Noname manuscript No.
(will be inserted by the editor)

Memory-Efficient Hierarchical Neural Architecture Search for Image Restoration

Haokui Zhang[1], **Ying Li**[1], **Chengrong Gong**[1], **Hao Chen**[2], **Zongwen Bai**[1,3], **Chunhua Shen**[2]

Received: date / Accepted: date

Abstract Recently, much attention has been spent on neural architecture search (NAS) approaches, which often outperform manually designed architectures on high-level vision tasks. Inspired by this, we attempt to leverage NAS technique to automatically design efficient network architectures for low-level image restoration tasks. In this paper, we propose a memory-efficient hierarchical NAS HiNAS (HiNAS) and apply to two such tasks: image denoising and image super-resolution. HiNAS adopts gradient based search strategies and builds an flexible hierarchical search space, including inner search space and outer search space, which in charge of designing cell architectures and deciding cell widths, respectively. For inner search space, we propose layer-wise architecture sharing strategy (LWAS), resulting in more flexible architectures and better performance. For outer search space, we propose cell sharing strategy to save memory, and considerably accelerate the search speed. The proposed HiNAS is both memory and computation efficient. With a single GTX1080Ti GPU, it takes only about *1 hour* for searching for denoising network on BSD 500 and *3.5 hours* for searching for the super-resolution structure on DIV2K. Experimental results show that the architectures found by HiNAS have fewer parameters and enjoy a faster inference speed, while achieving highly competitive performance compared with state-of-the-art methods.

Keywords Neural architecture search · Hierarchical search space · Memory and computation efficiency · Denosing · Super-resolution

Part of this work was done when H. Zhang was visiting The University of Adelaide. This work was in part supported by National Natural Science Foundation of China (61871460, 61941112, 61761042), Shaanxi Provincial Key RD Program (2020KW-003), Natural Science Foundation of China of Shaanxi (No. 2020JM-556) and the Fundamental Research Funds for the Central Universities (3102019ghxm016).

HZ, YL, CG
[1]Northwestern Polytechnical University, China

ZB
[1]Northwestern Polytechnical University and [3]Shaanxi Key Laboratory of Intelligent Processing for Big Energy Data, Yan'an University, China

HC, CS
[2]The University of Adelaide, Australia

1 Introduction

As a classical task in computer vision, image restoration aims to estimate the underlying image from its degraded measurements, which is known as an ill-posed inverse procedure. Depending on the type of degradation, image restoration can be categorized into different sub-problems, e.g., denoising, super-resolution, debluring, dehazing, inpainting and deraining, etc. In this work, we focus on image denoising and image super-resolution.

Image denoising aims to restore a clean image from a noisy one. Owing to the fact that noise corruption always occurs in the image sensing process and may degrade the visual quality of collected images, image denoising is needed for various computer vision tasks (Chatterjee and Milanfar, 2009). Traditional image denoising methods generally focus on modeling natural image priors and use the priors to restore the clean image, including sparse models (Dong et al., 2012), Markov random field models (Lan et al., 2006), etc. One drawback of these methods is that most of them involve a complex optimization problem and can be time-consuming for inference (Dabov et al., 2007; Gu et al., 2014).

The purpose of single image super-resolution is recovering a high-resolution image from the corresponding low-resolution one, which is widely applied on various applications. In the early stage, single image super-resolution methods mainly are interpolation-based methods. Such as nearest-neighbor, bilinear and bicubic. Although theses interpolation-based methods are simple in theory and efficient in inference, the performances are poor. Later, more advanced model-based approaches are proposed, where powerful priors are employed. Compared with interpolation-based methods, model-based approaches have higher reconstruction results. However, model-based approaches suffer some new drawbacks: 1) time-consuming problem. Similar with traditional image denoising methods, model-based image super-resolution algorithms usually involves a time-consuming optimization process, which hinder model-based methods from applying on time restricted applications. 2) Low generalization ability. When image statistics change, the performance may degrade quickly.

Recently, deep learning models have been successfully applied in various computer vision tasks and set new state-of-the-art. Motivated by this, most recent works on image denoising and image super-resolution have shifted their approaches to deep learning, which builds a mapping function from low quality images to the desired corresponding high quality images with deep learning models and have often outperformed conventional methods significantly (Mao et al., 2016; Tai et al., 2017; Tobias Plötz, 2018; Liu et al., 2019d; Dong et al., 2015; Zhang et al., 2018c,e; Dai et al., 2019). To date most deep learning methods focus on improving the quality of the restored images, and largely neglect the inference speed. As such, these image restoration models typically contain millions or even tens of millions of parameters and parts of these methods involve a recurrent optimization process to improve restoration quality, resulting in low inference speed. In addition, discovering state-of-the-art neural network architectures is not trivial and it requires substantial efforts.

Recently a growing interest is witnessed in developing algorithmic solutions to automate the manual process of architecture design. Architectures automatically found by algorithms have achieved highly competitive performance in high-level vision tasks such as image classification (Zoph and Le, 2017), object detection (Ghiasi et al., 2019; Wang et al., 2020) and semantic segmentation (Liu et al., 2019a; Nekrasov et al., 2019).

Very recently, several works about using NAS algorithms to design architectures for image restoration have been proposed. For instance, FALSR (Chu et al., 2019a), E-CAE (Suganuma et al., 2018) and EvoNet (Liu et al., 2019c). Compared with the manually designed architectures, the architectures found by NAS algorithms achieve higher performance and have fewer parameters. However, the search process of these methods are computationally expensive and time consuming. FALSR takes about 3 days on 8 Tesla V100 GPUs to find the best architectures. By using of four Tesla V100 GPUs, E-CAE spends 44 hours on searching. In this paper, we design a memory-efficient NAS algorithm to automatically search for neural architectures efficiently for low-level image processing tasks. We show the effectiveness of our method on two such applications, namely image denoising and image super-resolution. Our main contributions can be summarized as follows.

1. Based on gradient based search algorithms, we propose a memory-efficient hierarchical neural architecture search approach for image denoising and image super-resolution, termed HiNAS. To our knowledge, this is the first attempt to apply differentiable architecture search algorithms to low-level vision tasks.
2. We propose layer-wise architecture sharing strategy to improve the flexibility of search space and propose cell sharing to save memory, obtaining a memory and computation efficient image restoration tasks aimed NAS algorithm which only takes several 1 to 3.5 hours for searching for networks architectures for image restoration tasks with a single GTX 1080 Ti GPU.
3. We apply our proposed HiNAS on two denoising datasets of different noise modes for evaluation. Experiments show that the networks found by our HiNAS achieves highly competitive performance compared with state-of-the-art algorithms, while having fewer parameters and a faster speed.
4. We conduct comparison experiments to analyse the network architectures found by our NAS algorithm in terms of the internal structure, offering some insights in architectures found by NAS.

2 Related work

CNNs for Image Denoising and Super-resolution Currently, due to the popularity of convolutional neural networks (CNNs), image restoration algorithms including image denoising and image super-resolution have achieved a significant performance boost. For image denoising, DnCNN (Zhang et al., 2017a) and IrCNN (Zhang et al., 2017b), predict the residue present in the image instead of the denoised image as the input to the loss function is ground truth noise as compared to the original clean image. Lately, FFDNet (Zhang et al., 2018b) attempts to address spatially varying noise by appending noise level maps to the input of DnCNN.

NLRN (Liu et al., 2018a) incorporates non-local operations into a recurrent neural network (RNN) for image restoration. N3Net (Tobias Plötz, 2018) formulates a differentiable version of nearest neighbor search to further improve DnCNN. Recently, some algorithms focus on denoising for real-noisy images since many existing denoisers tend to be overfitted to additive white Gaussain noise (AWGN) and generalize poorly to real-world noisy images with more sophisticated noise. CBDNet (Guo et al., 2019) uses simulated camera pipeline to supplement real training data. In (Jaroensri et al., 2019) camera simulator was proposed to accurately simulate the degradation and noise transformation performed by camera pipelines.

For image super-resolution, the first attempt is proposed in (Dong et al., 2015), where Dong et al. build a shallow CNN model which consists of three convolutional layers. Although three layers model is relatively shallow, it achieved impressive performance. Later, consistent with the development of CNN in other fields, CNN based super-resolution approaches trend to introduce deeper CNN models. For instance, Kim et al. proposed VDSR (Kim et al., 2016a) and DRCN (Kim et al., 2016b). Both VDSR and DRCN contain 20 convolutional layers and employ residual learning strategy. To address the issue of lacking of long-term memory (one state is usually influenced by a specific prior state). Tai et al. (Tai et al., 2017) designed memory block based on recursive unit and gate unit, then build a MemNet by stacking the proposed block and connect them with skip connections. In addition to designing deeper networks to improve efficiency, some attempts take spatial correlations into consideration and several trials embed attention mechanism in their networks. For example, NLRN (Liu et al., 2018a) incorporates non-local module in an recurrent network. Based on SENet (Hu et al., 2018), Zhang et al. (Zhang et al., 2018d) build a very deep residual channel attention network for SR. Recently, Dai et al. (Dai et al., 2019) further extended the first-order channel attention to second-order channel attention and proposed SAN network. Liu et al. (Liu et al., 2020) groups several residual modules together via skip connection to build a residual feature aggregation framework, then proposed an enhanced spatial attention block to further improve the framework.

In fact, except repairing quality, runtime-efficient is also very important for image restoration tasks. Image restoration algorithms which can restore high-quality images from low-quality images in real time will be very useful. Based on light-weight designing, Ahn et al. (Ahn et al., 2018) proposed a cascading residual network and Hui et al. (Hui et al., 2019) combined light-weight designing with information multi-distillation. Very recently, Lee et al. (Lee et al., 2020) boost the performance of FSRCNN (Hui et al., 2018) by using knowledge distillation without any negative influence on inference speed. However, it is hard to get the best of both worlds. Compared with performance focused methods, speed focused approaches usually show lower performance. Balancing the performance and speed remains a challenge.

Network architecture search (NAS). NAS aims to design automated approaches for discovering high-performance neural architectures such that the procedure of tedious and heuristic manual design of neural architectures can be eliminated from the deep learning pipeline. Early trials employ evolutionary algorithms (EAs) for optimizing neural architectures and parameters. The best architecture may be obtained by iteratively mutating a population of candidate architectures (Liu et al., 2018b). An alternative to EA is to use reinforcement learning (RL) techniques, e.g., policy gradients (Zoph et al., 2018; Wang et al., 2020) and Q-learning (Zhong et al., 2018), to train a recurrent neural network that acts as a meta-controller to generate potential architectures—typically encoded as sequences—by exploring a predefined search space. However, EA and RL based methods have a common drawback, inefficient in search, often requiring a large amount of computations. Speed-up techniques are proposed to remedy this issue. Exemplar works include hyper-networks (Zhang et al., 2018a), network morphism (Elsken et al., 2018) and shared weights (Pham et al., 2018).

