Searching Efficient Model-guided Deep Network for Image Denoising

Qian Ning,[1] Weisheng Dong,[1*] Xin Li,[2] Jinjian Wu,[1] Leida Li,[1] Guangming Shi[1]

[1]School of Artificial Intelligence, Xidian University   [2]West Virginia University

Abstract

Neural architecture search (NAS) has recently reshaped our understanding on various vision tasks. Similar to the success of NAS in high-level vision tasks, it is possible to find a memory and computationally efficient solution via NAS with highly competent denoising performance. However, the optimization gap *between the super-network and the sub-architectures has remained an open issue in both low-level and high-level vision. In this paper, we present a novel approach to filling in this gap by connecting model-guided design with NAS (MoD-NAS) and demonstrate its application into image denoising. Specifically, we propose to construct a new search space under model-guided framework and develop more stable and efficient differential search strategies. MoD-NAS employs a highly reusable width search strategy and a densely connected search block to automatically select the operations of each layer as well as network width and depth via gradient descent. During the search process, the proposed MoG-NAS is capable of avoiding* mode collapse *due to the smoother search space designed under the model-guided framework. Experimental results on several popular datasets show that our MoD-NAS has achieved even better PSNR performance than current state-of-the-art methods with fewer parameters, lower number of flops, and less amount of testing time.*

1. Introduction

The field of image restoration, especially image denoising, has advanced rapidly in recent years. Many deep learning-based methods have achieved great performance in image denoising applications such as Trainable Non-linear Reaction Diffusion Network (TNRD) [3], Denoising Convolutional Neural Network (DnCNN) [42], Memory Network for Image Restoration (MemNet) [33], Non-local recurrent network (NLRN) [19], Evolutionary search for convolutional autoencoders (E-CAE) [32], Dual residual networks (DuRN) [22], and Neural nearest neighbors networks (N3Net) [28]. Most recently, neural architecture search (NAS) based approach [40] has been proposed and

demonstrated highly competitive performance for the task of image denoising.

Despite the extensive study of NAS in computer vision community, the main-stream applications of NAS have been limited to middle-to-high level vision tasks such as image classification [48, 21], semantic segmentation [18], and object detection [4, 34]. Only a handful of works on leveraging NAS to low-level vision tasks have been published so far (e.g., image superresolution [12, 5] and denoising [32, 40]). For example, hierarchical NAS has been studied for various low-level vision tasks such as image denoising [40] and superresolution [12]; however, they have adopted a similar search space to that in high-level vision tasks [21]. Similar to DARTS [21], those NAS methods[40, 12] for low-level tasks have only searched the operations of layers, ignoring flexibility in terms of network width and depth. Meanwhile, existing NAS strategies are known for suffering from the problem of instability, which is believed to arise from the notorious "optimization gap" between the super-network and its sub-architectures [36].

The motivation behind this work is mainly two-fold. On one hand, inspired by recent works of model-guided network for image restoration [8, 41], we propose to construct a *new search space* by leveraging the domain knowledge implied in model-guided neural architecture. To the best of our knowledge, this is the first work incorporating the domain knowledge of image restoration into NAS to design a new search space for low-level tasks. On the other hand, recent works on densely connected search space [10] and network pruning [24] have motivated us to pursue more stable and flexible NAS strategies specifically tailored for the new constructed search spaces. Unlike previous NAS methods [40, 12, 21] that only searched the operations of each layer, our proposed method can automatically select not only the operations of each layer but also the network width and depth, achieving the objective of efficiency. The key contributions of this paper are summarized as follows.

- We propose to search efficient model-guided deep networks for image denoising. Our approach consists of a new constructed search space and a tailored search strategy for selecting network width and depth automatically, achieving lightweight and low inferring time

---
*Corresponding author (wsdong@mail.xidian.edu.cn).

simultaneously. The new search space is built under model-guided framework, in which the deep denoiser is based on U-net. The construction of search space by model-guided design guarantees the stability of our method, avoiding mode collapse during the search process.

- Selecting operations of each layer as well as the network width and depth is based on more flexible search strategies. By combining highly reusable width-search with densely connected blocks for depth search, efficient and effective networks have been searched for image denoising and compression artifact reduction.

- We have conducted different benchmark experiments on our searched network for image denoising and compression artifact reduction. Experimental results on several popular datasets show that our MoD-NAS performs comparably or even better than current state-of-the-art methods with fewer parameters, lower number of flops, and less amount of running time.

2. Related Works

2.1. Model-guided design for image denoising

The problem of image denoising can be formulated as $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$, where $\boldsymbol{x}, \boldsymbol{y}$ denotes the clean/noisy image pair and $\boldsymbol{n}$ denotes the additive white Gaussian noise. Based on the above model, we can obtain the clean image $\boldsymbol{x}$ by Maximum a Posterior (MAP) estimation as

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}} \log P(\boldsymbol{x}|\boldsymbol{y}) = \underset{\boldsymbol{x}}{\operatorname{argmax}} \log P(\boldsymbol{y}|\boldsymbol{x}) + \log P(\boldsymbol{x}), \quad (1)$$

where $P(\boldsymbol{y}|\boldsymbol{x})$ and $P(\boldsymbol{x})$ denote Gaussian likelihood and prior terms respectively, which can be expressed as

$$P(\boldsymbol{y}|\boldsymbol{x}) \propto exp(-\frac{1}{\sigma_n^2}||\boldsymbol{y} - \boldsymbol{x}||_2^2) \quad (2)$$

$$P(\boldsymbol{x}) \propto exp(-\lambda \boldsymbol{R}(\boldsymbol{x})), \quad (3)$$

where σ_n^2 is the noise variance and $\boldsymbol{R}(\boldsymbol{x})$ is the regularization function (e.g., sparsity-based [14], nonlocal self-similarity based [9, 6, 43]). As mentioned in [43, 8], we can solve image denoising problem by leveraging a deep denoising network as a plug-and-play prior [44]. Compared with hand-crafted priors, deep denoising priors achieve a more complex and flexible effect. With a mass of training data, deep network attempts to learn a nonlinear mapping function $\boldsymbol{v} = f_{DN}(\boldsymbol{x})$ [8], which can be used as a deep denoising prior as follows

$$P(\boldsymbol{x}) \propto exp(-\lambda||\boldsymbol{x} - \boldsymbol{v}||_2^2), \text{ where } \boldsymbol{v} = f_{DN}(\boldsymbol{x}). \quad (4)$$

Therefore, Eq. (1) can be rewritten as

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{\sigma_n^2}||\boldsymbol{y} - \boldsymbol{x}||_2^2 + \lambda||\boldsymbol{x} - \boldsymbol{v}||_2^2, \quad (5)$$

where $\boldsymbol{v} = f_{DN}(\boldsymbol{x})$ and λ denotes the regularization parameter. Then, Eq. (5) can be solved by iterative regularization [27] with a relaxation parameter $\delta^{(t)} = \frac{1}{1+\lambda^{(t)}\sigma_n^2}$,

$$\boldsymbol{v}^{(t)} = f_{DN}^{(t)}(\boldsymbol{x}^{(t-1)}) \quad (6a)$$

$$\boldsymbol{x}^{(t)} = \frac{\boldsymbol{y} + \lambda^{(t)}\sigma_n^2 \boldsymbol{v}^{(t)}}{1 + \lambda^{(t)}\sigma_n^2} = \delta^{(t)}\boldsymbol{y} + (1 - \delta^{(t)})\boldsymbol{v}^{(t)}. \quad (6b)$$

The basic idea of model-guided design (MoD) is to unfold conventional model-based iterative algorithms Eq. (6) into the implementation by cascaded DNN. MoD-based IR-CNN [43] and DPDNN [8] have shown promising results on different image restoration tasks such as denoising, deblurring, and super-resolution. Since Eq. (6a) is solved by a manually designed denoising network, there is room for further improvement (e.g., via NAS) in terms of efficiency and better denoising performance.

