the $L1$ loss in our setting. The controller's policy $\pi(c; \boldsymbol{\theta})$ is fixed when training the child network. The Adam optimizer [53] is used to optimize $\mathbf{w}$. We estimate the gradient

of $\mathbb{E}_{c\sim\pi(c;\boldsymbol{\theta})}[L(c;\mathbf{w})]$ with the Monte Carlo estimate

$$\nabla_{\mathbf{w}}\mathbb{E}_{c\sim\pi(c;\boldsymbol{\theta})}[L(c;\mathbf{w})] \approx \frac{1}{M}\sum_{i=1}^{M}\nabla_{\mathbf{w}}L(c_i;\mathbf{w}), \tag{12}$$

where c_i denotes a sampled child network by the controller's policy. We choose $M = 1$, which means that we sample just one child network for each mini batch.

2) TRAINING THE CONTROLLER

In the controller training phase, $\mathbf{w}$ is fixed, and $\boldsymbol{\theta}$ is trained by REINFORCE [40] algorithm. We optimize $\boldsymbol{\theta}$ to maximize the expectation of reward signal, which can be expressed as

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{P(a_{1:T};\boldsymbol{\theta})}[R], \tag{13}$$

where $a_{1:T}$ is the configuration sequence for the child network c. In the REINFORCE, the gradient of the expected reward is approximated as

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{P(a_{1:T};\boldsymbol{\theta})}[R] = \sum_{t=1}^{T}[\nabla_{\boldsymbol{\theta}}\log P(a_t|a_{t-1:1};\boldsymbol{\theta})(R-b)] \tag{14}$$

where b is the baseline which is used to reduce the variance. The moving average of the reward signal is used for the baseline in our algorithm. As explained with Eq. 2, we use the PSNR of validation set and complexity-based penalty to calculate reward signal. Adam [53] is used to optimize the reward.

IV. DATASETS AND EXPERIMENTS

A. SETTINGS

1) DATASETS, DEGRADATION METHODS, AND METRICS

We choose DIV2K [54] dataset for the training and validation. The DIV2K dataset is widely used as a training set of various image restoration tasks. It contains 1,000 images, consisted of 800 for training, 100 for validation, and the other 100 images for test. The validation images are used as the data for measuring reward signal of controller network.

We measure the performance on four different benchmark dataset; Set5 [55], Set14 [56], BSDS100 [57], and Urban100 [58]. To compare the performances with others, we measure the PSNR and SSIM [59] of the test image on the Y channel of YCbCr color domain. We create the synthetic low-resolution image by applying Matlab's imresize function [60].

2) IMPLEMENATATION DETAILS

We construct the controller by a two-stacked LSTM network with 64 hidden states. We connect three fully connected layers to the end of each LSTM block to get the configure sequence for the child network. We use word embedding [61] to make the input of the LSTM layer from the previous LSTM block's output.

We construct our MBNASNet with three branches ($B = 3$), four multi-branch blocks ($D = 4$), and two PSNs ($M = 2$) which have three operations as the candidate operations.

TABLE 1. Mean and variance of searched networks from three controllers which are trained from different random seeds.

Experiment	mean PSNR	mean CBP	var. PSNR
1	34.134	0.561	0.00143
2	34.131	0.543	0.00096
3	34.130	0.554	0.00097

The number of output feature maps for SFENet and MSNet is unified to 32. The number of intermediate features in PSNs is 128, which is four times bigger than the number of the output feature maps.

3) HYPER-PARAMETER SETTINGS

In the search phase, we alternatively train the controller and child network for one epoch each. We initialize both the controller parameter $\boldsymbol{\theta}$ and the child network parameter $\mathbf{w}$ by using the variance scaled initialization [62] with 0.02 scaling value. We train the controller and the child network for 500 epochs. For one epoch, we apply 100 iterations for the controller, and 1,000 iterations for the child network. The learning rate of the controller is fixed to 3×10^{-4}. The learning rate of the child network initialized to 3×10^{-4} and decreased by half for every 100 epochs. We use 16 low-resolution image patches of size 64×64 from DIV2K train images as a mini-batch of the child network. We augment the patches by randomly applying horizontal flip and 90°, 180°, 270° rotation. The λ in Eq. 2 is set to 2, and $p(c;\mathbf{w})$ is the validation PSNR of child network. We randomly extract 1,000 low-resolution image patches from DIV2K validation images and compute PSNR to calculate the reward.

In the training phase, we sample 500 configuration sequences from the trained controller network and choose the architecture which has the best performance in the DIV2K validation set as our MBNASNet. We train the selected network for 1,000 epochs and finetune the trained network for 1,000 more epochs. The hyper-parameter settings are the same as the search phase except for the learning rate. The learning rate of the child network is initialized to 3×10^{-4} and decreased half by 200 epochs.