Our work is closely related to DARTS (Liu et al., 2019b), ProxylessNAS (Cai et al., 2019) and Auto-Deeplab (Liu et al., 2019a). All these methods take the architecture searching process as an optimization process. They first build a supernet consists of all possible layer types and corresponding weight parameters as search space; then they optimize the parameters of this supernet via gradient descent algorithm; finally they derive the final architecture according to weight parameters. DARTS is based on the continuous relaxation of the architecture representation, allowing efficient search of the cell architecture using gradient descent, which has achieved competitive performance. Motivated by this search efficiency, following Darts, ProxylessNAS and Auto-Deeplab, HiNAS also employed the gradient based approach as its search strategy. Differing from Darts, the later two NAS algorithms included widths in their search space. ProxylessNAS discovers sequential structures and chooses kernel widths within manually designed blocks (Inverted Bottlenecks (He et al., 2016)). By introducing multiple paths with different widths, Auto-Deeplab extended its search space to cover widths. The search space of our proposed HiNAS resembles Auto-Deeplab. The three main differences are: 1) in designing the search space, we abandon pooling operations to retain high resolution feature maps and rely on automatically selected operation with different receptive fields to adapt the receptive field; 2) we propose layer-wise architecture sharing strategy and cell sharing strategy to improve flexibility and memory efficiency, improving performance and accelerating search speed; In addition, the first three methods are proposed high-level image understanding tasks. DARTS (Liu et al., 2019b) and ProxylessNAS (Cai et al., 2019) are proposed for image classification. Auto-Deeplab (Liu et al., 2019a) finds architectures for semantic segmentation. HiNAS is proposed for low-level image restoration tasks.

In terms of using NAS algorithm to search for neural network architectures for low-level image restoration tasks, three most related works are are EvoNet (Liu et al., 2019c), E-CAE (Suganuma et al., 2018) and FALSR (Chu et al., 2019a). EvoNet searches for networks for medical image denoising via EA. E-CAE employs EA to search for an architectures of convolutional autoencoders for image inpainting and denoising. FALSR is proposed for super resolution tasks. FALSR combines RL and EA and design a hybrid controller as its model generator. All three methods mentioned above require a relatively large amount of computations and takes a large amout of GPU time for searching. Compared with these methods, our proposed HiNAS has different search space, search strategy and higher search efficiency.

3 Our Method

Our proposed HiNAS is a gradient-based architecture search algorithm. It search L different computation cells, where L denotes the number of cells. The final architecture is build by stacking the found L cells with different widths one by one. To able to search both cell topological architectures and cell widths, HiNAS builds a hierarchical search space. The inner space is responsible for searching inner cell topological architectures and the outer space in charge of searching cell widths. To further improve the efficiency of HiNAS, we propose cell sharing strategy, allowing features from different levels of outer search space to share one cell.

In this section, we first introduce how to search for architectures of cells based on continuous relaxation (inner search space). Then we explain how to determine the widths via multiple candidate paths and cell sharing (outer search space). Then, we elaborate on residual learning frameworks of image deraining and super-resolution. Last, we present our search strategy and our the loss functions.

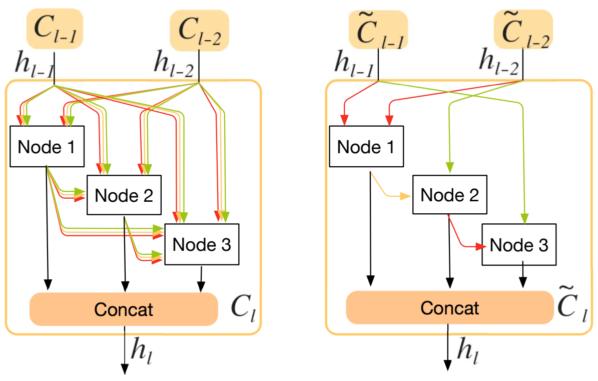

Fig. 1 Inner cell architecture search. Left: supercell that contains all possible layer types. Right: the cell architecture search result, a compact cell, where each node only keeps the two most important inputs and each input is connected to the current node with a selected operation.

3.1 Inner Search Space

To search for inner cell architecture, we first build a super-cell containing N nodes. Different nodes in this supercell are connected with different path and each path contains all possible operations. Each operation in each path corresponds to a weight α, which denotes the importance of this layer type in current path. The purpose of cell architecture search is learning a set of continuous variables $\{\alpha\}$. During search, continuous variables are updated via gradient descent algorithm. At the end of search, a discrete architecture can be obtained by keeping the top two likely layer types and discarding the rest for each node. Next, we explain this process in mathematical formula.

Shared inner cell architecture search. We denote the super cell in layer l as C_l, which takes outputs of previous cells and the cell before previous cells as inputs and outputs a tensor h_l. As shown in Figure 1. The left side is supercell containing all possible operations. It is a directed acyclic graph consists of three nodes. Inside C_l, each node takes the two inputs of the current cell and the outputs of all previous nodes as input and outputs a tensor h_l. When all layers share the same cell architecture, the output of the ith node in cell is calculated as:

$$\begin{aligned} x_{l,i} &= \sum_{x_j \in I_{l,i}} O^{j\to i}(x_j), \\ O^{j\to i}(x_j) &= \sum_{k=1}^{S} \alpha_k^{j\to i} o^k(x_j), \end{aligned} \tag{1}$$

where $I_{l,i} = \{h_{l-1}, h_{l-2}, x_{l,j<i}\}$ is the input set of node i. h_{l-1} and h_{l-2} are the outputs of cells in layers $l-$

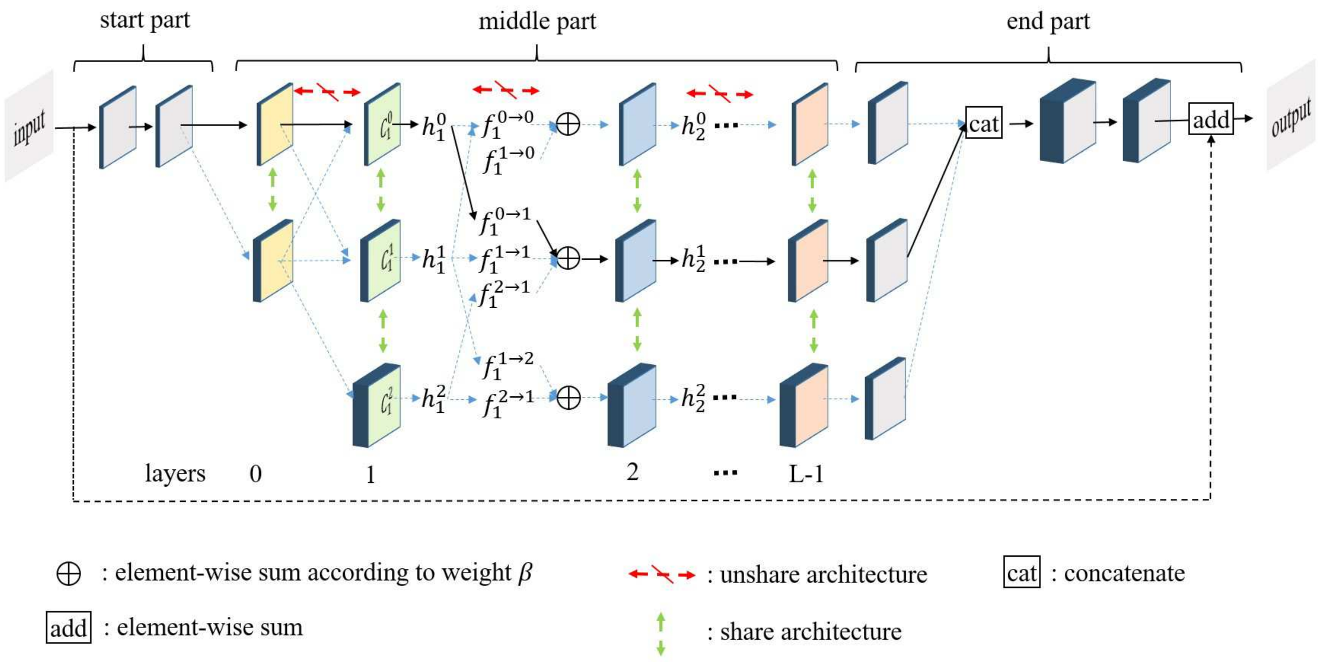

Fig. 2 The whole framework of supernet. The supernet contains three parts, including start part, middle part and end part. The start and end parts are manually pre-designed and their architectures are fixed during search. The middle part is designed by search algorithm. It consists of L layers, each of which contains three cells of different widths. In the proposed HiNAS, cells in the same layer share same architecture and cells in different layers can search for different architectures, which further improves the flexibility of search space, resulting in better performance, while having no additional computing burden. During search, all paths are activated and the weight β is updated via gradient descent. At the end of search, each layer keeps only one path, which is marked in a black arrow.

1 and $l-2$, respectively. $O_l^{j\to i}$ is the set of possible layer types. $\{o^1, o^2, \cdots, o^S\}$ correspond to S possible operations. $\alpha_k^{j\to i}$ denotes the weight of operator o^k.

Layer-wise architecture sharing (LWAS). In fact, in the early stage, the proposed NAS methods employ a very complex search space to search the whole network directly, requiring a huge number of GPUs and taking a long time. For instance, Zoph et al. use hundreds of GPUs train for several days to search for architectures on Cifar-10 (Zoph and Le, 2017). Later, Zoph et al. propose NASNet search space, which search for an architectural building block then build the whole network by stakcing the found block with pre-designed outline structure (Zoph et al., 2018). Compared with the method in (Zoph and Le, 2017), NASNet search space has much faster search speed. Inspired by this, NASNet search space is widely used in most of recent works. NASNet search space significantly reduce the complicity of search space, which is very useful for improving the search speed in RL based on EA based NAS algorithms. Reducing the complicity of search space means training fewer student networks to get an reliable controller in RL and means training fewer gene network to finish the EA process. NASNet search space accelerates the search speed on one hand and reduces the flexibility of search space on the other hand. It restricts different layers of networks share the same cell architecture. In our proposed HiNAS, we adopt a more flexible search space that is searching for different cell architectures for different layers. In our gradient based HiNAS, we need to introduce more continuous variables α without any additional computational cost. In our new search space, the output of the ith node is computed with:

$$\begin{aligned} x_{l,i} &= \sum_{x_j \in I_{l,i}} O_l^{j\to i}(x_j), \\ O_l^{j\to i}(x_j) &= \sum_{k=1}^{S} \alpha_l^{k,j\to i} o^k(x_j), \end{aligned} \tag{2}$$

here, $\alpha_l^{k,j\to i}$ is l related. The sets of continuous variables α in different layers are different. Cells in the same layer share same set of continuous variables α. h_l is the concatenation of the outputs of N nodes and it can be expressed as:

$$\begin{aligned} h_l &= \text{Cell}(h_{l-1}, h_{l-2}) \\ &= \text{Concat}\{x_{l,i} | i \in \{1, 2, \cdots, N\}\}. \end{aligned} \tag{3}$$

Search space. In this paper, we employs the following 7 types operators:

– conv: 3×3 convolution;

– sep: 3×3 separable convolution;
– sep: 5×5 separable convolution;
– dil: 3×3 convolution with dilation rate of 2;
– dil: 5×5 convolution with dilation rate of 2;
– skip: skip connection;
– none: no connection and return zero.

To preserve pixel-level information for low-level image processing, we abandon downsample operations such as pooling layers and setting stride to 2 in convolution layers. For convolution operators, we employ three convolution types including common convolution, separable convolution and dilation convolution. Each convolution operator, in order, consists of a LeakyReLU activation layer, a convolution layer and a batch normalization layer. We provide two different kernel sizes 3 and 5 for separable and dilation convolutions. The search algorithm can adapt receptive field by selecting convolution operators with different kernel sizes. An example of found compact cell is shown in the right-side of Figure 1.