2.2. Network architecture search (NAS)

Search Strategy. NAS has been proposed to overcome the difficulty of manually designing neural architectures for deep learning and achieved remarkable performance in various high-level tasks. Some early works have adopted reinforcement learning (RL) [48, 49] and evolutionary algorithm (EA) [20, 29] as search strategies. However, both RL-based and EA-based methods require tremendous GPU resources and running time. For example, RL-based NAS-Net [49] and EA-based AmoebaNet [29] take $48k$ and $76k$ GPU hours respectively. Given such prohibitive complexity, a differential search strategy such as DARTS [21] was proposed to relax the discrete search space by a differentiable proxy so NAS can be optimized by gradient descent. Many recent works including ours and [18, 35, 10, 40] have been inspired by this strategy of differential NAS.

Search Space Design. A cell-based search space has been proposed by NASNet [49], where the cell is defined by a directed acyclic graph with several nodes. Those cell-based methods [49, 29, 21] search the operations between nodes (so-called feature maps) in a cell and repeat the cell to gain complete network architecture. However, networks found by cell-based search spaces often suffer from long inferring time. In order to improve on efficiency, a flurry of works such as ProxylessNAS [2], FBNet [35], DenseNAS [10] have proposed a new search space based on MobileNetV2 [30]. By searching for the expansion ratios and kernel sizes of MBConv layers, those NAS methods have often achieved better results than previous methods [48, 49, 29]. These works inspire us to construct a new search space based on another popular architecture (i.e., U-net) and automatically search the layer operations, network width and depth, achieving lightweight and low inferring time simultaneously.
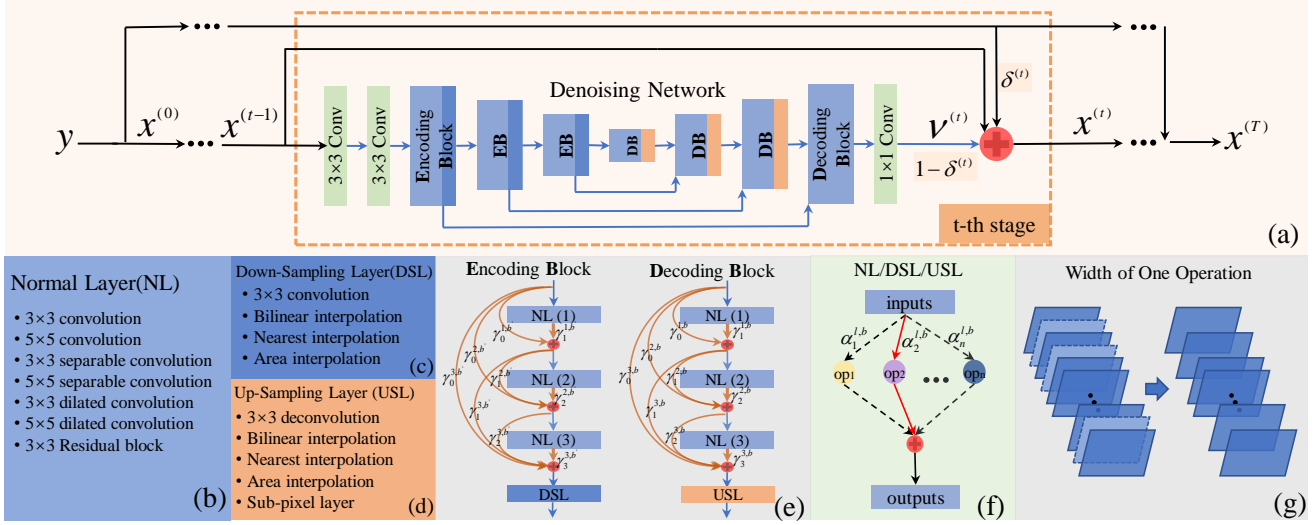
Figure 1. (a) The overall architecture of the proposed network; (b)-(d) a list of candidate operations to be searched for NL, DSL and USL respectively; (e) network depth searching; (f) layer operations searching; (g) network width searching.

NAS for Image Restoration. So far, there have been few works on applying NAS to image restoration. E-CAE [32] exploits for convolutional autoencoders for image inpainting and denoising by employing EA as search strategy, requiring enormous computational resource and costing a lot of time. HiNAS [40] employs gradient-based search strategy on cell-based search space for image denoising and deraining with less search time. Similar to HiNAS, we adopt gradient-based search strategy but design a new search space under model-guided framework for low-level tasks. Besides, our method can automatically search operations of each layer, network width and depth, while HiNAS has only searched the operations of each layer.

3. Model-guided Design with Neural Architecture Search (MoD-NAS)

We first introduce proposed search space constructed under the model-guided framework. Then we present the strategies of searching operations for each layer as well as network width and depth. Finally, we summarize the overall search procedure of MoD-NAS.

3.1. The Search Space: Searchable U-net Under Model-guided Framework

Mathematically, the solution to Eq. (5) can be obtained by iteratively updating $v^{(t)}$ and $x^{(t)}$ following Eq. (6a) and Eq. (6b). By unfolding iterative updating algorithm through a deep network, model-guided methods [43, 8, 26] have achieved excellent performance. However, those networks are still hand-crafted and their optimality remains questionable. Under the model-guided framework, the key is to design a deep network $f_{DN}(x)$. As advocated in [40], NAS serves as an appealing remedy for optimizing neural archi-

tectures. Meanwhile, we focus on searching computationally efficient and lightweight denoising networks with additional domain knowledge of image denoising.

Despite the U-net was proposed for medical image segmentation, the U-net has been demonstrated to have great performance in image restoration domain (e.g., dehazing [7], video deraining [37], video deblurring [31]), showing the strong ability of denoising. Inspired by those success of U-net in image restoration, we propose to construct a new search space under model-guided framework, in which deep denoiser $f_{DN}(x)$ is based on U-net structure. When using U-net as a deep denoising prior, the one-step iteration Eq. (6) can be unfolded into a deep network implementation as shown in Fig. 1(a). The addition operator (red module) faithfully sums up the two terms in Eq. (6b).

Note that the denoising network within the dashed orange box only represents one-step implementation after unfolding; its concatenation into multiple stages unfolds the iterative solution to Eq. (5). Specifically, all layers of both encoding block (EB) and decoding block (DB) are searchable (highlighted by blue and orange modules). Taking DB as an example (refer to the right part of Fig. 1(e)), except the last DB, each block consists of three normal layers and one upsampling layer. The last DB consists of three normal layers only. Similarly, each EB consists of three normal layers and one down-sampling layer. During the search, there are *seven*, *four* and *five* candidate operations to be searched in normal layer, down-sampling layer and up-sampling layer respectively. The detailed candidate operations for each layer are listed in Fig. 1(b-d). The network width and depth can also be automatically selected via proposed searching strategies. By relaxing all discrete architectures of network space into differential formulations, we can search the architectures with gradient descent algo-
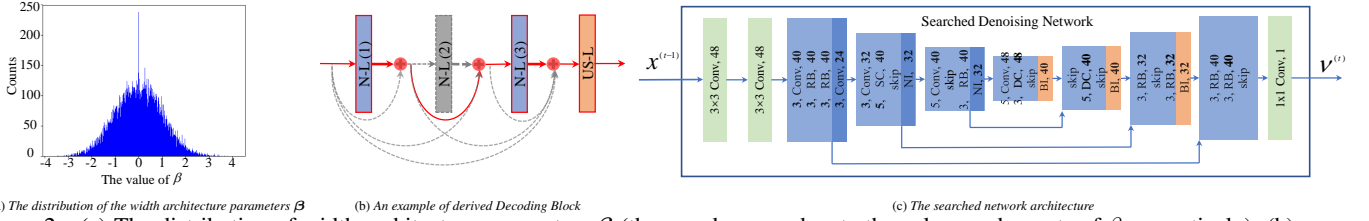
(a) *The distribution of the width architecture parameters $\boldsymbol{\beta}$* (b) *An example of derived Decoding Block* (c) *The searched network architecture*

Figure 2. (a) The distribution of width architecture parameters $\boldsymbol{\beta}$ (the x and y axes denote the values and counts of β respectively); (b) an example of derived Decoding Block (the selected layer is highlighted by solid red bounding box and the discarded layer is shown by dashed gray bounding box); (c) the searched network architecture where RB denotes Residual Block, SC denotes Separable Convolution, DC denotes Dilated Convolution, NI denotes Nearest Interpolation, BI denotes Bilinear Interpolation. In each layer, the first number denotes the kernel size and the last number denotes the number of output channels (zoom in for better view). More details of searched networks can be found in appendix.

rithms such as ADAM [15].