B. EXPERIMENTS ON SINGLE IMAGE SUPER-RESOLUTION (SISR)

1) MBNAS SEARCH RESEULT

The proposed MBNASNet has four multi-scale blocks ($D = 4$) and two PSNs ($M = 2$) with three branches ($B = 3$). We sample 500 architectures and choose the best architecture from them. For $\times 2$ scale, the configuration sequence of each branch is found to be

$$\begin{aligned}\mathbf{s}_1 &= \{0, 0, 1, 2\},\\ \mathbf{s}_2 &= \{0, 0, 1, 2\},\\ \mathbf{s}_3 &= \{2, 0, 1, 2\}.\end{aligned} \tag{15}$$

We note that our searched structure has two same blocks with different channel attention and one block with

TABLE 2. PSNR and SSIM on benchmark datasets (Set5, Set14, B100, and Urban100) for ×2 and ×3 SR tasks. We emphasize the best and the second-best performances with the red and blue colors, respectively. Methods with bold characters are NAS-based methods, and the "Design time" at the last column indicates the times taken for the search process. All four indicated design times are calculated with the same GPU (NVIDIA Tesla V100). Other NAS-based methods do not report more than ×3 SR results due to huge search times, whereas we could. *In the case of the HNAS, the complexity is an estimated one because they do not explicitly reveal the number of parameters. Also, the + sign at the HNAS denotes that they used self-ensemble, which generally gives higher PSNR than the baseline.

Model	scale	Params	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM	Design time
Bicubic	×2	-	33.66 / 0.9299	30.24 / 0.8688	29.56 / 0.8431	26.88 / 0.8403	-
SRCNN [4]		57K	36.66 / 0.9542	32.45 / 0.9067	31.36 / 0.8879	29.50 / 0.8946	-
VDSR [7]		665K	37.53 / 0.9587	33.03 / 0.9124	31.90 / 0.8960	30.76 / 0.9140	-
LapSRN [13]		813K	37.52 / 0.9591	33.08 / 0.9130	31.80 / 0.8950	30.41 / 0.9101	-
MemNet [10]		677K	37.78 / 0.9597	33.28 / 0.9142	32.08 / 0.8978	31.31 / 0.9195	-
MSAN-X [17]		870K	37.86 / 0.8909	33.52 / 0.9167	32.12 / 0.8983	31.91 / 0.9255	-
SelNet [38]		970K	37.89 / 0.9598	33.61 / 0.9160	32.08 / 0.8984	- / -	-
CARN [12]		1,582K	37.76 / 0.9590	33.52 / 0.9166	32.09 / 0.8978	31.92 / 0.9256	-
A^2F-M [63]		1,000K	38.04 / 0.9607	33.67 / 0.9184	32.18 / 0.8996	32.27 / 0.9294	-
***HNAS-C+** [30]		∼400K	38.11 / 0.964	33.60 / 0.920	32.17 / 0.902	31.93 / 0.928	-
MoreMNAS-A [28]		1,039K	37.63 / 0.9584	33.23 / 0.9138	31.95 / 0.8961	31.24 / 0.9187	56 days
FALSR-A [29]		1,021K	37.82 / 0.9595	33.55 / 0.9168	32.12 / 0.8987	31.93 / 0.9256	24 days
DeCoNASNet [37]		1,713K	37.96 / 0.9594	33.63 / 0.9175	32.15 / 0.8986	32.03 / 0.9265	12 hours
MBNASNet(ours)		999K	38.04 / 0.9595	33.70 / 0.9178	32.19 / 0.8992	32.17 / 0.9281	24 hours
Bicubic	×3	-	30.39 / 0.8682	27.55 / 0.7742	27.21 / 0.7385	24.46 / 0.7349	-
SRCNN [4]		57K	32.75 / 0.9090	29.30 / 0.8215	28.41 / 0.7863	26.24 / 0.7989	-
VDSR [7]		665K	33.66 / 0.9213	29.77 / 0.8314	28.82 / 0.7976	27.14 / 0.8279	-
MemNet [10]		677K	34.09 / 0.9248	30.00 / 0.8350	28.96 / 0.8001	27.56 / 0.8376	-
MSAN-X [17]		1,054K	34.19 / 0.9246	30.27 / 0.8403	29.03 / 0.8030	27.96 / 0.8473	-
SelNet [38]		1,159K	34.27 / 0.9257	30.30 / 0.8399	28.97 / 0.8025	- / -	-
CARN [12]		1,582K	34.29 / 0.9255	30.29 / 0.8407	29.06 / 0.8034	28.06 / 0.8493	-
A^2F-M [63]		1,003K	34.50 / 0.9278	30.39 / 0.8427	29.11 / 0.8054	28.28 / 0.8546	-
MBNASNet(ours)		1,003K	34.30 / 0.9255	30.25 / 0.8415	29.08 / 0.8042	28.08 / 0.8501	30 hours