3.2 Outer Search Space

Now, the main idea of searching specific architectures inside cells has been presented. Beside search architectures inner cells, we still need to either *heuristically* set the width of each cell or *automatically* search for a proper width for each cell to build the overall network.

Multiple candidate paths. In conventional CNNs, the change of widths of convolution layers is often related to the change of spatial resolutions. For instance, once the features are downsampled, the widths of following convolution layers are doubled. In our HiNAS, instead of using downsample operations such as pooling layers and setting stride stride to 2 in convolution layers, we rely on operations with different receptive field such as dilation convolution operations of 3×3 and 5×5, and separable convolution operations of 3×3 and 5×5 to adjust the receptive field automatically. Thus the conventional experience of adjusting width no longer applies to our case. To solve this problem, we employ the flexible hierarchical search space and leave the task of deciding width of each cell to the NAS algorithm itself, making the search space more general. In fact, several NAS algorithms in the literature also search for the outer layer width, mostly for high-level image understanding tasks. For example, FBNet (Wu et al., 2019) and MNASNet (Tan et al., 2019) consider different expansion rates inside their modules to discover compact networks for image classification.

In this section, we introduce the outer layer width search space which determines the widths of cells in different layers. Similarly, we build a supernet that contains several supercells with different widths in each layer. As illustrated in Figure 2, the supernet mainly consists of three parts:

1) *start part*, consisting of input layer and two convolution layer. A relatively shallow feature is extracted by feeding input data to two convolution layers and a copy of input data is propagated to end part via skip connection;

2) *middle part*, containing L layers and each layer having three supercells of different widths. The main part of supernet. The shallow feature extracted in start part is fed to different cells of layer 0, then the outputs of layer 0 are fed to different cells of layer 1, and so on;

3) *end part*, concatenating the outputs of layer $L-1$, then feeding them to two convolution layers to generate the residual, finally element-wise summing the learned residual and the output of skip connection to get the final result.

Except layer 0, which contains two cells of different widths, our supernet provides three paths of cells with different widths, which correspond to three decisions: 1) reducing the width; 2) keeping previous width; 3) increasing the width. After searching, only one cell at each layer is kept. The continuous relaxation strategy mentioned in the cell architecture search section is reused in here.

At each layer l, there are three cells C_l^0, C_l^1 and C_l^2 with widths $\gamma^0 \times W$, $\gamma^1 \times W$ and $\gamma^2 \times W$, where W is the basic width and γ^0, γ^1 and γ^2 are width changing factors. The output feature of each layer is

$$h_l = \{h_l^0, h_l^1, h_l^2\}, \tag{4}$$

where h_l^i is the output of C_l^i. The channel width of h_l^i is $2^i NW$, where N is the number of nodes in the cells.

Cell sharing. Each cell C_l^i is connected to C_{l-1}^{i-1}, C_{l-1}^i and C_{l-1}^{i+1} in the previous layer and C_{l-2}^i two layers before. We first process the outputs h_{l-1} from those layers with a 1×1 convolution to form features f_{l-1} with width $2^i W$, matching the input of C_l^i. Then the output

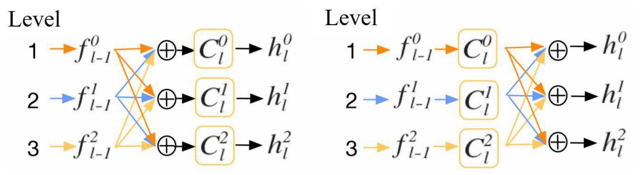

Fig. 3 Comparison of cases of whether using cell sharing or not. Left: features from different levels share same cell. Using cell sharing; Right: features from different levels use different cells.

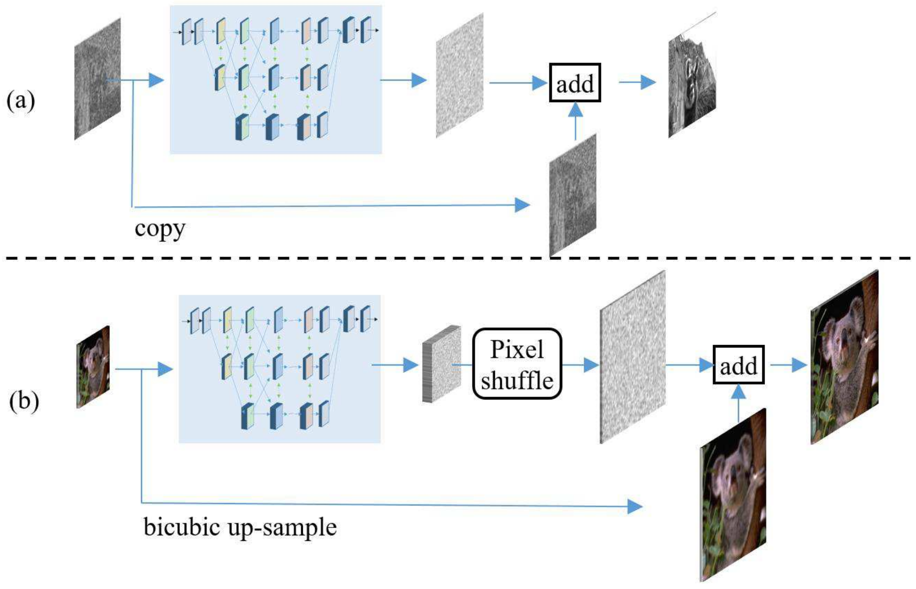

Fig. 4 Residual learning frameworks. (a) residual learning framework for denoising: (b) residual learning framework for super-resolution.

for the ith cell in layer l is computed with

$$h_l^i = C_l^i\left(\sum_{k=i-1}^{i+1} \beta_k^i f_{l-1}^k, f_{l-2}^i\right), \tag{5}$$

where β_k^i is the weight of f_{l-1}^k. We combine the three outputs of C_{l-1} according to corresponding weights then feed them to C_l^i as input. Here, features f_{l-1}^{i-1}, f_{l-1}^i and f_{l-1}^{i+1} come from different levels, but they share the cell C_l^i during computing h_l^i.

Note the similarity of this design with Auto-Deeplab, which is used to select feature strides for image segmentation. However, in Auto-Deeplab, the outputs from the three different levels are first processed by separate cells with different sets of weights before summing into the output:

$$h_l^i = \sum_{k=i-1}^{i+1} \beta_k^i C_l^k(f_{l-1}^k, f_{l-2}^i), \tag{6}$$

A comparison between Eqs. (5) and (6) is shown in Figure 3, where the inputs from layer $l-1$ are not shown out for simplicity.

For the hierarchical structure which has three candidate paths, the cell in each candidate path is used once with Eq. (5) and it is used three times with Eq. (6). By sharing the cell C_l^i, we are able to save the memory consumption by a factor of 3 in the supernet. Cell sharing has two main advantages: 1) improving applicability. NAS in general consumes much memory and computation. Improving memory efficiency enables much broader applications. 2)improving searching efficiency. As cell sharing saves memory consumption in the supernet, during search, we can use larger batch sizes during search to increase the search speed. We can also use a deeper and wider supernet for more accurate approximations.

Outer layer widths search looks like an extension of inner cell search, but the way to get the final widths of different layers are different from the method we used to get the final compact cell architecture in inner cell architecture search. If we follow previous actions in inner cell search, the the widths of adjacent layers in the final network may change drastically, which has a negative impact on the efficiency, as explained in (Ma et al., 2018). To avoid this problem, we view the β values as probability, then use the Viterbi decoding algorithm to select the path with the maximum probability as the final result.

3.3 Residual Learning

In our HiNAS, instead of learning the final restoration results directly, we learning residual information between low quality image and high quality to further improve the performance. Figure 4 shows residual learning frameworks for image denoising and image super-resolution.

For image denoising, a copy of input data is propagated to the end of the framework directly, then added to the output of the network to generate the final restoration result, as shown in Figure 4 (a).

For image super-resolution, residual learning is not a simple matter, as the spatial resolutions of input and output are different. In image super-resolution, the inputs are low resolution images and the outputs are high resolution images. There are two widely used strategies to overcome the mismatch problem of spatial resolution. One is upsampling the input image using bicubic interpolation then applying a CNN model (Dong et al., 2015), increasing the computational complexity. The other one is upscaling the output features at the very end of the network using pixel shuffle operation, reducing the computational complexity (Shi et al., 2016). In our paper, we adopt the latter strategy. As shown in Figure 4 (b), taking a $M \times N \times 3$-sized image as input, the network generates a $M \times N \times 3SS$-sized output, where M, N and S denote the spatial resolution of input image and upscale factor of super-resolution, respectively. Then, pixel shuffle operation is used to convert the $M \times N \times 3SS$-sized output to $MS \times NS \times 3$-sized residual. Finally, the learned residual is add to the upscaled input image to generate the final restoration result.

3.4 Searching Using Gradient Descent

The searching process in our proposed HiNAS is the optimization process of supernet. Differing from previous image restoration approaches which often take a single item $L2$ norm as their losses, the loss of our HiNAS has two items. Inspired by the fact that PSNR and SSIM (Wang et al., 2004) are the two most widely used evaluation metrics in image restoration tasks, we design a loss containing too items, which are correspond to the two evaluation metrics, respectively. Our optimization function can be represented as:

$$\begin{aligned}
&\text{loss} = \|\mathrm{f}_{\text{net}}(x) - y\|_2^2 + \lambda \cdot \mathrm{l}_{\text{ssim}}(\mathrm{f}_{\text{net}}(x), y),\\
&\mathrm{l}_{\text{ssim}}(x, y) = \log_{10}(\text{ssim}(x, y)^{-1}), \qquad (7)\\
&\mathrm{f}_{\text{net}}(x) = \mathrm{f}_{\text{res}}(x) + \mathrm{f}_{\text{skip}}(x),
\end{aligned}$$

where x and y represent the input image and corresponding ground-truth. $\mathrm{l}_{\text{ssim}}(\cdot)$ is a loss item that is designed to enforce the visible structure of the result. $\mathrm{f}_{\text{res}}(\cdot)$ is the supernet, which learning residual from input images. $\mathrm{f}_{\text{res}}(\cdot)$ is skip connection and it is a copy operation for image denoising and bicubic up-sample operation for image super-resolution. $\text{ssim}(\cdot)$ is structural similarity (Wang et al., 2004). λ is a weighting coefficient and it is empirically set to 0.6 in all of our experiments.

During optimization of the supernet with gradient descent, we find that the performance of network founded by HiNAS is often observed to *collapse* when the number of search epochs becomes large. The very recent method of Darts+ (Liang et al., 2019), which is concurrent to this work here, presents similar observations. Because of this collapse issue, it is hard to pre-set the number of search epochs. To solve this problem, we keep the supernet obtaining the best performance during evaluation as the final search result. Specifically, we split the training set into three disjoint parts: Train W, Train A and Validation V. Sub-datasets W and A are used to optimize the weights of the supernet (kernels in convolution layers) and weights of different layer types and cells of different widths (α and β). During optimizing, we periodically evaluate the performance of the trained supernet on the validation dataset V and record the best performance. After the search process is finished, we choose the supernet which corresponds to the best performance record as the result of the architecture search. More details are presented in the search settings of Section 4.

4 Experiments

4.1 Datasets and implementation details

We carry out the denoising experiments on one widely used dataset, BSD500 (Martin et al., 2001). Following previous methods (Mao et al., 2016; Liu et al., 2018a, 2019d), we use as the training set the combination of 200 images from the training set and 100 images from the validation set, and test on 200 images from the test set. On this dataset, we generate noisy images by adding white Gaussian noises to clean images with $\sigma = 30, 50, 70$.