3.2. The Search Strategies

Layer Operations. As shown in Fig. 1(b-d), there are $seven$, $four$ and $five$ candidate operations to be searched in normal layer (NL), down-sampling layer (DSL) and up-sampling layer (USL) respectively. It is worth mentioning that each convolution operation starts with a ReLU activation function but isn't followed by a batch normalization layer since it requires more GPU memory [17]. And each interpolation operation is followed by a 1×1 convolution for channel conversion. Taking the normal layer as an example, let $\mathbb{O}$ denotes the set of candidate operations listed in Fig. 1(b). Every candidate operation $o \in \mathbb{O}$ of each layer ℓ in block b has been allocated an architecture parameter $\alpha_o^{\ell,b}$. We have adopted the $softmax$ function to compute the architecture weight for every operation of layer ℓ in block b:

$$\omega_o^{\ell,b} = \frac{exp(\alpha_o^{\ell,b})}{\sum_{o' \in \mathbb{O}} exp(\alpha_{o'}^{\ell,b})}. \tag{7}$$

Finally, the output of operations in layer ℓ in block b (refer to Fig. 1(f)) can be expressed by

$$\boldsymbol{z}^{\ell,b} = \sum_{o \in \mathbb{O}} \omega_o^{\ell,b} \cdot o(L^{\ell-1,b}), \tag{8}$$

where $L^{\ell-1,b}$ denotes the inputs of layer ℓ in block b. In a similar manner, all other types of layers can also be relaxed.

In summary, the task of choosing the best operation for layer ℓ in block b has been translated to the problem of optimizing architecture parameters $\alpha^{\ell,b}$ which can be solved through gradient descent algorithms [21]. After the supernet is trained, we only need to choose the operation $\hat{o}^{\ell,b}$ with the largest architecture weight $\omega_{\hat{o}}^{\ell,b}$ computed by $\alpha^{\ell,b}$ with Eq. (7) and discard the others. In other words, the selection of layer operations can be formulated into the following:

$$\hat{o}^{\ell,b} = \underset{o \in \mathbb{O}}{\mathrm{argmax}}\, \omega_o^{\ell,b}. \tag{9}$$

As shown in Fig. 1(f), only one operational pathway highlighted by solid red arrows has been selected.

Network Width. HiNAS [40] and Autolab [18] search the width of a cell by stacking cells with different widths side-by-side (i.e., $1W$, $2W$, and $4W$, where W is the basic width which has to be set manually before the search). Such strategy suffers from prohibitive cost of computation, GPU memory and searching time. Besides, this search strategy can only search a finite number of width parameters (as determined by W) and all layers in one cell have to share the same width. It follows that many potential architectures (e.g., with varying layer width in one cell) are excluded from the search, implying the lack of flexibility.

Inspired by the rapid advances in network pruning [24], we propose a new *highly reusable width-search* method for searching the width of every layer. As shown in the left part of Fig. 1(g), every channel of operation o of layer ℓ in block b has been assigned an architecture parameter $\beta_c^{o,\ell,b}$, where c denotes the *c-th* channel of totally C channels. During the search process, the width and other architecture parameters will be optimized together by the gradient descent algorithm. Once the supernet has been trained, the probability distribution of the width architecture parameters $\boldsymbol{\beta}$ typically observes a heavy-tailed distribution with a single peak at the origin (as shown in Fig. 2(a)). Such observation implies that we can discard the channel with small architecture parameters $\beta_c^{o,\ell,b}$ around zero. Specifically, we have chosen the top-$\hat{M}$ channels with the largest architecture parameters $\beta_c^{o,\ell,b}$. The criterion for our selection can be written as:

$$\sum_{i=1}^{\hat{M}} |\beta_i^{o,\ell,b}| \geq 90\% \cdot \sum_{j=1}^{C} |\beta_j^{o,\ell,b}| \tag{10a}$$

$$\hat{M} \mod 2^n = 0, \tag{10b}$$

where Eq. (10a) reflects the idea of preserving large parameters only and Eq. (10b) is used for GPU acceleration (we have set $n = 3$ in our experiment) [10]. After the pruning, preserved channels are shown in the right part of Fig. 1(g).

Note that the actual number of preserved channels is a variable, which makes our architecture more flexible than fixed-width searching such as HiNAS [40] and E-CAE [32].

Network Depth. Searching the suitable depth of a network is crucial to the success of NAS-related applications. Previous works [36, 35, 2] usually search the depth of network by adding a skip operation in the candidates. Once a skip operation has been selected, it is equivalent to the reduction of network depth by one layer. An undesirable consequence of such search strategy is that the produced network might be too shallow when many skip operations have been selected. One possible explanation of this phenomenon is that the skip has the same probability as the other operations and skip is often more frequently selected during search because it does not have any parameter. In this paper, we propose a new *densely connected block* to address this issue by utilizing dense connections [45] to implement the strategy of searching the suitable depth of the network.

As shown in Fig. 1(e), every candidate connection highlighted by orange color have been assigned an architecture parameter. Namely the path from layer i to layer ℓ in block b has a parameter $\gamma_i^{\ell,b}$. Similar to the relaxation of layer operations, we adopt $softmax$ function to compute the probability $p_i^{\ell,b}$ of each path. Let $L^{\ell,b}$ denotes the outputs of layer ℓ of block b, then

$$
\begin{aligned}
L^{\ell,b} &= \sum_{i=0}^{\ell-1} p_i^{\ell,b} \cdot L^{i,b} + p_\ell^{\ell,b} \cdot z^{\ell,b} \\
p_i^{\ell,b} &= \frac{exp(\gamma_i^{\ell,b})}{\sum_{j=0}^{\ell} exp(\gamma_j^{\ell,b})},
\end{aligned} \tag{11}
$$

where $z^{\ell,b}$ denotes the output of operations in layer ℓ in block b. After the supernet is trained, we only choose the paths with the largest probability $p_i^{\ell,b}$ in each layer and discard the others as shown in Fig. 2(b).

3.3. Overall Search Procedure

Putting things together, we can see how the search space constructed by MoD can be seamlessly integrated with the strategy of NAS as follows (refer to Algorithm 1). The unfolding process can be summarized into the procedure of **Forward inferring** of Algorithm 1. During this procedure, the mapping $f_{DN}^{(t)}$ corresponds to the denoising network in Fig. 1(a) at *t-th* stage. Our MoD-NAS adopts a differentiable search strategy, where the search process can be optimized by gradient descent algorithms such as ADAM [15]. We train the supernet with the following MSE loss:

$$
(\mathbb{W}, \mathbb{A}) = \underset{\mathbb{W},\mathbb{A}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathcal{F}(\boldsymbol{y}_i; \mathbb{W}, \mathbb{A}) - \boldsymbol{x}_i\|_2^2, \tag{12}
$$

where $\boldsymbol{y}_i$ and $\boldsymbol{x}_i$ denote the i-th pair of degraded and original image patches respectively and $\mathcal{F}(\boldsymbol{y}_i; \mathbb{W}, \mathbb{A})$ denotes

---

Algorithm 1 Proposed MoD-NAS Algorithm.

- **Initialization**:
    (1) Initialize x as $\boldsymbol{x}^{(0)} = \boldsymbol{y}$.
- **While** the search process not converge **do**
    (1) **Forward inferring**: for $t = 1, 2, ..., T$, **do**
        a) Compute $\boldsymbol{v}^{(t)} = f_{DN}^{(t)}(x^{(t-1)})$;
        b) Compute $\boldsymbol{x}^{(t)}$ via Eq. (6b);
        c) $t = t + 1$.
        End for
    (2) **Backpropagation**:
        a) Update weights by descending $\nabla\mathbb{W}\mathcal{L}_{train}(\mathbb{W}, \mathbb{A})$;
        b) Update architecture parameters by descending $\nabla\mathbb{A}\mathcal{L}_{val}(\mathbb{W}, \mathbb{A})$.
- **Derive the final architecture**:
    (1) derive the final architecture based on the depth parameter $\boldsymbol{\gamma}$, operations parameters $\boldsymbol{\alpha}$ via Eq. (9) and width parameters $\boldsymbol{\beta}$ via Eq. (10) in sequence.