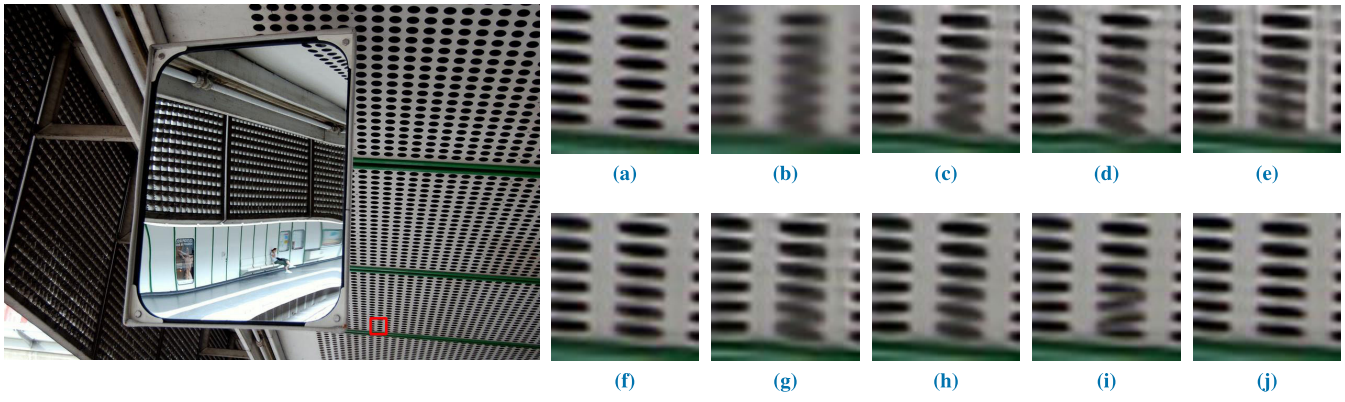

FIGURE 4. Qualitative result on the 4th image from the Urban100 dataset for ×2 SR task. We compare our method with nine conventional SR methods. (a) ground truth. (b) bicubic downsampled image. (c) VDSR. (d) LapSRN. (e) MemNet. (f) CARN. (g) MoreMNAS. (h) FALSR. (i) DeCoNASNet. (j) Proposed.

a larger receptive field to capture multi-scale features efficiently.

On the other hand, the searched configuration sequence for ×3 scale is

$$\begin{aligned} \mathbf{s}_1 &= \{1, 1, 0, 2\}, \\ \mathbf{s}_2 &= \{1, 1, 0, 2\}, \\ \mathbf{s}_3 &= \{1, 1, 2, 2\}. \end{aligned} \tag{16}$$

The ×3 scale SR task generally needs a larger receptive field than the ×2 to extract multi-scale features, and our searched ×3 network satisfies this property. It takes about 24 hours to train the controller and the child network by one Tesla V100 GPU in the search phase, which is far less than other NAS-based methods such as MoreMNAS [28] and FALSR [29]. To show the robustness of our search algorithm, we search three times from different random seeds. Table 1