For image super-resolution experiments, we use as the training set 800 high-resolution images from DIV2K dataset and test on 5 most widely used benchmark datasets, including Set5, Set14, BSD100, Urban100 and Manga109. We carry out experiments with Bicubic degradation models. The high resolution images are respectively degraded to 1/2, 1/3 and 1/4 of original spatial resolution.

Search settings. The supernets we build for image denoising and image super-resolution are different. For image denoising, the supernet contains 3 cells, each of which consists of 4 nodes. We perform architecture search on BSD500. Specifically, we randomly choose 2% of training samples as the validation set (Validation V). The rest are equally divided into two parts: one part is used to update the kernels of convolution layers (Train W) and the other part is used to optimize the parameters of the neural architecture (Train A). For image super-resolution, we build a supernet that has 2 cells

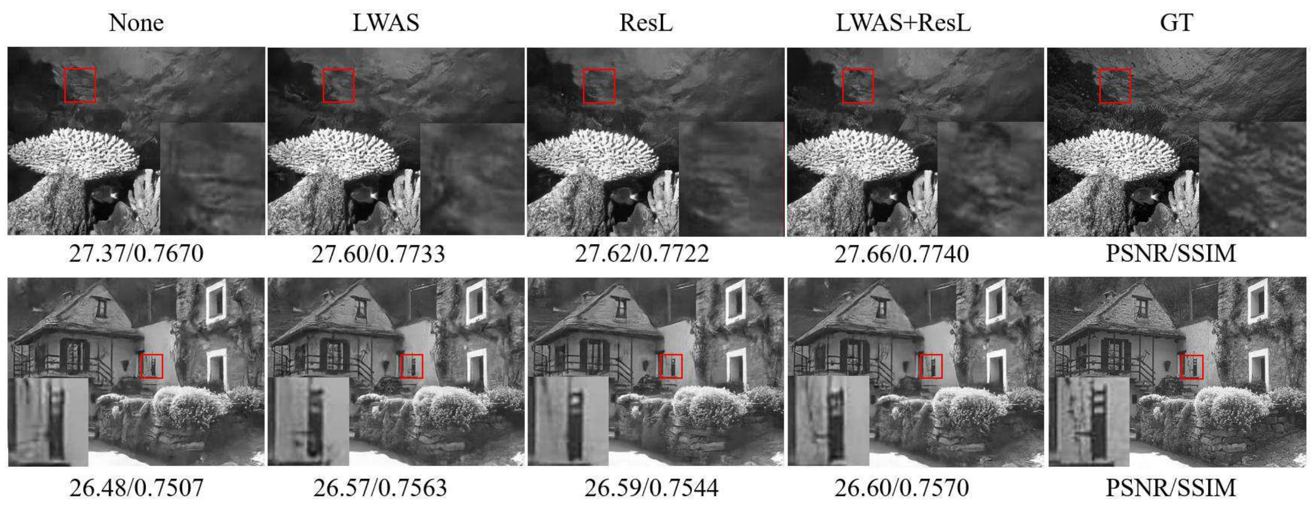

Fig. 5 Visual results of image denoising. The five columns, from left to right, are denoising results of using basic search space, using layer-wise architecture search strategy, using residual learning strategy, using both strategies and ground truths. The top and bottom rows are the denoising results for image '101027' and '207038' from BSD200 when $\sigma = 50$.

Table 1 Effects of layer-wise architecture sharing (LWAS) and residual learning (ResL) strategies for denoising

Methods	LWAS	ResL	Parameters (M)	$\sigma = 30$		$\sigma = 50$		$\sigma = 70$	
				PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
HiNAS	✗	✗	0.29	29.13	0.8356	26.77	0.7584	25.36	0.6995
HiNAS	✓	✗	0.34	29.16	0.8392	26.82	0.7647	25.45	0.7072
HiNAS	✗	✓	0.29	29.22	0.8420	26.83	0.7631	25.41	0.7074
HiNAS	✓	✓	0.34	**29.25**	**0.8420**	**26.84**	**0.7654**	**25.46**	**0.7076**

and each cell involves 3 nodes. We perform search on DIV2K. Similarly, the training samples are divided into three parts, including Validation V, Train W and Train A.

The batch size of training the supernet is set to 8 for image denoising and 24 for image super-resolution. We train the supernet at most 100 epochs and optimize the parameters of kernels and architecture with two optimizers. For learning the kernels of convolution layers, we employ the standard SGD optimizer. The momentum and weight decay are set to 0.9 and 0.0003, respectively. The learning rate decays from 0.025 to 0.001 with the cosine annealing strategy (Loshchilov and Hutter, 2017). For learning the parameters of an architecture, we use the Adam optimizer, where both learning rate and weight decay are set to 0.001. In the first 20 epochs, we only update the parameters of kernels, then we start to alternately optimize the kernels of convolution layers and architecture parameters from epoch 21.

During the training process of searching, we randomly crop patches of 64×64 and feed them to the network. During evaluation, we split each image to some adjacent patches of 64×64 and then feed them to the network and finally join the corresponding patch results to obtain final results of the whole test image. From epoch 61, we evaluate the supernet for every epoch and update the best performance record.

Training settings With the SGD optimizer, we train the denoising and super-resolution networks for 400k and 600k iterations, where the initial learning rate, batch size are set to 0.05 and 24, respectively. For data augmentation, we use random crop, random rotations $\in \{0°, 90°, 180°, 270°\}$, horizontal and vertical flipping. For random crop, the patches of 64×64 are randomly cropped from input images.

4.2 Ablation study

In this section, we focus on presenting ablation analysis on each component proposed in this paper. We first analyze the effects of layer-wise architecture sharing strategy and using residual learning; then we show the benefits of searching for outer layer width; finally we verify can our desinged loss item l_{ssim} really improves image restoration results.

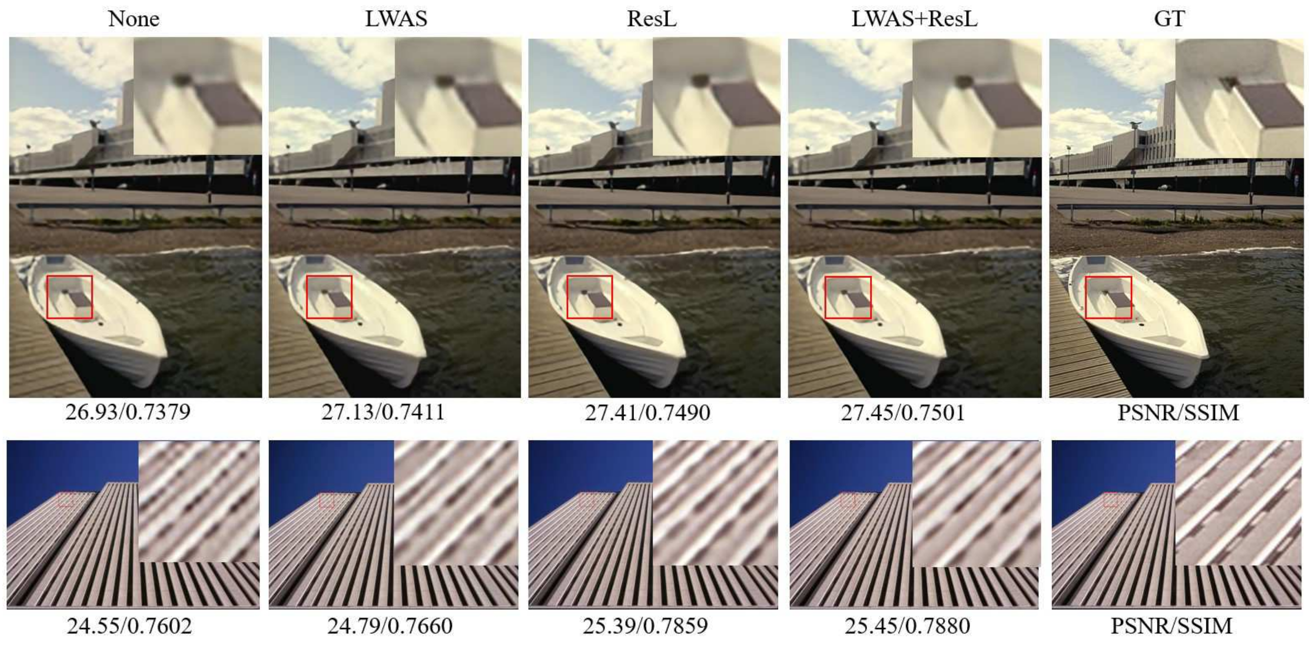

Fig. 6 Visual results of × 4 image super-resolution. The five columns, from left to right, are super-resolution results of using basic search space, using layer-wise architecture search strategy, using residual learning strategy, using both strategies and ground truths. The first row is the super-resolution results for image '78004' from BSD100. The second row is the super-resolution results for image '016' from Urban100.

Table 2 Effects of layer-wise architecture sharing (LWAS) and residual learning (ResL) strategies for 5× super-resolution.

Methods	Unshared cell	Residual learning	Set5		Set14		BSD100		Urban100	
			PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
HiNAS	✗	✗	30.48	0.8800	27.22	0.7751	26.86	0.7342	24.63	0.7503
HiNAS	✓	✗	30.89	0.8827	27.46	0.7788	27.05	0.7375	24.87	0.7566
HiNAS	✗	✓	31.66	0.8923	27.84	0.7877	27.31	0.7455	25.44	0.7759
HiNAS	✓	✓	**31.74**	**0.8928**	**27.84**	**0.7883**	**27.35**	**0.7467**	**25.52**	**0.7785**

4.2.1 Effects of layer-wise architecture sharing and residual learning

To evaluate the effects of using layer-wise architecture sharing strategy (LWAS) and using residual learning (ResL), we compare architectures founded in four different search settings. The first setting is using the basic search space designed in this paper. Based on the first setting, the second and third setting employ either layer-wise architecture sharing strategy or residual learning strategy. The last setting is using both strategies. The comparison results for image denoising and super-resolution are listed in Table 1 and Table 2, respectively. The corresponding visual results are shown in Figure 5 and Figure 6.

From Table 1 and 2, we can see that both layer-wise cell architecture sharing strategy and residual learning strategy improve the performance and architectures using both strategies obtain the best performance. For images denosing, layer-wise cell architecture sharing strategy and residual learning strategy are equally important. For instance, with $\sigma = 50$, layer-wise architecture sharing strategy improves PSNR and SSIM by 0.05 dB and 0.0063, respectively. Residual learning strategy improves PNSR and SSIM by 0.06 dB and 0.0047, respectively. Compared with layer-wise architecture sharing, residual learning strategy improves more in PSNR and less in SSIM. Using both strategies improves PNSR and SSIM from 26.77 dB and 0.7584 to 26.84 dB and 0.7654, respectively. From Figure 5, we can see that denoising results of last setting shows more details. For example, in the first line of Figure 5, the restoration image of using both strategies shows clean texture of waves.