---

the reconstructed image patch by the supernet with the parameters set of operation weights $\mathbb{W}$ and the parameters set of architecture $\mathbb{A}$. We alternatively optimize the operation weights by descending $\nabla\mathbb{W}\mathcal{L}_{train}(\mathbb{W}, \mathbb{A})$ on the training set , and optimize the architecture parameters by descending $\nabla\mathbb{A}\mathcal{L}_{val}(\mathbb{W}, \mathbb{A})$ on the validation set as shown in procedure **Backpropagation** of Algorithm 1. When the supernet has been trained, we derive the final architecture based on the parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ as shown in Algorithm 1. An example of searching for the final denoising network architecture is shown in Fig. 2(c).

4. Experimental Results

4.1. Experimental Settings

Benchmark Datasets. We have randomly selected 4000 images from the Waterloo dataset for training. Following [42, 28, 8, 13], three standard benchmark datasets (Set12, BSD68, Urban100) are used for testing. The noisy images are generated by adding white Gaussian noise to the corresponding clean images with $\sigma = 15, 25, 50$ following [42, 8, 13]. It is worth mentioning that the search experiments are conducted with $\sigma = 25$ and the training experiments are conducted with $\sigma = 15, 25, 50$.

Search Settings. The training dataset [25] has been equally divided into nonoverlapping two parts: one for updating the weights of network operations (Training $\mathbb{W}$) and the other for updating the architecture parameters (Training $\mathbb{A}$). We randomly select 12 noisy patches sized by 64×64 as the inputs. Two ADAM optimizers [15] with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ are adopted to optimize parameters sets $\mathbb{W}$ and $\mathbb{A}$ respectively. The learning rate of two optimizers decays from 10^{-3} to 10^{-5} with the cosine annealing schedule [23] within 140 epochs. The stages of supernet is set to $T = 2$ and the initial number of channels is set to $C = 48$.
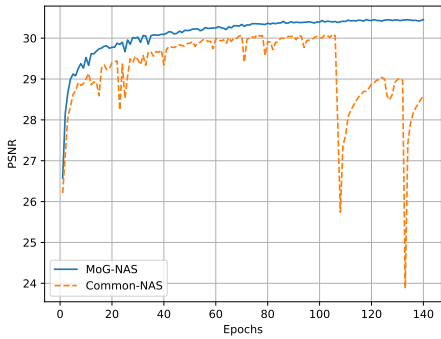
Figure 3. The performance of supernet evaluated on Set12 dataset during search process with $\sigma = 25$. The blue line refers to the search process with MoD-NAS and orange line refers to the search process with differentiable NAS (without MoD-NAS).

In the first 40 epochs, we only update the parameters set $\mathbb{W}$ of operations and the parameters sets $(\mathbb{W}, \mathbb{A})$ are optimized alternately in the remaining epochs. Our network is implemented by the Pytorch framework and the total searching time takes about **7** hours using one NVIDIA 2080Ti GPU.
Training Settings. We train the network that searched by our method with MSE loss. We randomly select 32 noisy patches sized by 128×128 as the inputs. The ADAM algorithm [15] with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ is adopted to optimize the network. The learning rate decays from 10^{-3} to 10^{-5} with the cosine annealing schedule [23] within 600 epochs. The searched denoising network is shown in Fig. 2(c). Our network is implemented by Pytorch and the total training takes about **10** hours when $T = 3$ and using one NVIDIA RTX 2080Ti GPU.
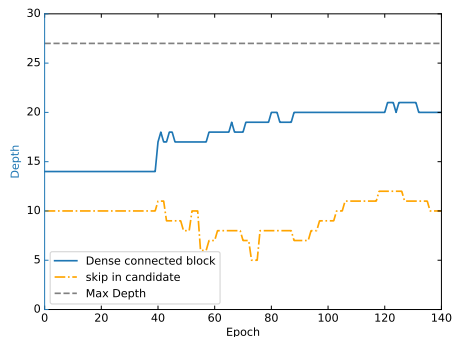

Figure 4. The benefit of searching for network depth. The blue line refers to the search process with dense connected block and orange line refers to the search process adding skip in candidates and gray line refers to the max depth of one U-net.

4.2. Ablation Study

Benifits of MoD-NAS. During the searching process of differentiable NAS, there is a phenomenon that the performance of supernet drops a lot when the number of search epochs becomes large. This phenomenon known as *mode collapse* has been observed in both high-level tasks[16] and

low-level tasks [40]. DATRS+ [16] for image classification and HiNAS [40] for image denoising have employed a similar way called *early stopping*. However, this heuristic strategy does not solve the problem of mode collapse but gets around it.

In this paper, by incorporating model-guided design with differentiable NAS (MoD-NAS), the performance of supernet remains stable and even increases slightly when the number of search epochs becomes large. To demonstrate this advantage, we have compared the performance of our MoD-NAS against common differentiable NAS [21] during the search process evaluated on the Set12 dataset as shown in Fig. 3. Note that we have removed the model-guided design part by only searching one denoising network as the baseline control; while all other settings are kept the same as MoD-NAS. From Fig. 3, we can see that the performance of proposed MoD-NAS stays stable during the whole search process, while the baseline method suffers from apparent collapses, especially when $Epochs > 100$. Besides, during the search process, our proposed MoD-NAS demonstrates improved stability, showing a much smoother PSNR curve than the baseline. We argue that such benefit of the proposed MoD-NAS method can be explained away by MoD-NAS providing a smoother search space than differentiable NAS [21]. Therefore, the proposed MoD-NAS enjoys excellent convergence property thanks to its search space is built under mode-guided framework with domain knowledge.

Benefits of Searching for Network Width. To verify the effectiveness of searching network width, we have conducted the experiment MoD-NAS(T=3) ($C = 64$), which changes all channels of MoD-NAS(T=3) to 64. The experiment results of different widths of MoD-NAS(T=3) have been shown in Tab. 1, from which we can see that MoD-NAS(T=3) achieved almost the same result on Set12 dataset. Therefore, MoD-NAS(T=3) achieves a better trade-off between the number of parameters or flops and accuracy in searching for network width, reaching the goal of efficiency.

Table 1. Comparisons of different width of MoD-NAS(T=3) on Set12 dataset with $\sigma = 25$.

Models	parameters	flops	PSNR
MoD-NAS(T=3)	1256k	1.45G	30.88
MoD-NAS(T=3) ($C = 64$)	3245k	3.46G	30.89

Benefits of Searching for Network Depth. We also have conducted experiments to verify the validity of proposed densely connected block for searching the depth of network. As shown in Fig. 4, the blue line shows the search with dense connected search strategies. It can be seen that the search procedure is more stable than adding skips (orange line) and the derived network converges rapidly (while adding skips in candidates converges to a shallow network).

Table 2. Average PSNR results for **Gaussian** image denoising on three benchmark datasets. The best performance is shown by bold and the second best performance is shown by underline. The testing time is the total time of evaluating the whole Urban100 dataset.

Methods	Set12			BSD68			Urban100			params	flops	testing time
	15	25	50	15	25	50	15	25	50			
BM3D [6]	32.37	29.97	26.72	31.07	28.57	25.62	32.35	29.70	25.95	-	-	-
TNRD [3]	32.50	30.06	26.81	31.42	28.92	25.97	31.86	29.25	25.88		-	-
DnCNN [42]	32.86	30.44	27.18	31.73	29.23	26.23	32.68	29.97	26.28	556k	1.28G	21.45s
N3Net [28]	-	30.55	27.43	-	29.30	26.39	-	30.19	26.82	706k	1.62G	594.38s
MemNet [33]	32.96	30.60	27.03	30.76	29.19	25.25	32.33	30.68	25.56	2905k	6.69G	93.83s
DPDNN [8] [a]	32.91	30.54	27.50	31.83	29.27	26.40	32.98	30.30	26.85	1363k	5.25G	68.62s
FOCNet [13]	33.07	30.73	27.68	31.83	29.38	26.50	33.15	30.64	27.40	-	-	125.13s
RDN [47]	32.95	30.66	27.60	31.74	29.29	26.41	33.04	30.50	27.40	21970k	50.6G	386.95s
E-CAE [32]	-	-	26.53	-	-	25.86	-	-	24.51	1062k	2.45G	40.04s
HiNAS [40] [b]	32.50	30.35	27.25	31.16	28.92	26.04	31.92	29.52	26.01	630k	-	145.89s
MoD-NAS(T=1)	33.09	30.73	27.62	31.86	29.40	26.50	32.93	30.38	26.88	**418k**	**0.48G**	**11.45s**
MoD-NAS(T=3)	33.21	30.88	27.83	31.91	29.47	26.59	33.19	30.73	27.37	1253k	1.45G	18.12s
MoD-NAS(T=6)	**33.28**	**30.95**	**27.87**	**31.94**	**29.50**	**26.61**	**33.34**	**30.88**	**27.54**	2506k	2.91G	33.39s

[a] DPDNN can be viewed as a baseline method since it is a model-guided method and the denoising network is manually designed based on U-net.