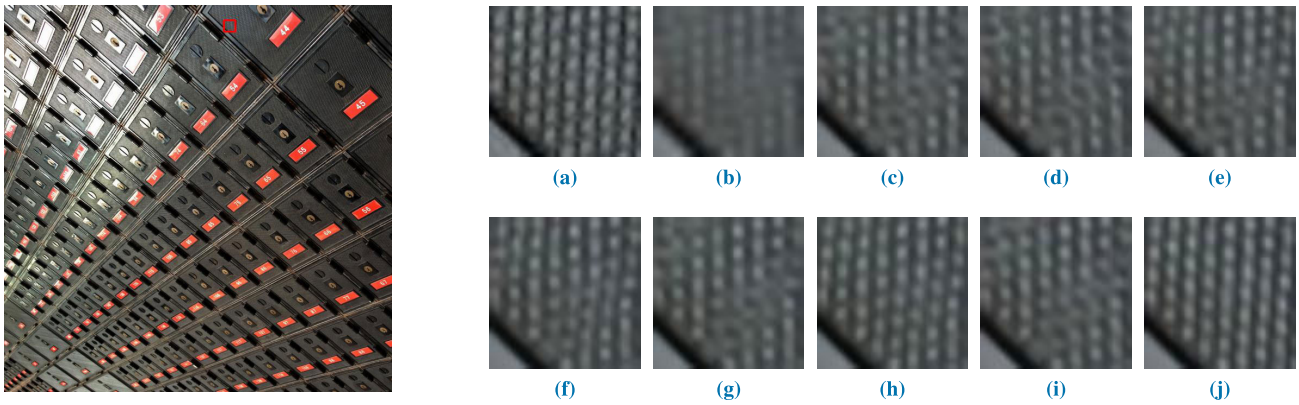

FIGURE 5. Qualitative result on the 6th image from the Urban100 dataset for ×2 SR task. We compare our method with nine conventional SR methods. (a) ground truth. (b) bicubic downsampled image. (c) VDSR. (d) LapSRN. (e) MemNet. (f) CARN. (g) MoreMNAS. (h) FALSR. (i) DeCoNASNet. (j) Proposed.

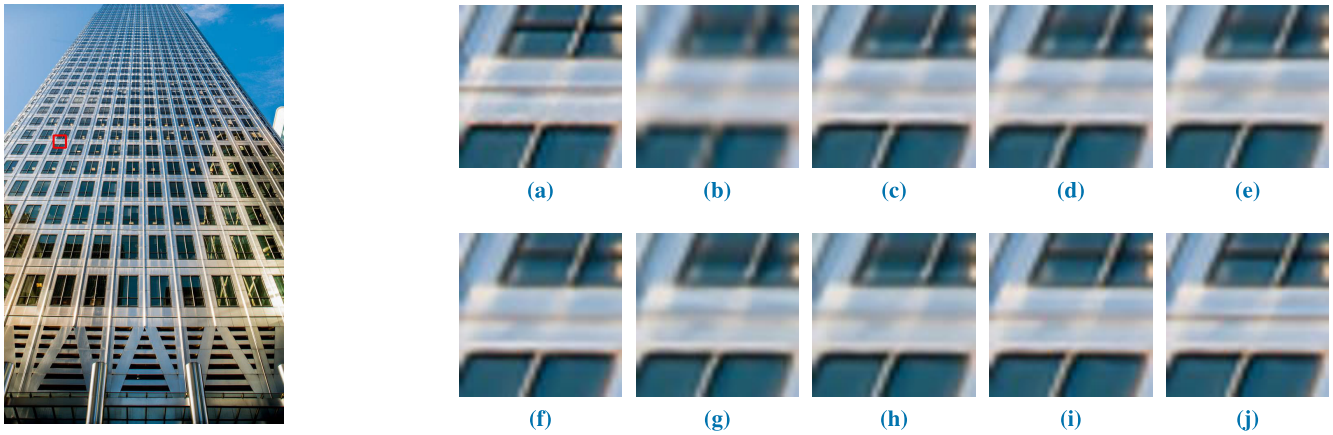

FIGURE 6. Qualitative result on the 30th image from the Urban100 dataset for ×2 SR task. We compare our method with nine conventional SR methods. (a) ground truth. (b) bicubic downsampled image. (c) VDSR. (d) LapSRN. (e) MemNet. (f) CARN. (g) MoreMNAS. (h) FALSR. (i) DeCoNASNet. (j) Proposed.