For image super-resolution, from setting one to setting two, PSNR and SSIM show slight improvement. From setting one to setting three, PSNR and SSIM are significantly improved by using residual learning. Setting four still achieves the best performance. Taking the results on BSD100 as an example, PSNR and SSIM are improved to 27.08 dB and 0.7375 by using layer-

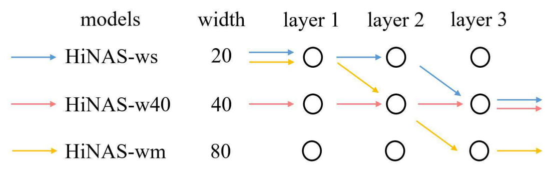

Fig. 7 Comparisons of different search settings.

Table 3 Comparisons of different search settings.

Models	# parameters (M)	PSNR	SSIM
HiNAS-ws	0.34	29.25	0.8420
HiNAS-w40	0.57	29.14	0.8409
HiNAS-wm	0.75	29.19	0.8413

wise cell architecture sharing, 27.31 dB and 0.7455 by using residual learning. Using both layer-wise cell architecture sharing and residual learning achieve the best performance, PSNR=27.35 dB and SSIM=0.7467. According to results shown in Figure 6, the produced results of the using both strategies are most close to the ground truths.

4.2.2 Benefits of searching for outer layer width

In this section, to evaluate the benefits of searching outer layer width, we apply our HiNAS on BSD500 with three different search settings, which are denoted as HiNAS-ws, HiNAS-w40, HiNAS-wm. For HiNAS-ws, both inner cell architectures and out layer width are found by our HiNAS algorithm. For the latter two settings, only the inner cell architectures are found by our algorithm and the outer layer widths are set manually. The width of each cell are set to 40 for HiNAS-w40. In HiNAS-wm, we set the basic width of the first cell to 20, then double the basic width cell by cell. The three settings are shown in Figure 7. The comparison results for denoising on BSD500 of $\sigma = 30$ are listed in Table 3.

As shown in Table 3, from HiNAS-ws to HiNAS-w40, PSNR and SSIM decrease from 29.25 dB and 0.8420 to 29.14 dB and 0.8409, respectively. In addition, the number of parameters increases from 0.34 M to 0.57 M. From HiNAS-w40 to HiNAS-wm, PSNR and SSIM show slight improvement, 0.05 dB for PSNR and 0.0004 for SSIM. Meanwhile the corresponding number of parameters is increased to 0.75 M. With searching for the outer layer width, HiNAS-ws achieves the best performance, while having the least parameters.

4.2.3 Benefits of Using $\mathrm{l_{ssim}}$ *Loss*

Here we analyze how our designed loss item $\mathrm{l_{ssim}}$ improves image restoration results. We implement two baselines: 1) HiNAS* trained with single MSE loss; and

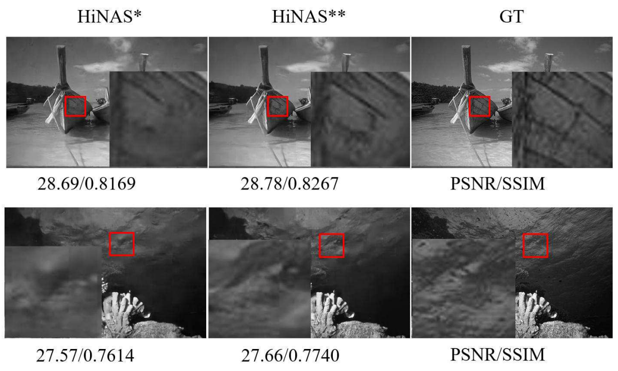

Fig. 8 Visual results of image denoising. The first row is the denoising results for image '81095' from BSD200 when $\sigma = 50$. The second row is the denoising results for '101027' from BSD200 when $\sigma = 50$.

Table 4 Benefits of using l_{ssim} loss item for image denoising. HiNAS* is trained with single loss MSE and HiNAS** is trained with the combination loss MSE and $\mathrm{l_{ssim}}$. The last row is the comparison results. Blue and red denote the performance increase and decrease over HiNAS*.

Methods	$\sigma = 30$		$\sigma = 50$		$\sigma = 70$	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
HiNAS*	29.03	0.8254	26.77	0.7498	25.42	0.6962
HiNAS**	29.14	0.8403	26.77	0.7635	25.48	0.7129
Comparison	0.11	0.0149	0.009	0.0137	0.06	0.0167

Table 5 Benefits of using l_{ssim} loss item for $\times$ 4 image super-resolution.

Methods	Set5		BSD100		Urban100	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
HiNAS*	31.72	0.8904	27.38	0.7367	25.47	0.7691
HiNAS**	31.74	0.8928	27.35	0.7467	25.52	0.7785
Comparison	0.02	0.0024	0.03	0.01	0.05	0.0094

2) HiNAS** trained with the combination MSE loss and $\mathrm{l_{ssim}}$.

The experimental results of image denoising on BSD500 dataset with $\sigma = 30$ are listed in Table 4 and shown in Figure 8, from which we can see that HiNAS** achieves better results. Comparing the denosing results in Figure 8, we can find that the denoising images of HiNAS ** reverse more rich texture information as compared with that of HiNAS*. Note that even trained with single MSE loss, HiNAS* still outperforms the competitive models, such as N3Net, NLRN, etc.

The experimental results of $\times$ 4 image super-resolution are reported in Table 5. Figure 9 shows the corresponding visual results. For image denoising, loss item l_{ssim} benefits both PSNR and SSIM metrics. However, for

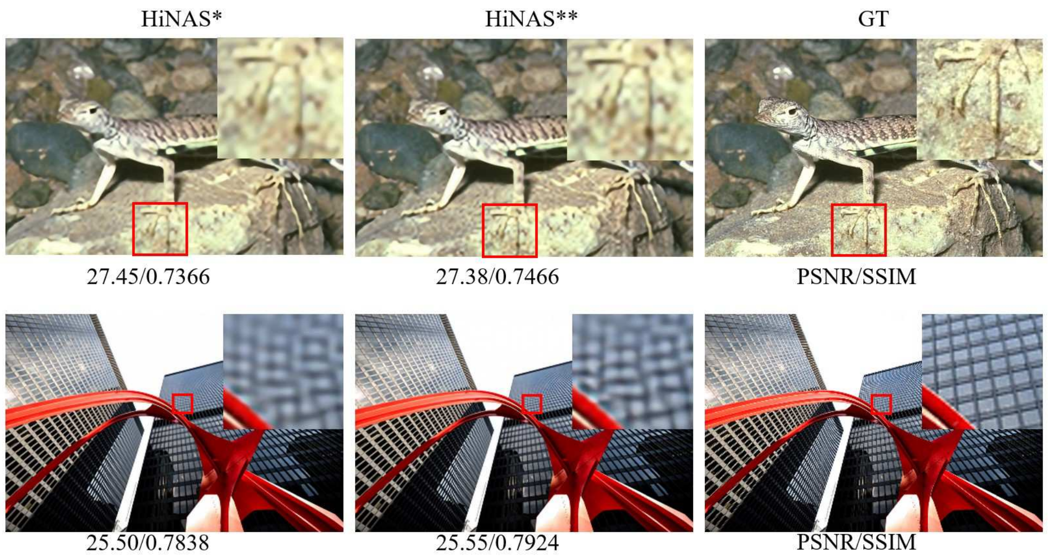

Fig. 9 Visual results of 4× image super-resolution. The two rows are the results for image '87046' from BSD100 and image '062' from Urban100.

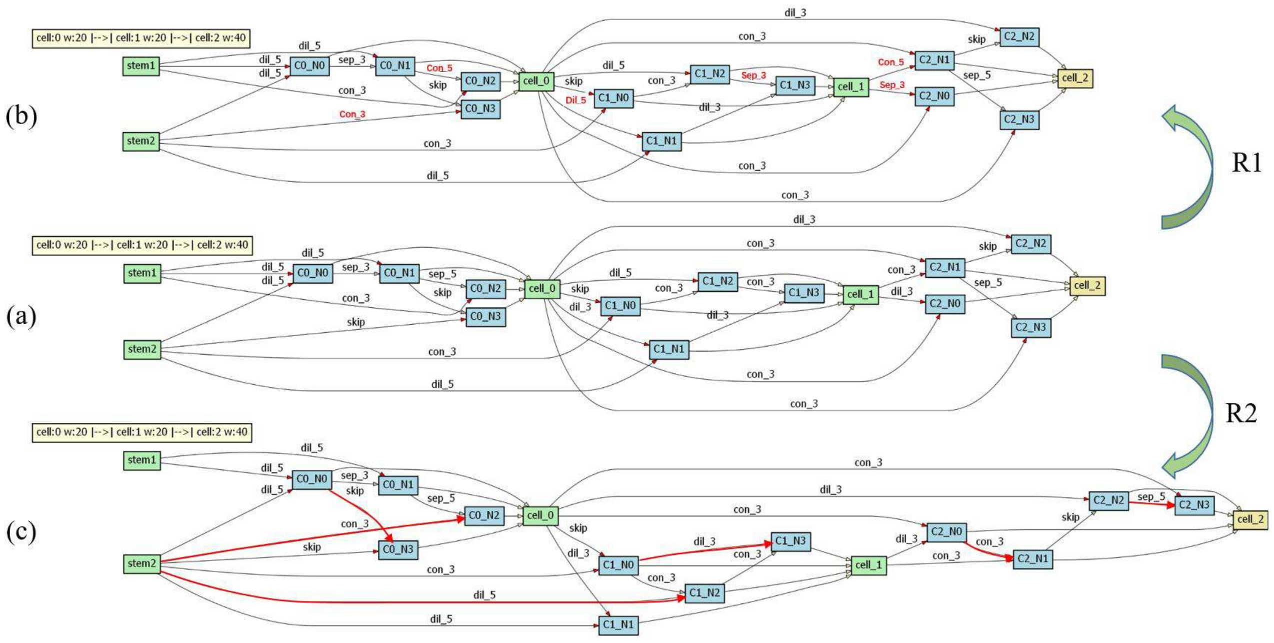

Fig. 10 Architecture analysis for image denoising. 'Conv', 'sep' and 'dil' denote conventional, separable and dilated convolutions. 'Skip' is skip connection. (a) The architecture found by HiNAS (b) modified cells, $R1$; (c) modified cells, $R2$. The operations and paths marked in red denote modification parts

image super-resolution, loss item l_{ssim} does not always improves both PSNR and SSIM. On BSD100, using loss item l_{ssim} improves SSIM by 0.01, but decreases PSNR by 0.03 dB. On Set5 and Urban100, HiNAS** trained with the combination loss shows even better results over HiNAS*. On the other dataset Set14 which is not listed in here, HiNAS** also shows better performance than HiNAS*. We conjecture that this is caused by the property of l_{ssim}, which is designed based on SSIM evaluation metric. Using l_{ssim} will improve SSIM directly and affect PSNR indirectly. The changes of SSIM and PSNR are not always consistent. Figure 9 shows two examples and the images in the first first row are come from BSD100. Comparing the middle image with the image in the left side in the first row, we can see that, HiNAS** has higher SSIM and lower PSNR compared with HiNAS*. However, from visual results, we can not see any difference. In the second row, the middle image

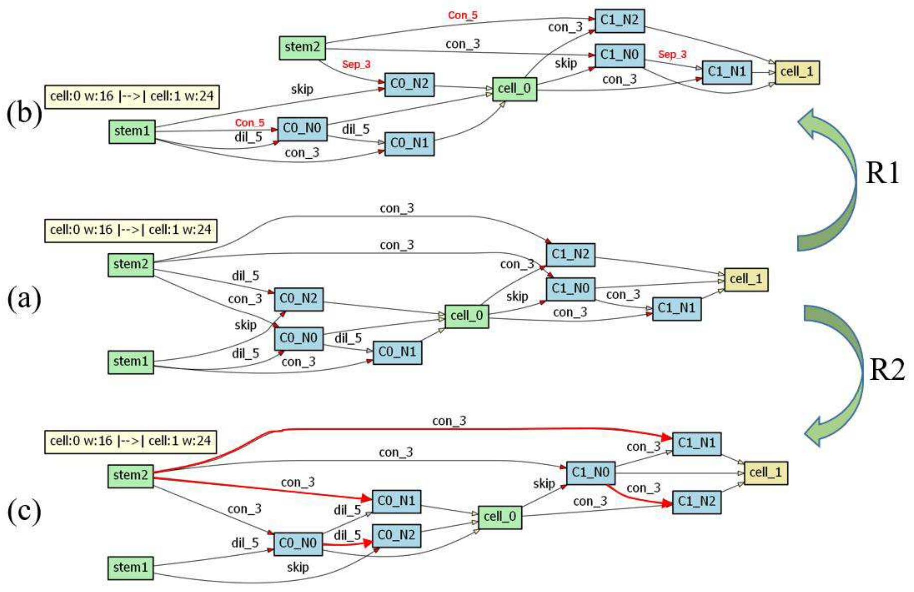

Fig. 11 Architecture analysis for image super-resolution. (a) The architecture found by HiNAS (b) modified cells, $R1$; (c) modified cells, $R2$. The operations and paths marked in red denote modification parts

Table 6 Architecture analysis for image denoising.