[b] Since the code of HiNAS is not publicly available, we have tried our best to reproduce HiNAS based on the technical details of the original paper.

4.3. Comparisons with State-of-the-art Methods

For image denoising, we have compared MoD-NAS with several current state-of-the-art methods on three commonly used datasets. The average PSNR results of the benchmark methods in Tab. 2 are either directly cited from the original papers or reproduced by running the officially released source codes. The testing time is shown in Tab. 2 is the total time of evaluating the whole Urban100 dataset when $\sigma = 25$. To gain deeper insight into the deep unfolding framework, we have considered three different T values and compared their PSNR performance. We have shown the experimental results of $T = 1$, $T = 3$ and $T = 6$ in Tab. 2.

It can be observed that our MoD-NAS is consistently superior to other competing methods in terms of PSNR performance for all datasets. More importantly, our single-stage model (MoD-NAS(T=1)) has achieved comparable and even better performance than most benchmark methods with the fewest parameters, the lowest number of flops, and the least amount of testing time. As for bigger models, MoD-NAS(T=3) and MoD-NAS(T=6) have achieved better results than other benchmark methods with a larger margin as shown in Tab. 2. Although the number of parameters in MoD-NAS(T=3) and MoD-NAS(T=6) is comparable to those in previous methods such as N3Net [28], DPDNN[8], MoD-NAS(T=3) and MoD-NAS(T=6) have a great advantage over others in terms of flops and testing time. For instance, our finalized MoD-NAS(T=3) has $1253k$ parameters, which is only **5.7%** that of RDN [47] and **43%** that of MemNet[33]. When compared with RDN [47], our MoD-NAS(T=3) reduces the testing time on Urban100 dataset by as much as **95.3%** with even better PSNR results. When compared with model-guided method DPDNN [8] that the denoising network was manually designed based on U-net, our single-stage network MoD-NAS(T=1) has gained

$0.09dB$ over DPDNN [8] on the average with only **30.7%** parameters, **9.1%** flops and **16.7%** testing time. We have also shown the visual comparison of denoised images with competing methods in Fig. 5, from which one can clearly observe the superiority of our method in terms of more fine-detailed texture patterns. The average SSIM and more visual results can be found in appendix.
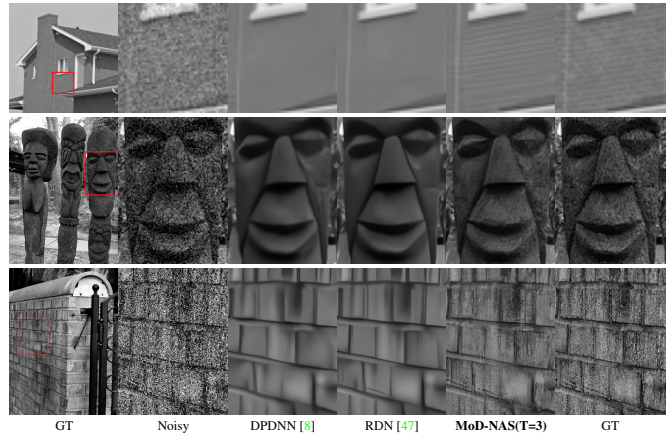

Figure 5. Denoising visual quality comparison. The first row shows the comparison for an image from Set12 with $\sigma = 15$; the second row shows the comparison for an image from BSD68 with $\sigma = 25$; the third row shows the comparison Urban100 for an image from with $\sigma = 50$ (zoom in for better view).

4.4. Comparisons with Other NAS Methods

Table 3. Comparison of other NAS methods on BSD200 dataset with $\sigma = 50$.

Models	PSNR	GPU (Memory)	search+training time	search strategy
E-CAE	26.17	4 Tesla P100 (16GB)	96 hours	EA
HiNAS	26.77	1 Tesla V100 (32GB)	16.5 hours	gradient
MoD-NAS(T=3)	27.26	1 RTX 2080Ti (11GB)	17 hours	gradient

Only a small number of NAS methods (e.g., image

super-resolution [12, 5] and denoising [32, 40]) have been proposed for low-level vision tasks. Here we compare our proposed MoD-NAS with E-NAS [32] and HiNAS [40] in Tab. 2 and Tab. 3. Apparently, in Tab. 3, methods with search strategy based on gradient descent have advantages on the cost of GPU memory and searching/training time. When compared with HiNAS [40], our proposed method has the following main advantages.

- We have proposed a new efficient search space under model-guided framework, which deep denoiser is based on U-net to address the problem that networks found by cell-based search spaces which Hi-NAS adopted often suffer from long testing time. As shown in Tab. 2, MoD-NAS(T=1) takes **7.8%** testing time of HiNAS and haves less parameters than HiNAS, which demonstrates MoD-NAS achieves lightweight and low inferring time simultaneously with more competent performance than HiNAS.

- Compared with HiNAS and E-CAE in Tab. 2 and Tab. 3, our searched networks MoD-NAS(T=1,3,6) achieve much better performance in terms of PSNR, which indicates the superiority of MoD-NAS

- By employing a new highly reusable width search strategy for searching the network width, our supernet can be searched by using one $11GB$ 2080Ti GPU while HiNAS [40] is trained by using one $32GB$ V100 GPU.

4.5. Real image denoising on SIDD Dataset.

To demonstrate the generalization ability of our searched network, we evaluate the performance of MoD-NAS(T=3) on a real blind denoising task with SIDD [1] benchmark. SIDD dataset has provided one medium training set (320 image pairs) and a validation set (40 image pairs) for fast training and evaluation, but the testing results can only be obtained by online submission. We have trained MoD-NAS(T=3) on the medium training set and obtained results of testing sets by online submission. Tab. 4 shows the PSNR and SSIM results of different methods on SIDD testing set. Note that the results on testing sets are cited from the official website[1]. From Tab. 4, we can see that MoD-NAS(T=3) can achieve promising results in terms of PSNR and SSIM comparing with other methods. We have shown the real image denoising visual comparison on SIDD dataset in Fig. 6, from which we can see that our method has achieved a better result than other methods (e.g., more effective noise suppression).

4.6. Image Compression Artifact Reduction.

We apply the proposed MoD-NAS search method to image compression artifact reduction for further evaluating the generalization capability of our method. We employ the

---

Table 4. Comparison of different methods on SIDD testing set.

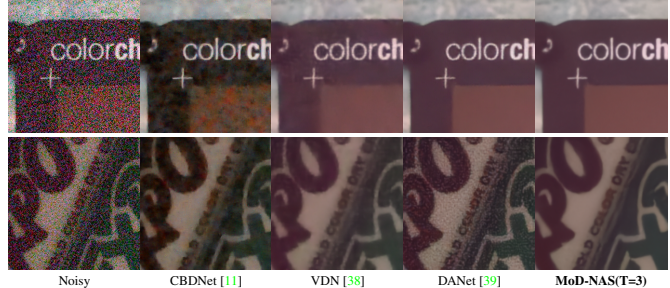| Methods	CBDNet [11]	VDN [38]	DANet [39]	**MoD-NAS(T=3)**
PSNR	33.28	39.26	39.25	39.29
SSIM	0.868	0.955	0.955	0.955
params	4346k	2325k	9154k	1253k

Figure 6. Real image denoising visual quality comparison on SIDD testing set(zoom in for better view).

Table 5. Comparisons of different methods on image compression artifact reduction with $q = 20$ on LIVE1 dataset.