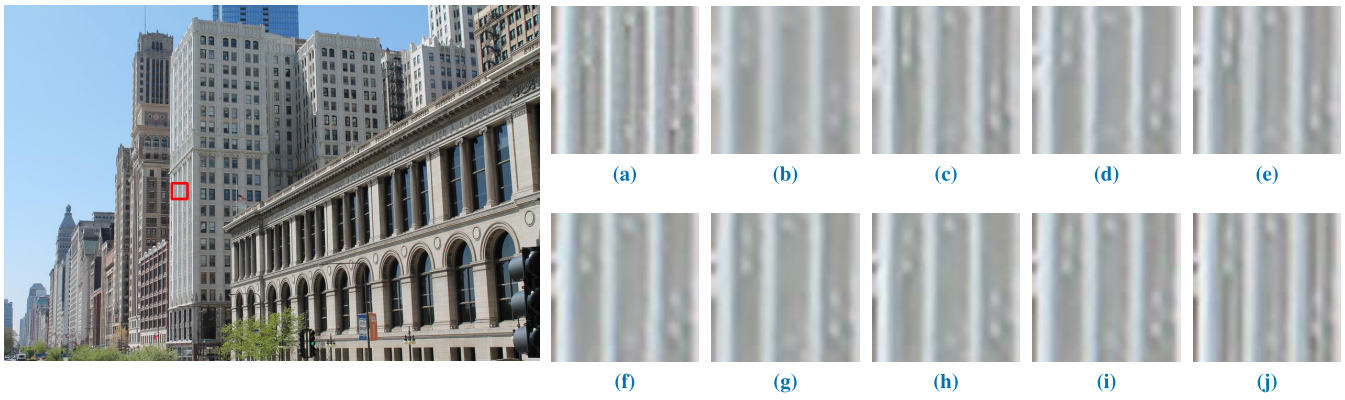

FIGURE 7. Qualitative result on the 97th image from the Urban100 dataset for ×2 SR task. We compare our method with nine conventional SR methods. (a) ground truth. (b) bicubic downsampled image. (c) VDSR. (d) LapSRN. (e) MemNet. (f) CARN. (g) MoreMNAS. (h) FALSR. (i) DeCoNASNet. (j) Proposed.

indicates the mean and variance of 500 searched networks from three different controllers for ×3 image super-resolution of Set5.

2) IMAGE SUPER-RESOLUTION RESULTS

Bicubic image down-sampling is widely used as the image degradation setting of super-resolution task. We measure PSNR and SSIM on four public benchmark dataset to compare our method with eleven state-of-the-art methods: SRCNN [4], VDSR [7], LapSRN [13], MemNet [10], MSAN [17], SelNet [38], CARN [12], A^2F [63], MoreMNAS [28], FALSR [29], HNAS [30], and DeCoNASNet [37]. Among these, MoreMNAS, FALSR, DeCoNASNet, HNAS, and ours are NAS-based aproaches. HNAS uses large training

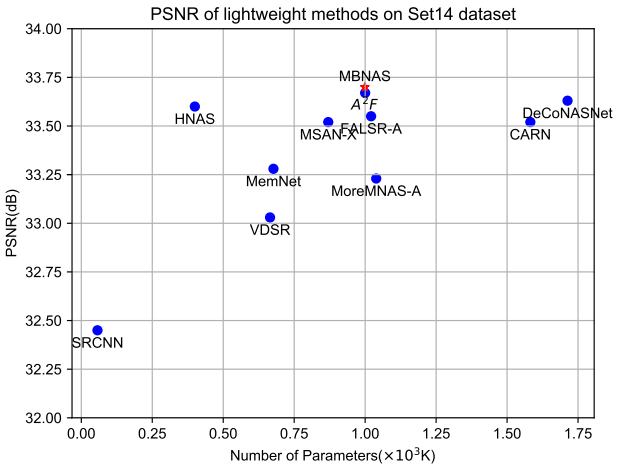

FIGURE 8. The graphical result of conventional lightweight methods and our MBNASNet on Set14 dataset. The Blue dots are conventional lightweight methods, and the red star is our MBNASNet method.

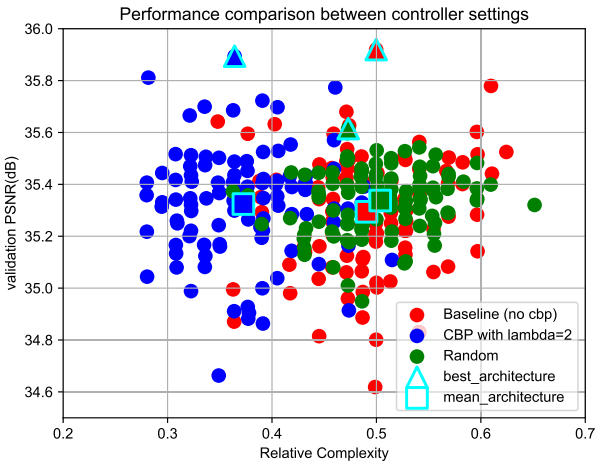

FIGURE 9. The result of three experiments for the controller. The blue dots are from the CBP, the reds are the Baseline, and the greens are Random settings. The "Relative Complexity" is defined the same as cbp in equation(3), meaning the cbp in the case of NAS design results. In the case of random and baseline, since the "penalty" is not defined, we denote it as "Relative Complexity."

patch (96 × 96) when training and applies self-ensemble to get better performances.