Methods	HiNAS	HiNAS, $R1$	HiNAS, $R2$
PSNR	**29.25**	29.21	29.12
SSIM	**0.8420**	0.8418	0.8304

shows cleaner grid structure than the image in the left side. In summary, for image super-resolution, l_{ssim} still benefits improving the performance.

Table 7 Architecture analysis for super-resolution.

Methods	Set5		BSD100		Urban100	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
HiNAS	**31.74**	**0.8928**	**27.35**	**0.7467**	**25.52**	**0.7785**
HiNAS, R1	31.70	0.8926	27.34	0.7458	25.52	0.7782
HiNAS, R2	31.58	0.8821	27.13	0.7312	25.13	0.7585

4.2.4 Architecture Analysis

The denoising and suepr-resolution architectures found by HiNAS are shown in Figure 10 (a) and Figure 11 (a), from which we can see that:

1. In the both denoising and suepr-resolution networks found by our HiNAS, the width of the last layer has the maximum number of channels. This is consist with previous manually designed networks.
2. Instead of connecting all the nodes with a certain convolution, HiNAS connects different nodes with different types of operators. We believe that these results prove that HiNAS is able to select proper operators.

From Figure 10 (a) and Figure 11 (a), we also find that the networks found by HiNAS consist of many fragmented branches, which might be the main reason why the designed networks have better performance than previous denoising models. As explained in (Ma et al., 2018), the fragmentation structure is beneficial for accuracy. Here we verify if HiNAS improves the accuracy by designing a proper architecture or by simply integrating various branch structures and convolution operations. We modify the architecture found by our HiNAS in two different ways and then compare the modified architectures with unmodified architectures.

The first modification is replacing conventional convolutions in the searched architectures with other convolutions to verify is the operation that HiNAS selected for each path is irreplaceable, as shown in Figure 10 (b) and Figure 11 (b). The other modification is to change the topological structure inside each cell, as shown in Figure 10 (c) Figure 11 (c), which is aiming to verify if the connection relationship built by our HiNAS is indeed appropriate. Following the two proposed modifications, we modify different operations and connections in different nodes. Modified architectures achieve lower performance. However, limited by space, we only show two examples here. The modification parts are marked in red. The comparison results are listed in Tables 6 and 7, where the two mentioned modifica-

Methods	cell sharing	# param. (M)	$\sigma = 30$		$\sigma = 50$		$\sigma = 70$		search cost		search method
			PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	GPU	hours	
E-CAE	-	1.10	28.23	0.8047	26.17	0.7255	24.83	0.6636	4 V100	44.0	EA
HiNAS	✗	0.34	29.21	0.8405	26.77	0.7628	25.41	0.7053	1 1080Ti	3.6	gradient
HiNAS	✓	**0.34**	**29.25**	**0.8420**	**26.84**	**0.7654**	**25.46**	**0.7076**	1 1080Ti	1.0	gradient

Table 8 Comparisons with NAS methods of image denoising. For E-CAE, the search and training time costs computed on V100 GPUs are provided by authors.

Methods	cell sharing	# param. (M)	Set5		BSD100		Urban100		search cost		search method
			PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	GPU	hours	
MoreMNAS	-	1.04	37.63	0.9584	31.95	0.8961	31.24	0.9187	8 V100	168	RL + EA
FALSR	-	0.41	37.66	0.9586	31.96	0.8965	31.24	0.9187	8 V100	72	RL + EA
HiNAS	✗	0.34	**37.68**	0.9599	**31.98**	**0.9036**	31.27	0.9229	1 1080Ti	12.6	gradient
HiNAS	✓	**0.34**	37.66	**0.9608**	31.98	0.9035	**31.28**	**0.9237**	1 1080Ti	3.5	gradient

Table 9 Comparisons with NAS methods of image super-resolution.

tion operations, are denoted as $R1$ and $R2$. From Tables 6 and 7, we can see that both modifications reduce the accuracy for image denoising and super-resolution. Replacing convolution operations slight reduces PSNR and SSIM. Compared with replacing convolution operations, changing topological structures is more influential on the performance.

From the comparison results, we can draw a conclusion: HiNAS does find a proper structure and select proper convolution operations, instead of simply integrating a complex network with various operations. *The fact that a slight perturbation to the found architecture deteriorates the accuracy indicates that the found architecture is indeed a local optimum in the architecture search space.*

4.3 Comparisons with other NAS methods

Inspired by recent advances in NAS, four NAS methods have been proposed for low-level image restoration tasks (Suganuma et al., 2018; Chu et al., 2019a; Liu et al., 2019c). E-CAE (Suganuma et al., 2018) is proposed for image inpainting and denoising. MoreMNAS (Chu et al., 2019b) and FALSR (Chu et al., 2019a) are proposed for super resolution. EvoNet (Liu et al., 2019c) searches for networks for medical image denoising. In this section, we first compare our HiNAS with E-CAE for searching for architectures for image denoising on BSD500; then we compare our HiNAS with MoreMNAS and FALSR for searching super-resolution networks on DIV2K. Tables 8 and 9 show the details.

Compared with previous NAS methods of low-level tasks, our HiNAS is much faster in searching. For instance, by using 8 Tesla V100 GPUs, FALSR takes about 72 hours to find the best super-resolution architecture on DIV2K and MoreMNAS takes 168 hours. In contrast to them, HiNAS only takes about 3.5 hours with one GTX 1080Ti GPU. The fast search speed of our HiNAS benefits from the following three advantages.

1. HiNAS uses a gradient based search strategy. E-CAE, MoreMNAS and FALSR need to train a large number of children networks (genes) to update their populations or controllers . For instance, FALSR trained about 10k models during its searching process. In sharp contrast, our HiNAS only needs to train one supernet in the search stage.
2. In searching for the outer layer width, we share cells across different feature levels, saving memory consumption in the supernet. As a result, we can use larger batch sizes for training the supernet, which further speeds up search. Comparing the last two rows of Tables 8 and 9, we can see that the proposed cell sharing strategy significantly accelerates the search speed without any negative influence to performance.

4.4 Comparisons with state-of-the-art methods

Now we compare the HiNAS designed networks with a number of recent methods and use PSNR and SSIM to quantitatively measure the restoration performance of those methods. Table 10 shows the comparison results of denoising on BSD500 and Figure 12.

Table 10 shows that N3Net and HiNAS beat other models by a clear margin. Our proposed HiNAS achieves the best performance when σ is set to 50 and 70. When the noise level σ is set to 30, the SSIM of NLRN is slightly higher (0.002) than that of our HiNAS, but the PSNR of NLRN is much lower (nearly 1dB) than that of HiNAS.

Overall our HiNAS achieves better performance than others. In addition, compared with the second best model

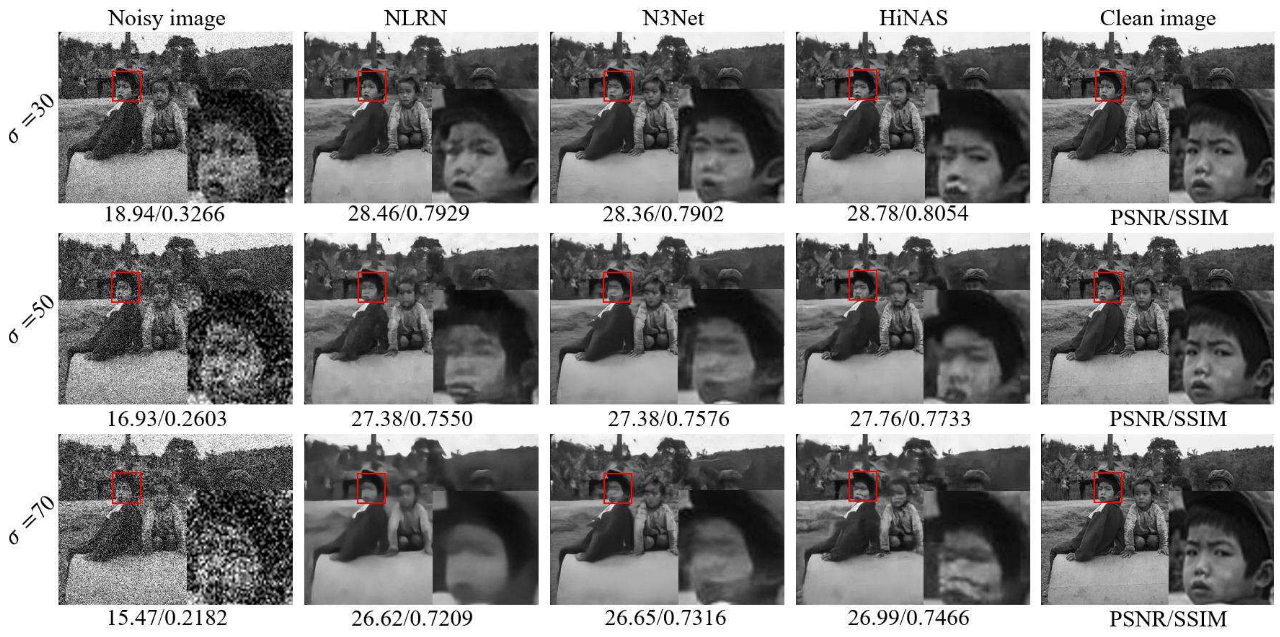

Fig. 12 Visual denoising results of NLRN, N3Net and HiNAS. The first row is the denoising results of image '15011' from BSD 200 when $\sigma = 30$. The second row and the third row are the denoising results of $\sigma = 50$ and $\sigma = 70$.

Table 10 Denoising experiments. Comparisons with state-of-the-arts on the BSD500 dataset. We show our results in the last row. Time cost means GPU-seconds for inference on the 200 images from the test set of BSD500 using one single GTX 1080Ti graphic card. Best and second best results are marked in red and blue.