Methods	TNRD [3]	DnCNN [42]	MemNet [33]	RDN [47]	**MoD-NAS-AR**
PSNR	31.46	31.59	31.83	32.07	32.30
SSIM	0.8769	0.8802	0.8846	0.8882	0.8945
params	-	668k	2905k	21970k	1670K

same searching and training setting as the denoising experiments. The results of JPEG quality $q = 20$ are listed in Tab. 5 and shown in Fig. 10. As shown in Tab. 5, the MoD-NAS-AR searched by MoD-NAS achieves much better performance than other methods. When compared with RDN, MoD-NAS-AR gains **0.23**dB improvement in terms of PSNR with only **7.6%** parameters of RDN. More visual comparisons can be found in appendix.
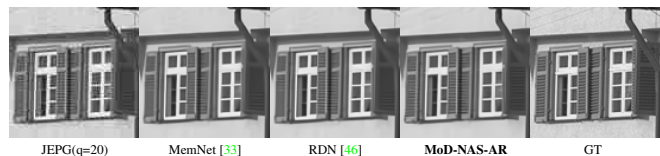

Figure 7. Image compression Artifact reduction visual quality comparison on LIVE1 dataset (zoom in for better view).

5. Conclusion

In this paper, we have presented a novel MoD-NAS based approach to image denoising. By incorporating the strengths of model-guided design and NAS, we have constructed a new search space and designed flexible search strategies specially tailored for the task of image denoising. Through searching in the space of concatenated U-net, we demonstrate how the joint consideration of layer operation, network width, and network depth can lead to a network solution with excellent performance including visual quality and convergence property. Our proposed network could achieve at least comparable and often even better PSNR results than current leading methods with less number of parameters and flops as well as less amount of testing time.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1692–1700, 2018. 8

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2018. 2, 5

[3] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 39(6):1256–1272, 2017. 1, 7, 8

[4] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. Detnas: Neural architecture search on object detection. 2019. 1

[5] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, Jixiang Li, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. *arXiv preprint arXiv:1901.07261*, 2019. 1, 8

[6] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. 2, 7

[7] Hang Dong, Jinshan Pan, Lei Xiang, Zhe Hu, Xinyi Zhang, Fei Wang, and Ming-Hsuan Yang. Multi-scale boosted dehazing network with dense feature fusion. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2157–2167, 2020. 3

[8] Weisheng Dong, Peiyao Wang, Wotao Yin, Guangming Shi, Fangfang Wu, and Xiaotong Lu. Denoising prior driven deep neural network for image restoration. *IEEE transactions on pattern analysis and machine intelligence*, 41(10):2305–2318, 2018. 1, 2, 3, 5, 7, 12

[9] Weisheng Dong, Lei Zhang, Rastislav Lukac, and Guangming Shi. Sparse representation based image interpolation with nonlocal autoregressive modeling. *IEEE Transactions on Image Processing*, 22(4):1382–1394, 2013. 2

[10] Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10628–10637, 2020. 1, 2, 4

[11] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1712–1722, 2019. 8

[12] Yong Guo, Yongsheng Luo, Zhenhao He, Jin Huang, and Jian Chen. Hierarchical neural architecture search for single image super-resolution. *IEEE Signal Processing Letters*, 27:1255–1259, 2020. 1, 8

[13] Xixi Jia, Sanyang Liu, Xiangchu Feng, and Lei Zhang. Focnet: A fractional optimal control network for image denoising. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6054–6063, 2019. 5, 7

[14] Kwang In Kim and Younghee Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1127–1133, 2010. 2

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4, 5, 6

[16] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping, 2020. 6

[17] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1132–1140, 2017. 4

[18] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 82–92, 2019. 1, 2, 4

[19] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. Non-local recurrent network for image restoration. In *Advances in Neural Information Processing Systems*, pages 1673–1682, 2018. 1

[20] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR 2018 : International Conference on Learning Representations 2018*, 2018. 2

[21] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR 2019 : 7th International Conference on Learning Representations*, 2019. 1, 2, 4, 6

[22] Xing Liu, Masanori Suganuma, Zhun Sun, and Takayuki Okatani. Dual residual networks leveraging the potential of paired operations for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7007–7016, 2019. 1

[23] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR (Poster)*, 2016. 5, 6

[24] Xiaotong Lu, Han Huang, Weisheng Dong, Xin Li, and Guangming Shi. Beyond network pruning: a joint search-and-training approach. In *Proc. of IJCAI*. 1, 4

[25] Kede Ma, Zhengfang Duanmu, Qingbo Wu, Zhou Wang, Hongwei Yong, Hongliang Li, and Lei Zhang. Waterloo Exploration Database: New challenges for image quality assessment models. *IEEE Transactions on Image Processing*, 26(2):1004–1016, Feb. 2017. 5

[26] Q. Ning, W. Dong, G. Shi, L. Li, and X. Li. Accurate and lightweight image super-resolution with model-guided deep unfolding network. *IEEE Journal of Selected Topics in Signal Processing*, pages 1–1, 2020. 3

[27] Stanley J. Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling and Simulation*, 4(2):460–489, 2005. 2

[28] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In *Advances in Neural Information Processing Systems*, pages 1087–1098, 2018. 1, 5, 7, 12

[29] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):4780–4789, 2019. 2

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 2

[31] Hyeonjun Sim and Munchurl Kim. A deep motion deblurring network based on per-pixel adaptive kernels with residual down-up and up-down modules. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2140–2149, 2019. 3

[32] Masanori Suganuma, Mete Ozay, and Takayuki Okatani. Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search. In *ICML 2018: Thirty-fifth International Conference on Machine Learning*, pages 4771–4780, 2018. 1, 3, 5, 7, 8

[33] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4549–4557, 2017. 1, 7, 8, 12, 13

[34] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2820–2828, 2019. 1

[35] Bichen Wu, Kurt Keutzer, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, and Yangqing Jia. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10734–10742, 2019. 2, 5

[36] Lingxi Xie, Xin Chen, Kaifeng Bi, Longhui Wei, Yuhui Xu, Zhengsu Chen, Lanfei Wang, An Xiao, Jianlong Chang, Xiaopeng Zhang, et al. Weight-sharing neural architecture search: A battle to shrink the optimization gap. *arXiv preprint arXiv:2008.01475*, 2020. 1, 5

[37] Wenhan Yang, Jiaying Liu, and Jiashi Feng. Frame-consistent recurrent video deraining with dual-level flow. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1661–1670, 2019. 3

[38] Zongsheng Yue, Hongwei Yong, Qian Zhao, Lei Zhang, and Deyu Meng. Variational denoising network: Toward blind noise modeling and removal. In *Advances in Neural Information Processing Systems*, pages 1690–1701, 2019. 8

[39] Zongsheng Yue, Qian Zhao, Lei Zhang, and Deyu Meng. Dual adversarial network: Toward real-world noise removal and noise generation. *arXiv: Computer Vision and Pattern Recognition*, 2020. 8

[40] Haokui Zhang, Ying Li, Hao Chen, and Chunhua Shen. Memory-efficient hierarchical neural architecture search for image denoising. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3657–3666, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[41] Kai Zhang, Luc Van Gool, and Radu Timofte. Deep unfolding network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3217–3226, 2020. 1

[42] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. 1, 5, 7, 8

[43] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2808–2817, 2017. 2, 3

[44] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Deep plug-and-play super-resolution for arbitrary blur kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1671–1681, 2019. 2

[45] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018. 5

[46] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2472–2481, 2018. 8, 12, 13

[47] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. 7, 8

[48] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *ICLR 2017 : International Conference on Learning Representations 2017*, 2017. 1, 2

[49] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018. 2

A. More Comparisons of Gaussian image denoising with State-of-the-art Methods

Here, we have reported SSIM results and more visual comparisons of Gaussian image denoising results in Tab. 6 and Fig. 9 respectively. As shown in Tab. 6, it is easy to see that MoD-NAS is superior to all competing methods in terms of SSIM values. Besides, the visual comparison results with different σ are shown in Fig. 9 where our searched network performs better than other competing methods in terms of more recovered image texture details. For example, the first row of Fig. 9 shows the result of 'Img_027' from BSD68 with $\sigma = 25$, where MoD-NAS-B has recovered with fewer visible artifacts and sharper edge of bird hair than other competing methods; the second row of Fig. 9 shows the result of 'Img_013' from Urban with $\sigma = 50$, where our searched network has recovered the stripe shape of wooden window better than others.

the average PSNR results of different stages T from two to seven with $\sigma = 25$. It can be seen that the PSNR increases as the number of stages increases which indicates that we can choose the suitable number of stages T based on the real application to balance the trade-off between performance and cost.