Table 2 shows the comparison with several state-of-the-art SISR networks, where boldfaced methods are NAS-based ones as ours, and non-bold are conventional hand-crafted designs. Since our NAS-based approach is based on an efficient search algorithm, which is about twenty times faster than MoreMNAS and FALSR, we could conduct experiments on x3 super-resolution tasks while other NAS-based methods did not. As shown in Table 2, MBNASNet performs comparable to hand-crafted state-of-the-art methods and outperforms the NAS-based ones in many situations. Specifically, HNAS shows good performance for Set5 dataset, but MBNASNet performs better for complex datasets such as Urban100 and B100 datasets because we extract multi-scale features successfully. Compared to a state-of-the-art hand-craft design A^2F-M [60], MBNASNet shows comparable results in the case of x2 SR, but slightly worse for x3. We believe the A^2F-M shows higher PSNR because they used more elements and technics (such as attentive auxiliary feature block and dense block connection) than our automatic design having only channel attention and feature fusion in block output. We believe we can bring possibly better results by employing more elements in our automated design, i.e., by further expanding search space. However, this may also induce huge design times so that we leave it as future work.

Since different initial conditions may lead to different results, we perform the design four times with different initial hyperparameters. But, there are just slight differences for all the cases in Table 2, with PSNR variance under 10^{-4}, validating the robustness of our method against different initial conditions. Hence, we denote the best PSNR among the four experiments, following the convention.

In Fig. 4 and Fig. 7, we display the qualitative result of our method and conventional methods. As shown in the figures, MBNASNet successfully restores the structures of the images. Specifically, our network recovers the gray vertical lines and holes in each image while other methods do not. In summary, we compare the overall ×2 performance of lightweight models graphically in Fig. IV-B1.

TABLE 3. Performance comparison between controller settings.

Search setting		Random	Baseline	CBP
Best	PSNR	35.62	**35.92**	35.89
	Penalty	0.473	0.500	**0.364**
Mean	PSNR	**35.34**	35.29	35.32
	Penalty	0.504	0.491	**0.373**

V. DISCUSSION

In this section, we discuss the effect of the proposed method's contributions; complexity-based penalty, multi-branch structure, and partially shared parameters.

A. EFFECT OF THE COMPLEXITY-BASED PENALTY TO THE PERFORMANCE OF CONTROLLER

To evaluate the controller's performance and the effect of complexity-based penalty in the search phase, we conduct three experiments.The first experiment uses a non-trained controller, which generates a random controller sequence (denoted as Random). The controller trained with the PSNR reward but without the complexity-based penalty is denoted as Baseline, and the one including the complexity-based penalty is denoted as CBP. We choose $\lambda = 2$ for the complexity-based penalty.

We sample 100 structures for each controller setting and measure the average and the best performance, as shown in Table 3. Also, their distributions are illustrated in Fig. 9, where blue dots are the results of the CBP with $\lambda = 2$, red dots correspond to the Baseline, and the greens to the Random. We can see that the Baseline setting finds better architectures than the Random in terms of PSNR, sometimes

TABLE 4. PSNR of MBNASNet with/without gradient flow control weights α on the public benchmark test data for ×2 SR tasks. We emphasize the difference between two experiment by blue texts. We train each architecture for 1000epochs.

Model	scale	Set 5	Set 14	B100	Urban100
MBNASNet without α	×2	37.9426(-0.034)	33.5727(-0.050)	32.1357(-0.006)	31.9743(-0.052)
MBNASNet with α		37.9767	33.6230	32.1413	32.0264

FIGURE 10. The PSNR on Set5 for three structures. The red line indicates our MBNASNet structure, the green line is the multi-branch structure with separate parameters, and the blue is the single branch structure.

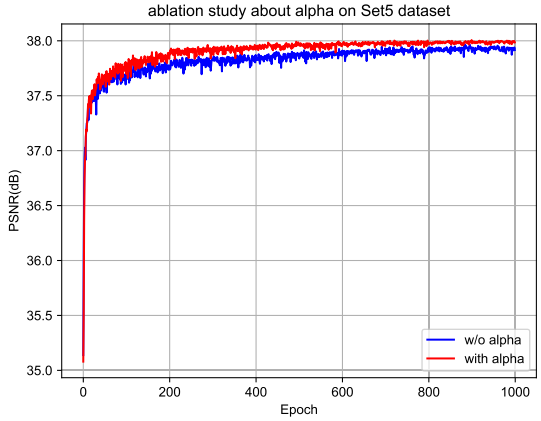

FIGURE 11. The PSNR on Set5 of MBNASNet architecture with/without gradient flow control weights α. The red line indicates our MBNASNet structure with α, and the blue is MBNASNet without α.

with increased complexity. On the other hand, the CBP setting successfully generates lightweight sequences that have comparable PSNR to the Baseline.

B. EFFECT OF MULTI-BRANCH STRUCTURE AND PARTIAL PARAMETER SHARING SCHEME

To compare and visualize the effect of multi-branch structure and partial parameter sharing (PPS) scheme, we create three networks; single-branch, multi-branch without PPS, multi-branch with PPS. We set the parameters of three experiments by $\sim 1,000K$ to fairly compare the results.