Methods	# parameters (M)	time cost (s)	$\sigma = 30$		$\sigma = 50$		$\sigma = 70$	
			PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
BM3D	-	-	27.31	0.7755	25.06	0.6831	23.82	0.6240
WNNM	-	-	27.48	0.7807	25.26	0.6928	23.95	0.3460
RED	-	-	27.95	0.8056	25.75	0.7167	24.37	0.6551
MemNet	0.68	32.67	28.04	0.8053	25.86	0.7202	24.53	0.6608
NLRN	1.03	6940.67	28.15	0.8423	25.93	0.7214	24.58	0.6614
E-CAE	1.10	-	28.23	0.8047	26.17	0.7255	24.83	0.6636
DBSN	6.61	127.56	-	-	26.18	0.7257	-	-
DuRN-P	0.82	-	28.50	0.8156	26.36	0.7350	25.05	0.6755
N3Net	0.71	80.74	28.66	0.8220	26.50	0.7490	25.18	0.6960
HiNAS	0.34	18.67	29.25	0.8420	26.84	0.7654	25.46	0.7076

N3Net, the network designed by HiNAS has fewer parameters and is faster in inference. As listed in Table 10, the HiNAS designed network has 0.34M parameters, which is 47.89% that of N3Net and 30.91% that of E-CAE. Compared with N3Net, the HiNAS designed network reduces the inference time on the test set of BSD500 by **76.88%**.

The comparison results of × 4 super-resolution experiments are listed in Table 9 and more super-resolution experimental results of × 2, × 3 and × 4 are listed in Table 12. Compared with parameter heavy models which have more than 10 M parameters, HiNAS achieves lower PSNR and highly competitive SSIM, while having much faster inference speed and much fewer parameters. For instance, the network designed by HiNAS has only 0.33 M parameters, which is about **1/47** that of SAN. The inference speed is **20.97** × that of SAN. Compared with fast super-resolution models, HiNAS achieves equal performance, while having faster inference speed. FSRCNN(KD), the method proposed in ECCV of this year leverage distillation to improve the performance of FSRCNN. This method has fewer parameters and faster inference speed compared with our HiNAS. However, the PSNR and SSIM of FSRCNN(KD) is much lower than ours and that of other fast super-resolution methods. So, we believe HiNAS is still comparable compared with FSRCNN(KD) when taking both inference speed and performance into consideration. Two examples are presented in Figure 13, from which we can draw two conclusions: 1) all three CNN based super-resolution

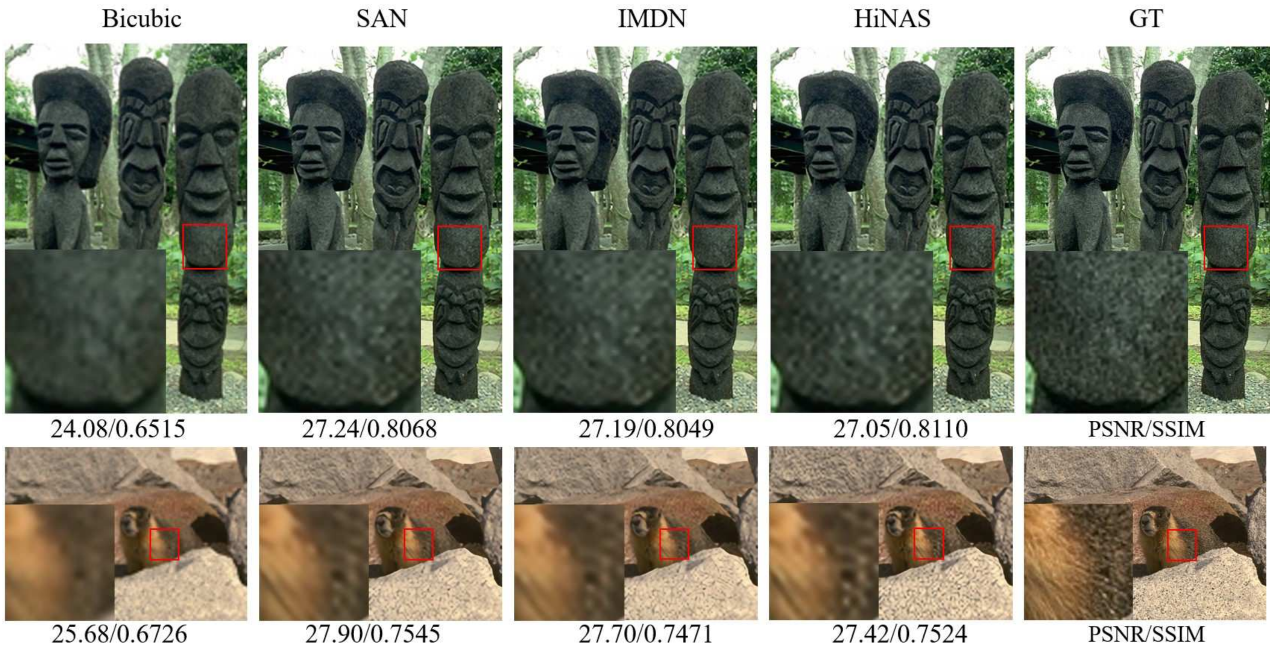

Fig. 13 Visual super-resolution results of Bicubic, SAN, IMDN and HiNAS. First row: the $\times$ 2 super-resolution results for image '101085' from BSD100. Second row: the $\times$ 4 super-resolution results for image '41069' from BSD100.

Table 11 Super-resolution experiments when scale factor is 3. We show our results in the last row. Time cost means GPU-seconds for inference on the 100 images from BSD100 using one single GTX 1080 Ti graphic card. Best and second best results are marked in red and blue.

Methods	Param. (M)	time cost (s)	Set5		Set14		BSD100		Urban100	
			PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
EDSR	43.00	-	34.65	0.9280	30.52	0.8462	29.25	0.8093	28.80	0.8653
SRMD	1.50	-	34.12	0.9254	30.04	0.8382	28.97	0.8025	27.57	0.8398
RDN	22.30	11.89	34.71	0.9296	30.57	0.8468	29.26	0.8093	28.80	0.8653
RCAN	16.00	14.60	34.74	0.9299	30.64	0.8481	29.32	0.8111	29.08	0.8702
SAN	15.70	34.59	34.75	0.9300	30.59	0.8476	29.33	0.8112	28.93	0.8671
RFANet	11.00	-	34.79	0.9300	30.67	0.8487	29.34	0.8115	29.1	0.872
bicubic	-	-	30.39	0.8682	27.55	0.7742	27.21	0.7385	24.46	0.7349
FSRCNN(KD)	0.013	0.58	33.31	0.9179	29.57	0.8276	28.61	0.7919	26.67	0.8153
VDSR	0.67	-	33.67	0.9210	29.78	0.8320	28.83	0.7990	27.14	0.8290
LapSRN	0.50	2.94	33.82	0.9227	29.87	0.8320	28.82	0.7980	27.07	0.8280
MemNet	0.68	7.37	34.09	0.9248	30.01	0.8350	28.96	0.8001	27.56	0.8376
IDN	0.55	2.90	34.11	0.9253	29.99	0.8354	28.95	0.8013	27.42	0.8359
IMDN	0.70	1.91	34.36	0.9270	30.32	0.8417	29.09	0.8046	28.17	0.8519
HiNAS	0.30	1.65	33.98	0.9270	29.62	0.8472	28.87	0.8141	27.51	0.8465

models achieve significantly better visual results than bicubic; 2) slightly higher PSNR can not guarantee plausible visual results. In the first row, both SAN and IMDN achieve higher PSNR compared to HiNAS, but we can not see any difference from the results. In the second row, the PSNR and SSIM of the result of SAN are slightly higher than that of HiNAS, while the visual results of SAN and HiNAS are still very close and it is hard to say which one is better. In summary, in terms of visual results, HiNAS is highly competitive with other state-of-the-art methods.

4.5 Trade-off between inference speed and performance

We visualize in Figure 14 the performance comparison of our HiNAS and the state of the art in terms of the inference speed, SSIM and the number of parameters. For image denoising, our HiNAS achieves huge advanteage compared with other denoising networks. For image super-resolution, our HiNAS achieves the best trade-off between inference speed and performance. In terms of the inference speed, our HiNAS is only slower than FSRCNN (KD), but the performance of HiNAS

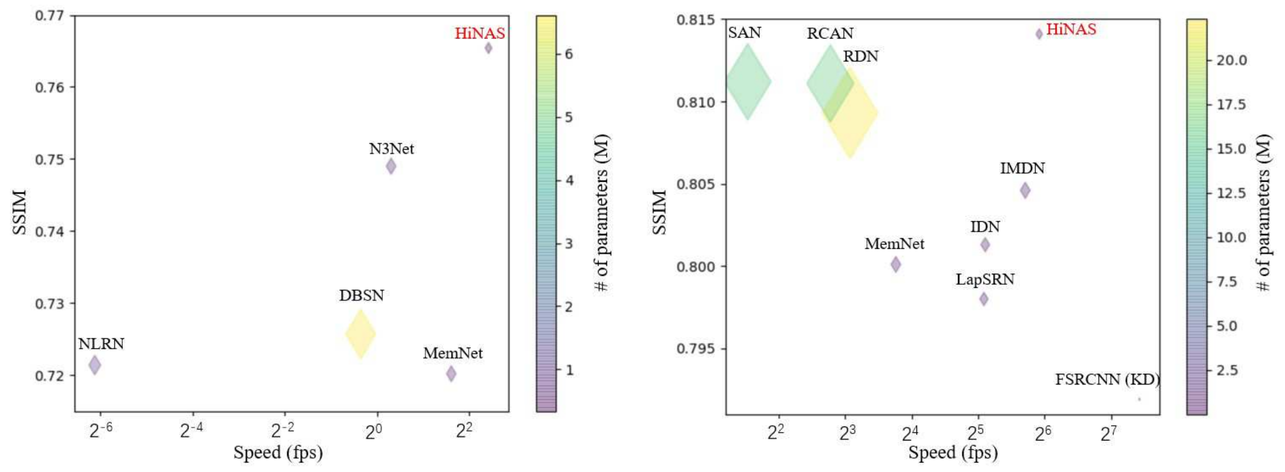

Fig. 14 Trade-off between inference speed and performance. (a) denoising results of test on BSD200 when $\sigma = 50$. (b) $\times$ 3 super-resolution results of test on BSD100. Marker size and color encode the numnber of parameters. HiNAS shows superior performance compared with a few recent methods.

are much better than that of FSRCNN (KD). Compared with other fast super-resolution networks, our HiNAS achieve higher SSIM, while having much faster inference speed and fewer parameters.

5 Conclusion

In this study, we propose HiNAS, an attempt of using NAS to automatically design effective network architectures for two low-level image restoration tasks, image denosing and super-resolution. HiNAS builds a hierarchical search space including inner search space and outer search space, which are able to search inner cell topological structures and outer layer widths, respectively. We propose layer-wise architecture sharing strategy to improve the flexibility of search space, improving performance. We propose cell sharing strategy to save memory, accelerating search process. Finally, instead of searching for networks to learning the restoration result directly, our HiNAS design networks to learning the residual information between low quality image and high quality image. HiNAS is both memory and computation efficient, taking only less than 1/6 day to search using a single GPU. Extensive experimental results demonstrate that the proposed HiNAS achieves highly competitive performance compared with state-of-the-art models, while having fewer parameters and faster inference speed. We believe that the proposed method can be applied to many other low-level image processing tasks.