D. Details of Searched Architectures

Architecture details of our searched MoD-NAS-B for image denoising and MoD-NAS-AR for image compression artifact reduction are described in Tab. 7 and Tab. 8 respectively.

Figure 8. The average PSNR performance as a function of parameter T (the total number of Unet stages) of proposed MoG-NAS with $\sigma = 25$ on Set12.

B. More Visual Comparisons of Image Compress Artifact Reduction with State-of-the-art Methods

In this section, we have shown more visual comparisons of image compression artifact reduction with $q = 20$ in Fig. 10. Taking the second row of Fig. 10 as an example, our searched MoD-NAS-AR has recovered the right letters ITH on the helmet with fewer artifacts than other competing methods.

C. Ablation study of impact of stage number T

To explore the impact of the number of unfolded stages on the denoising performance, we have conducted another experiment with varying the parameter T. Fig. 8 shows

'Img_027' from BSD68   Noisy with $\sigma = 25$ N3Net [28]   MemNet [33]

DPDNN [8]   RDN [46]   **MoD-NAS-B**   GT

'Img_013' from Urban100   Noisy with $\sigma = 50$ N3Net [28]   MemNet [33]

DPDNN [8]   RDN [46]   **MoD-NAS-B**   GT

'Img_017' from Urban100   Noisy with $\sigma = 50$ N3Net [28]   MemNet [33]

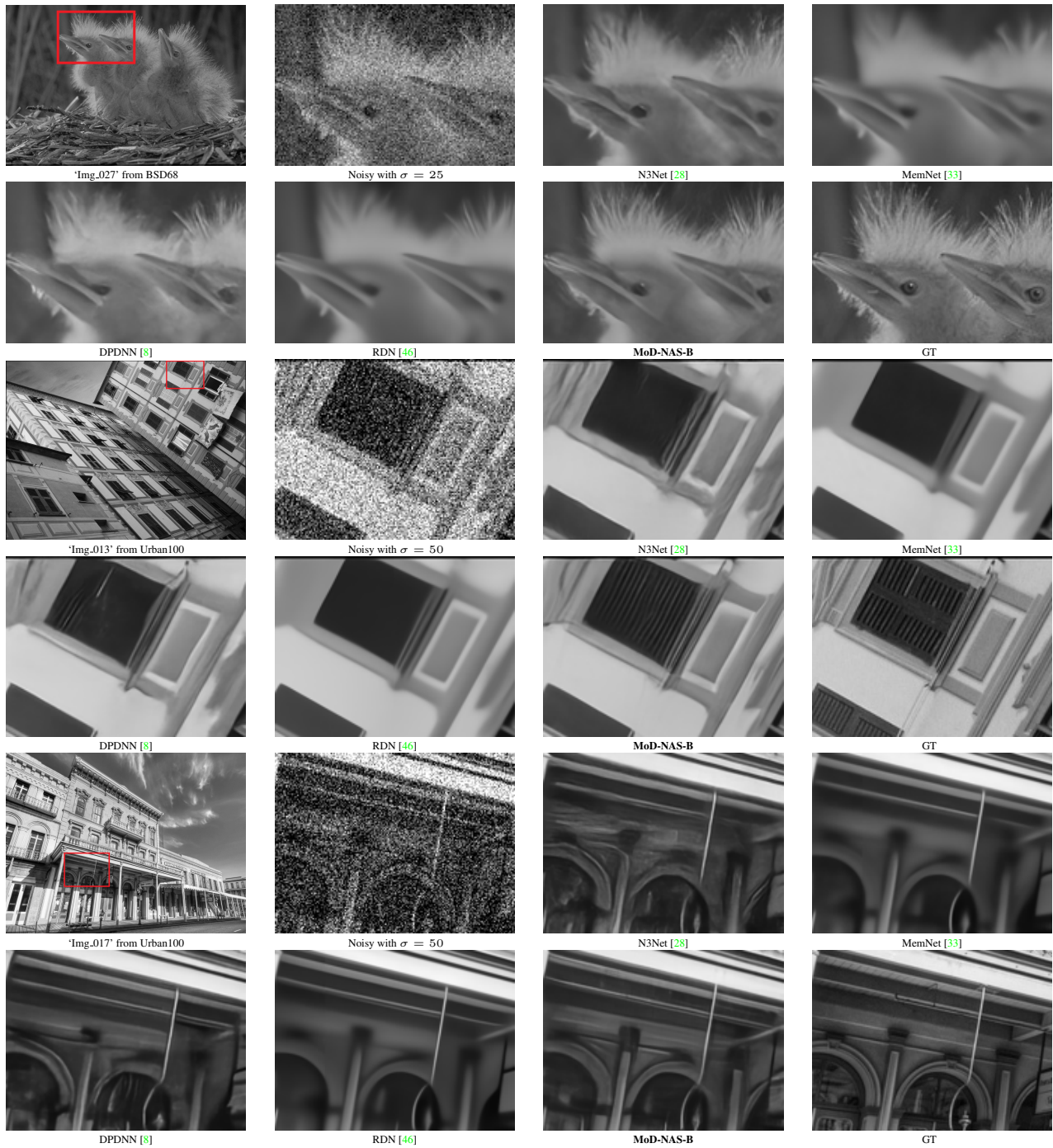DPDNN [8]   RDN [46]   **MoD-NAS-B**   GT

Figure 9. Image denoising visual quality comparison (zoom in for better view).

Table 6. Average SSIM results for **Gaussian** image denoising on three benchmark datasets. The best performance is shown by bold. The testing time is the total time of evaluating the whole Urban100 dataset.

Methods	Set12			BSD68			Urban100			params	flops	testing time
	15	25	50	15	25	50	15	25	50			
BM3D	0.8952	0.8504	0.7676	0.8717	0.8013	0.6864	0.9220	0.8777	0.7791	-	-	-
TNRD	0.8958	0.8512	0.7680	0.8769	0.8093	0.6994	0.9031	0.8473	0.7563	-	-	-
DnCNN	0.9031	0.8622	0.7829	0.8907	0.8278	0.8278	0.9255	0.8797	0.7874	556k	1.28G	21.45s
N3Net	-	0.8644	0.7939	-	0.7957	0.6455	-	0.8917	0.8148	706k	1.62G	594.38s
MemNet	0.9001	0.8652	0.7563	0.8848	0.7966	0.6466	0.9264	0.8793	0.7554	2905k	6.69G	93.83s
DPDNN	0.8970	0.8594	0.7907	0.8738	0.8123	0.7095	0.9322	0.8937	0.8166	1363k	5.25G	68.62s
RDN	0.9030	0.8680	0.7504	0.8884	0.7942	0.6431	0.9291	0.8804	0.7657	21970k	50.6G	386.95s
MoG-NAS(T=1)	0.9070	0.8689	0.7999	0.894	0.8350	0.7325	0.9319	0.8953	0.8163	**418k**	**0.48G**	**11.45s**
MoG-NAS(T=3)	0.9087	0.8720	0.8065	0.8950	0.8365	0.7371	0.9347	0.9011	0.8317	1253k	1.45G	18.12s
MoG-NAS(T=6)	**0.9101**	**0.8729**	**0.8070**	**0.8956**	**0.8371**	**0.7376**	**0.9364**	**0.9039**	**0.8358**	2506k	2.91G	33.39s