We measure the PSNR of each structure on the Set5 dataset. Fig. 10 shows the results of three structures for 400 epochs. We can find that the multi-branch structure converges faster than the single branch structure. Furthermore, with the partial parameter sharing scheme, we can successfully overcome the performance degradation phenomenon in the multi-branch structure.

C. EFFECT OF GRADIENT FLOW CONTROL WEIGHTS AND COMPLEXITY-BASED PENALTY COEFFICIENT

Gradient flow control weights allow MBNASNet to overcome the gradient vanishing problem by adjusting the gradient magnitude in the back-propagation process. We train MBNASNet with/without gradient flow control weights α and compare their performance in Table 4 and Fig. 11. The results show that α helps the MBNASNet converge to better point and achieve better performance.

To compare the effect of CBP weight λ, we train the controller with different λ values ($\lambda = 0.5, 1, 2, 4$) and compare their search results in Table 5. We can see that the mean CBP

TABLE 5. Mean PSNR and complexity-based penalty on different λ. The PSNR is calculated by Set5 benchmark dataset.

Experiment	mean PSNR	mean CBP
$\lambda = 0.5$	37.830	0.569
$\lambda = 1$	37.829	0.561
$\lambda = 2$	37.829	0.557
$\lambda = 4$	37.801	0.575

value tends to decrease (a lighter network is found), and the mean PSNR slightly decreases as the λ becomes larger. When the λ becomes too big ($\lambda = 4$), the controller fails to find a promising network in the search space. The experiments validate that the λ efficiently controls the trade-off between the performance and the number of parameters until $\lambda = 2$, and hence we use $\lambda = 2$ in other experiments.

VI. CONCLUSION

We have proposed a new NAS-based SR network, named as MBNASNet. We have attempted to improve the performance of the NAS-based SR by adopting a multi-branch network that can extract multi-scale features. In other words, we could obtain a better SR model by expanding the search space. We also regularized the reward signal of REINFORCE algorithm with a complexity-based penalty to favor a lightweight network. Besides, the partial parameter sharing scheme successfully reduces the number of parameters and helps the information transfer between each branch. It takes 24 hours to find promising network structures, which is a lot faster than the existing NAS-based design methods. The results show that the proposed method performs comparably to the

conventional hand-crafted structures and other NAS-based networks. We will release our codes and more result images at https://github.com/Junem360/MBNAS.

REFERENCES

[1] J. S. Isaac and R. Kulkarni, "Super resolution techniques for medical image processing," in *Proc. Int. Conf. Technol. Sustain. Develop. (ICTSD)*, Feb. 2015, pp. 1–6.

[2] W. W. W. Zou and P. C. Yuen, "Very low resolution face recognition problem," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 327–340, Jan. 2011.

[3] Y. Luo, L. Zhou, S. Wang, and Z. Wang, "Video satellite imagery super resolution via convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 12, pp. 2398–2402, Dec. 2017.

[4] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 184–199.

[5] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 391–407.

[6] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1874–1883.

[7] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1646–1654.

[8] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 136–144.

[9] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4799–4807.

[10] Y. Tai, J. Yang, X. Liu, and C. Xu, "MemNet: A persistent memory network for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4539–4547.

[11] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2472–2481.

[12] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 252–268.

[13] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 624–632.

[14] J. Li, F. Fang, K. Mei, and G. Zhang, "Multi-scale residual network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 517–532.

[15] L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *IEEE Trans. Image Process.*, vol. 15, no. 8, pp. 2226–2238, Aug. 2006.

[16] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," *IEEE Trans. Image Process.*, vol. 21, no. 11, pp. 4544–4556, Nov. 2012.

[17] J. W. Soh and N. I. Cho, "Lightweight single image super-resolution with multi-scale spatial attention networks," *IEEE Access*, vol. 8, pp. 35383–35391, 2020.

[18] C. Wang, Z. Li, and J. Shi, "Lightweight image super-resolution with adaptive weighted learning network," 2019, *arXiv:1904.02358*.

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

[20] 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 *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 19–34.

[21] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," 2018, *arXiv:1802.03268*.

[22] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1379–1388.

[23] 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 *Proc. Genetic Evol. Comput. Conf.*, Jul. 2019, pp. 419–427.

[24] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4780–4789.

[25] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.

[26] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7816–7827.

[27] M. Ahmad, M. Abdullah, H. Moon, S. J. Yoo, and D. Han, "Image classification based on automatic neural architecture search using binary crow search algorithm," *IEEE Access*, vol. 8, pp. 189891–189912, 2020.