References

Ahn N, Kang B, Sohn KA (2018) Fast, accurate, and lightweight super-resolution with cascading residual network. In: Proc. Eur. Conf. Comp. Vis., pp 252–268

Cai H, Wang T, Wu Z, Wang K, Lin J, Han S (2019) On-device image classification with proxyless neural architecture search and quantization-aware fine-tuning. In: Proc. IEEE Int. Conf. Comp. Vis., pp 0–0

Chatterjee P, Milanfar P (2009) Is denoising dead? IEEE Trans Image Process 19(4):895–911

Chu X, Zhang B, Ma H, Xu R, Li J, Li Q (2019a) Fast, accurate and lightweight super-resolution with neural architecture search. arXiv: Comp Res Repository abs/1901.07261

Chu X, Zhang B, Xu R, Ma H (2019b) Multi-objective reinforced evolution in mobile neural architecture search. arXiv: Comp Res Repository abs/1901.01074

Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3-d transform-domain collaborative filtering. IEEE Trans Image Process 16(8):2080–2095

Dai T, Cai J, Zhang Y, Xia ST, Zhang L (2019) Second-order attention network for single image super-resolution. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 11065–11074

Dong C, Loy CC, He K, Tang X (2015) Image super-resolution using deep convolutional networks. TPAMI 38(2):295–307

Dong W, Zhang L, Shi G, Li X (2012) Nonlocally centralized sparse representation for image restoration. IEEE Trans Image Process 22(4):1620–1630

Elsken T, Metzen JH, Hutter F (2018) Efficient multi-objective neural architecture search via lamar-

Table 12 Super-resolution experiments.

Methods	Scale	Param.(M)	time costs (s)	Set5		Set14		BSD100		Urban100	
				PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
EDSR	× 2	43.00		38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351
SRMD	× 2	1.50		37.79	0.9601	33.32	0.9159	32.05	0.8985	31.33	0.9204
DBPN	× 2	10.05		38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324
RDN	× 2	22.30		38.24	0.9614	34.01	0.9212	32.34	0.9017	32.89	0.9353
RCAN	× 2	16.00		38.27	0.9614	34.11	0.9216	32.41	0.9026	33.34	0.9384
SAN	× 2	15.70		38.31	0.9620	34.07	0.9213	32.42	0.9028	33.10	0.9370
RFANet	× 2	11.00		38.26	0.9615	34.16	0.9220	32.41	0.9026	33.33	0.9026
bicubic	× 2	-	-	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403
FSRCNN(KD)	× 2	0.013	0.67	37.33	0.9576	32.79	0.9105	31.65	0.8926	30.24	0.9071
VDSR	× 2	0.67		37.53	0.9590	33.05	0.9130	31.90	0.8960	30.77	0.9140
MemNet	× 2	0.68	8.50	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195
FALSR-B	× 2	0.33	-	37.61	0.9582	33.29	0.9143	31.97	0.8967	31.28	0.9191
IDN	× 2	0.55	5.62	37.83	0.9600	33.30	0.9148	32.08	0.8985	31.27	0.9196
IMDN	× 2	0.69	2.47	38.00	0.9605	33.63	0.9177	32.19	0.8996	32.17	0.9283
HiNAS	× 2	0.28	2.13	37.66	0.9608	32.98	0.9193	31.98	0.9035	31.28	0.9237
SPSR	× 4	24.80	16.47	30.40	0.8627	26.64	0.7930	25.51	0.6576	29.41	0.8537
EDSR	× 4	43.00	-	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033
SRMD	× 4	1.50	-	31.96	0.8925	28.35	0.7787	27.49	0.7337	25.68	0.7731
DBPN	× 4	10.00	-	32.47	0.8980	28.82	0.7860	27.72	0.7400	26.38	0.7946
RDN	× 4	22.30	10.71	32.47	0.8990	28.81	0.7871	27.72	0.7419	26.61	0.8028
RCAN	× 4	16.00	13.12	32.62	0.9001	28.86	0.7888	27.76	0.7435	26.82	0.8087
SAN	× 4	15.70	31.03	32.64	0.9003	28.92	0.7888	27.78	0.7436	26.79	0.8068
RFANet	× 4	11.00	-	32.66	0.9004	28.88	0.7894	27.79	0.7442	26.92	0.8112
bicubic	× 4	-	-	28.42	0.8104	26.00	0.7027	25.96	0.6675	23.14	0.6577
FSRCNN (KD)	× 4	0.013	0.54	30.95	0.8759	27.77	0.7615	27.08	0.7188	24.82	0.7393
VDSR	× 4	0.67	-	31.35	0.8830	28.02	0.7680	27.29	0.0726	25.18	0.7540
LapSRN	× 4	0.50	3.68	31.54	0.8852	28.09	0.7700	27.32	0.7275	25.21	0.7562
MemNet	× 4	0.68	6.86	31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630
IDN	× 4	0.55	2.47	31.82	0.8903	28.25	0.7730	27.41	0.7297	25.41	0.7632
IMDN	× 4	0.72	1.72	32.21	0.8948	28.58	0.7811	27.56	0.7353	26.04	0.7838
HiNAS	× 4	0.33	1.48	31.74	0.8928	27.84	0.7883	27.35	0.7467	25.52	0.7785

ckian evolution. arXiv: Comp Res Repository abs/1804.09081

Ghiasi G, Lin TY, Le Q (2019) NAS-FPN: Learning scalable feature pyramid architecture for object detection. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 7036–7045

Gu S, Zhang L, Zuo W, Feng X (2014) Weighted nuclear norm minimization with application to image denoising. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 2862–2869

Guo S, Yan Z, Zhang K, Zuo W, Zhang L (2019) Toward convolutional blind denoising of real photographs. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 1712–1722

He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. In: Proc. Eur. Conf. Comp. Vis., Springer, pp 630–645

Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 7132–7141

Hui Z, Wang X, Gao X (2018) Fast and accurate single image super-resolution via information distillation network. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 723–731

Hui Z, Gao X, Yang Y, Wang X (2019) Lightweight image super-resolution with information multi-distillation network. In: Proc. ACM Int. Conf. Multimedia, pp 2024–2032

Jaroensri R, Biscarrat C, Aittala M, Durand F (2019) Generating training data for denoising real rgb images via camera pipeline simulation. arXiv: Comp Res Repository abs/1904.08825

Kim J, Kwon Lee J, Mu Lee K (2016a) Accurate image super-resolution using very deep convolutional networks. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 1646–1654

Kim J, Kwon Lee J, Mu Lee K (2016b) Deeply-recursive convolutional network for image super-resolution. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 1637–1645

Lan X, Roth S, Huttenlocher D, Black M (2006) Efficient belief propagation with learned higher-order Markov random fields. In: Proc. Eur. Conf. Comp. Vis., Springer, pp 269–282

Lee W, Lee J, Kim D, Ham B (2020) Learning with privileged information for efficient image super-resolution. arXiv: Comp Res Repository abs/2007.07524

Liang H, Zhang S, Sun J, He X, Huang W, Zhuang K, Li Z (2019) Darts+: Improved differentiable architecture search with early stopping. arXiv: Comp Res Repository abs/1909.06035

Liu C, Chen LC, Schroff F, Adam H, Hua W, Yuille AL, Fei-Fei L (2019a) Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 82–92

Liu D, Wen B, Fan Y, Loy CC, Huang TS (2018a) Non-local recurrent network for image restoration. In: Proc. Advances in Neural Inf. Process. Syst., pp 1673–1682

Liu H, Simonyan K, Vinyals O, Fernando C, Kavukcuoglu K (2018b) Hierarchical representations for efficient architecture search

Liu H, Simonyan K, Yang Y (2019b) Darts: Differentiable architecture search

Liu J, Zhang W, Tang Y, Tang J, Wu G (2020) Residual feature aggregation network for image super-resolution. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 2359–2368

Liu P, El Basha M, Li Y, Xiao Y, Sanelli P, Fang R (2019c) Deep evolutionary networks with expedited genetic algorithms for medical image denoising. Medical Image Analysis 54:306–315

Liu X, Suganuma M, Sun Z, Okatani T (2019d) Dual residual networks leveraging the potential of paired operations for image restoration. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 7007–7016

Loshchilov I, Hutter F (2017) Sgdr: Stochastic gradient descent with warm restarts. In: Proc. Int. Conf. Learn. Representations

Ma N, Zhang X, Zheng HT, Sun J (2018) Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proc. Eur. Conf. Comp. Vis., pp 116–131

Mao X, Shen C, Yang YB (2016) Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: Proc. Advances in Neural Inf. Process. Syst., pp 2802–2810

Martin D, Fowlkes C, Tal D, Malik J, et al. (2001) A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. IEEE Int. Conf. Comp. Vis., pp 416–423

Nekrasov V, Chen H, Shen C, Reid I (2019) Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 9126–9135

Pham H, Guan MY, Zoph B, Le QV, Dean J (2018) Efficient neural architecture search via parameter sharing

Shi W, Caballero J, Huszár F, Totz J, Aitken AP, Bishop R, Rueckert D, Wang Z (2016) Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 1874–1883

Suganuma M, Ozay M, Okatani T (2018) Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search. In: Proc. Int. Conf. Mach. Learn.

Tai Y, Yang J, Liu X, Xu C (2017) Memnet: A persistent memory network for image restoration. In: Proc. IEEE Int. Conf. Comp. Vis., pp 4539–4547

Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le Q (2019) Mnasnet: Platform-aware neural architecture search for mobile. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 2820–2828

Tobias Plötz SR (2018) Neural nearest neighbors networks. In: Proc. Advances in Neural Inf. Process. Syst., pp 1673–1682

Wang N, Gao Y, Chen H, Wang P, Tian Z, Shen C (2020) NAS-FCOS: Fast neural architecture search for object detection. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn.

Wang Z, Bovik A, Sheikh H, Simoncelli E (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612

Wu B, Dai X, Zhang P, Wang Y, Sun F, Wu Y, Tian Y, Vajda P, Jia Y, Keutzer K (2019) Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 10734–10742

Zhang C, Ren M, Urtasun R (2018a) Graph hypernetworks for neural architecture search. arXiv: Comp Res Repository abs/1810.05749

Zhang K, Zuo W, Chen Y, Meng D, Zhang L (2017a) Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Trans Image Process 26(7):3142–3155

Zhang K, Zuo W, Gu S, Zhang L (2017b) Learning deep CNN denoiser prior for image restoration. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 3929–3938

Zhang K, Zuo W, Zhang L (2018b) FFDNet: Toward a fast and flexible solution for CNN-based image denoising. IEEE Trans Image Process 27(9):4608–4622

Zhang K, Zuo W, Zhang L (2018c) Learning a single convolutional super-resolution network for multiple degradations. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 3262–3271

Zhang Y, Li K, Li K, Wang L, Zhong B, Fu Y (2018d) Image super-resolution using very deep residual channel attention networks. In: Proc. Eur. Conf. Comp. Vis., pp 286–301

Zhang Y, Tian Y, Kong Y, Zhong B, Fu Y (2018e) Residual dense network for image super-resolution. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 2472–2481

Zhong Z, Yan J, Wu W, Shao J, Liu CL (2018) Practical block-wise neural network architecture generation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 2423–2432

Zoph B, Le QV (2017) Neural architecture search with reinforcement learning

Zoph B, Vasudevan V, Shlens J, Le QV (2018) Learning transferable architectures for scalable image recognition. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn., pp 8697–8710