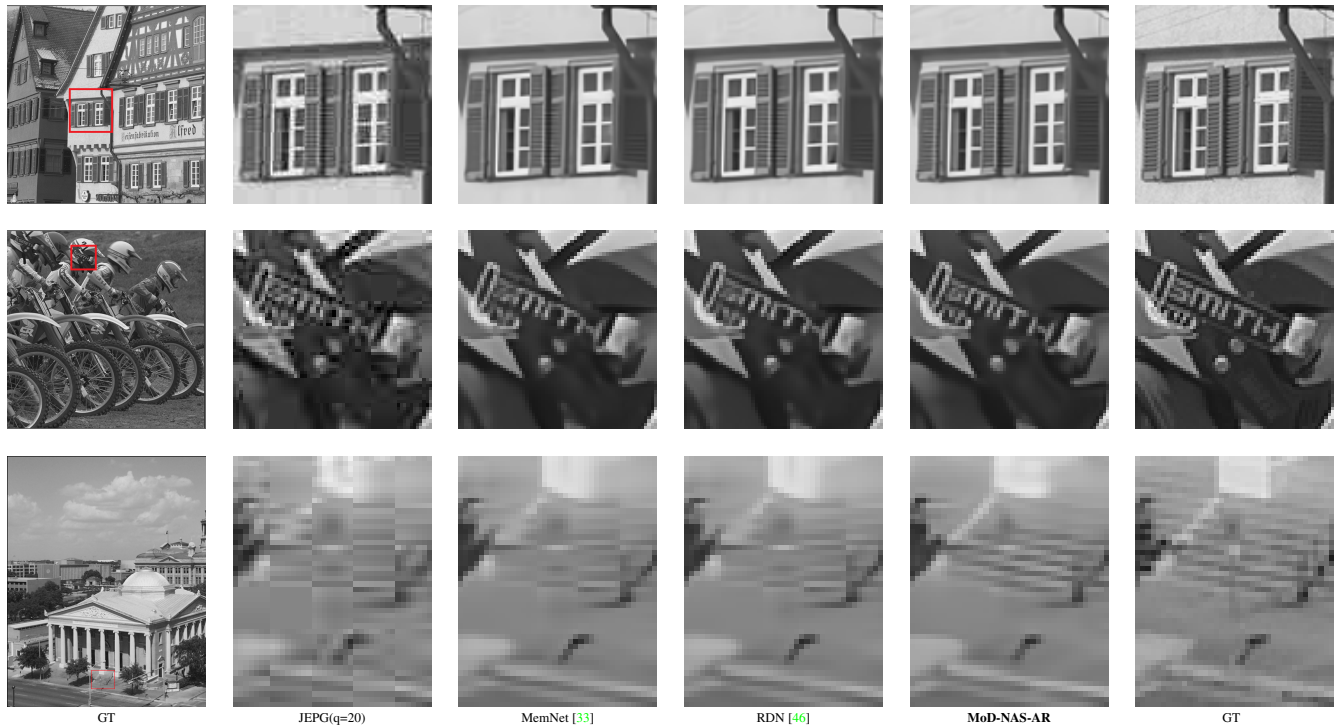| GT | JEPG(q=20) | MemNet [33] | RDN [46] | **MoD-NAS-AR** | GT |

Figure 10. Image compression Artifact reduction visual quality comparison on LIVE1 testing set(zoom in for better view).

Inputs size	Operation	kernel_size	C_{in}	C_{out}	Act	Stride
$128 \times 128 \times 1$	Conv	3×3	1	48	Relu	1
$128 \times 128 \times 48$	Conv	3×3	48	48	Relu	1
$128 \times 128 \times 48$	Conv	3×3	48	40	Relu	1
$128 \times 128 \times 40$	Residual block	3×3	40	40	Relu	1
$128 \times 128 \times 40$	Residual block	3×3	40	40	Relu	1
$128 \times 128 \times 40$	Conv	3×3	40	24	Relu	2
$64 \times 64 \times 24$	Conv	3×3	24	32	Relu	1
$64 \times 64 \times 32$	Separable Conv	5×5	32	40	Relu	1
$64 \times 64 \times 40$	skip	-	-	-	-	-
$64 \times 64 \times 40$	Nearest interpolation	-	40	32	-	-
$32 \times 32 \times 32$	Conv	5×5	32	40	Relu	1
$32 \times 32 \times 40$	skip	-	-	-	-	-
$32 \times 32 \times 40$	Residual block	3×3	40	40	Relu	1
$32 \times 32 \times 40$	Nearest interpolation	-	40	32	-	-
$16 \times 16 \times 32$	Conv	5×5	32	48	Relu	1
$16 \times 16 \times 48$	Dilated Conv	3×3	48	48	Relu	1
$16 \times 16 \times 48$	skip	-	-	-	-	-
$16 \times 16 \times 48$	Bilinear interpolation	-	48	40	-	-
$32 \times 32 \times 72$	Conv	3×3	72	36	-	1
$32 \times 32 \times 36$	skip	-	-	-	-	-
$32 \times 32 \times 36$	Dilated Conv	5×5	36	40	Relu	1
$32 \times 32 \times 40$	skip	-	-	-	-	-
$32 \times 32 \times 40$	Bilinear interpolation	-	40	40	-	-
$64 \times 64 \times 72$	Conv	3×3	72	36	-	1
$64 \times 64 \times 36$	Residual block	3×3	36	32	Relu	1
$64 \times 64 \times 32$	skip	-	-	-	-	-
$64 \times 64 \times 32$	Residual block	3×3	32	32	Relu	1
$64 \times 64 \times 32$	Bilinear interpolation	-	32	32	-	-
$128 \times 128 \times 56$	Conv	3×3	56	28	-	1
$128 \times 128 \times 28$	Residual block	3×3	28	40	Relu	1
$128 \times 128 \times 40$	Residual block	3×3	40	40	Relu	1
$128 \times 128 \times 40$	skip	-	-	-	-	-
$128 \times 128 \times 40$	Conv	1×1	40	1	Relu	1

Table 7. Architecture details of MoD-NAS-B for image denoising.

Inputs size	Operation	kernel_size	C_{in}	C_{out}	Act	Stride
$128 \times 128 \times 1$	Conv	3×3	1	48	Relu	1
$128 \times 128 \times 48$	Conv	3×3	48	48	Relu	1
$128 \times 128 \times 48$	Residual block	3×3	48	48	Relu	1
$128 \times 128 \times 48$	skip	-	-	-	-	-
$128 \times 128 \times 48$	Residual block	3×3	48	40	Relu	1
$128 \times 128 \times 40$	Conv	3×3	40	48	Relu	2
$64 \times 64 \times 48$	Dilated Conv	5×5	48	32	Relu	1
$64 \times 64 \times 32$	skip	-	-	-	-	-
$64 \times 64 \times 32$	Separable Conv	5×5	32	48	Relu	1
$64 \times 64 \times 48$	Area interpolation	-	48	48	-	-
$32 \times 32 \times 48$	skip	-	-	-	-	-
$32 \times 32 \times 48$	Residual block	3×3	48	40	Relu	1
$32 \times 32 \times 40$	skip	-	-	-	-	-
$32 \times 32 \times 40$	Conv	3×3	40	48	Relu	2
$16 \times 16 \times 48$	Separable Conv	5×5	48	48	Relu	1
$16 \times 16 \times 48$	Conv	5×5	48	32	Relu	1
$16 \times 16 \times 32$	skip	-	-	-	-	-
$16 \times 16 \times 32$	Bilinear interpolation	-	32	48	-	-
$32 \times 32 \times 96$	Conv	3×3	96	48	-	1
$32 \times 32 \times 48$	Residual block	3×3	48	48	Relu	1
$32 \times 32 \times 48$	Dilated Conv	5×5	48	48	Relu	1
$32 \times 32 \times 48$	Conv	5×5	48	32	Relu	1
$32 \times 32 \times 32$	Nearest interpolation	-	32	48	-	-
$64 \times 64 \times 96$	Conv	3×3	96	48	-	1
$64 \times 64 \times 48$	skip	-	-	-	-	-
$64 \times 64 \times 48$	Conv	3×3	48	48	Relu	1
$64 \times 64 \times 32$	Residual block	3×3	32	48	Relu	1
$64 \times 64 \times 48$	Deconvolution	3×3	48	40	Relu	2
$128 \times 128 \times 88$	Conv	3×3	88	44	-	1
$128 \times 128 \times 44$	Dilated Conv	3×3	44	40	Relu	1
$128 \times 128 \times 40$	Residual block	3×3	40	40	Relu	1
$128 \times 128 \times 40$	Conv	3×3	40	40	Relu	1
$128 \times 128 \times 40$	Conv	1×1	40	1	Relu	1

Table 8. Architecture details of MoD-NAS-AR for compression artifact reduction.