[28] X. Chu, B. Zhang, and R. Xu, "Multi-objective reinforced evolution in mobile neural architecture search," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 99–113.

[29] X. Chu, B. Zhang, H. Ma, R. Xu, and Q. Li, "Fast, accurate and lightweight super-resolution with neural architecture search," 2019, *arXiv:1901.07261*.

[30] Y. Guo, Y. Luo, Z. He, J. Huang, and J. Chen, "Hierarchical neural architecture search for single image super-resolution," *IEEE Signal Process. Lett.*, vol. 27, pp. 1255–1259, 2020.

[31] Y. Weng, Z. Chen, and T. Zhou, "Improved differentiable neural architecture search for single image super-resolution," *Peer Peer Netw. Appl.*, vol. 14, no. 3, pp. 1806–1815, May 2021.

[32] H. Huang, L. Shen, C. He, W. Dong, H. Huang, and G. Shi, "Lightweight image super-resolution with hierarchical and differentiable neural architecture search," 2021, *arXiv:2105.03939*.

[33] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 82–92.

[34] M. Ding, X. Lian, L. Yang, P. Wang, X. Jin, Z. Lu, and P. Luo, "HR-NAS: Searching efficient high-resolution neural architectures with lightweight transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 2982–2992.

[35] C. He, H. Ye, L. Shen, and T. Zhang, "MiLeNAS: Efficient neural architecture search via mixed-level reformulation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11993–12002.

[36] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*.

[37] J. Y. Ahn and N. I. Cho, "Neural architecture search for image super-resolution using densely constructed search space: DeCoNAS," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 4829–4836.

[38] J.-S. Choi and M. Kim, "A deep convolutional neural network with selection units for super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 154–160.

[39] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 286–301.

[40] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

[41] G. Li, G. Qian, I. C. Delgadillo, M. Müller, A. Thabet, and B. Ghanem, "SGAS: Sequential greedy architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1620–1630.

[42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[43] N. Nguyen and J. M. Chang, "Contrastive self-supervised neural architecture search," 2021, *arXiv:2102.10557*.

[44] C. Liu, P. Dollár, K. He, R. Girshick, A. Yuille, and S. Xie, "Are labels necessary for neural architecture search?" in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 798–813.

[45] S. Kaplan and R. Giryes, "Self-supervised neural architecture search," 2020, *arXiv:2007.01500*.

[46] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jan. 26, 2021, doi: 10.1109/TPAMI.2021.3054824.

[47] M. Lindauer and F. Hutter, "Best practices for scientific research on neural architecture search," *J. Mach. Learn. Res.*, vol. 21, no. 243, pp. 1–18, 2020.

[48] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-Bench-101: Towards reproducible neural architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7105–7114.

[49] B. Wu, K. Keutzer, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, and Y. Jia, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10734–10742.
[50] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, P. Vajda, and J. E. Gonzalez, "FBNetV2: Differentiable neural architecture search for spatial and channel dimensions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12965–12974.
[51] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[52] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15.
[54] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 114–125.
[55] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf.*, R. Bowden, J. Collomosse, and K. Mikolajczyk, Eds. Surrey, BC, Canada: BMVA Press, Sep. 2012, pp. 135.1–135.10, doi: 10.5244/C.26.135.
[56] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. Int. Conf. curves Surf.* Berlin, Germany: Springer, 2010, pp. 711–730.
[57] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2, Jul. 2001, pp. 416–423.
[58] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5197–5206.
[59] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[60] MATLAB. *Version 9.8.0.1298242 (R2020a)*. Natick, MA, USA: The MathWorks, 2020.
[61] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, May 2016, pp. 1–13.
[62] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[63] X. Wang, Q. Wang, Y. Zhao, J. Yan, L. Fan, and L. Chen, "Lightweight single-image super-resolution network with attentive auxiliary feature learning," in *Proc. Asian Conf. Comput. Vis.*, 2020, pp. 1–17.

JOON YOUNG AHN (Member, IEEE) received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2014 and 2021, respectively. He is currently with Samsung Electronics. His research interests include computer vision, machine learning, image restoration, and neural architecture search.

NAM IK CHO (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in control and instrumentation engineering from Seoul National University, Seoul, South Korea, in 1986, 1988, and 1992, respectively. From 1991 to 1993, he was a Research Associate with the Engineering Research Center for Advanced Control and Instrumentation, Seoul National University. From 1994 to 1998, he was an Assistant Professor of electrical engineering with the University of Seoul. In 1999, he joined the Department of Electrical and Computer Engineering, Seoul National University, where he is currently a Professor. His research interests include image processing, adaptive filtering, digital filter design, and computer vision.