Adaptive Dynamic Filtering Network for Image Denoising

Hao Shen[1,2,4], **Zhong-Qiu Zhao**[1,2,3,4*], **Wandi Zhang**[1,2,4]

[1]School of Computer Science and Information Engineering, Hefei University of Technology (HFUT)
[2]Intelligent Interconnected Systems Laboratory of Anhui Province (HFUT)
[3]Guangxi Academy of Sciences
[4]Intelligent Manufacturing Institute of HFUT
haoshenhs@gmail.com,z.zhao@hfut.edu.cn,wandizhang@mail.hfut.edu.cn

Abstract

In image denoising networks, feature scaling is widely used to enlarge the receptive field size and reduce computational costs. This practice, however, also leads to the loss of high-frequency information and fails to consider within-scale characteristics. Recently, dynamic convolution has exhibited powerful capabilities in processing high-frequency information (*e.g.,* edges, corners, textures), but previous works lack sufficient spatial contextual information in filter generation. To alleviate these issues, we propose to employ dynamic convolution to improve the learning of high-frequency and multi-scale features. Specifically, we design a spatially enhanced kernel generation (SEKG) module to improve dynamic convolution, enabling the learning of spatial context information with a very low computational complexity. Based on the SEKG module, we propose a dynamic convolution block (DCB) and a multi-scale dynamic convolution block (MDCB). The former enhances the high-frequency information via dynamic convolution and preserves low-frequency information via skip connections. The latter utilizes shared adaptive dynamic kernels and the idea of dilated convolution to achieve efficient multi-scale feature extraction. The proposed multi-dimension feature integration (MFI) mechanism further fuses the multi-scale features, providing precise and contextually enriched feature representations. Finally, we build an efficient denoising network with the proposed DCB and MDCB, named ADFNet. It achieves better performance with low computational complexity on real-world and synthetic Gaussian noisy datasets. The source code is available at https://github.com/it-hao/ADFNet.

Introduction

Image denoising aims to recover the clean image from the observed noisy image, which is an essential step in improving image perception. In the early, most methods (Aharon, Elad, and Bruckstein 2006; Zoran and Weiss 2011) were based on image priors derived from the statistics of natural images to remove noise. However, the performance degradation will appear once the noise distribution is inconsistent with the image priors. For the past few years, convolutional neural networks (CNNs), learning the mapping from
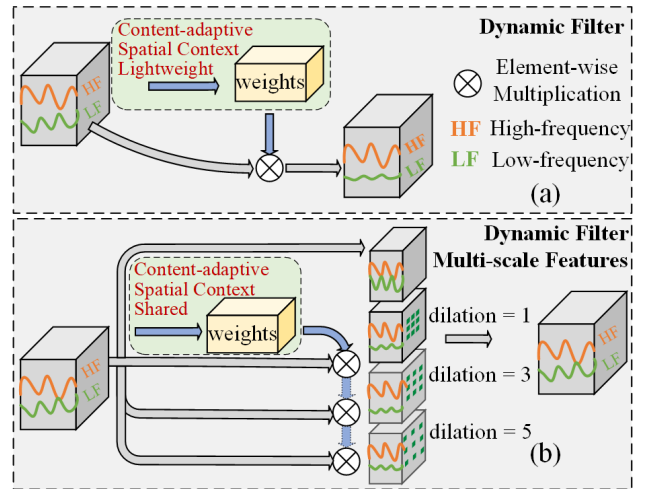

Figure 1: Dynamic convolution and multi-scale dynamic convolution schemes.

the noisy image to the corresponding clean image in a data-driven manner, have been used to perform image denoising and have achieved superior performance.

Deep CNN-based methods (Zhang et al. 2019, 2020) were recently proposed to expand the receptive field and pursue better performance. However, these methods also result in high computation consumption and time consuming, and could produce over-smoothed results with some high-frequency information (*e.g.,* edge, texture) lost. There are two reasons to explain it. First, regular static convolutions mainly focus on processing low-frequency information but are short of the capability to process high-frequency information (Hu et al. 2022). Second, feature downsampling operation is widely used in deep CNNs to reduce parameters and computation. This practice, however, inevitably degenerates high-frequency signals, thus losing some informative image details (Zou et al. 2022; Zhang et al. 2021a). Compared to static convolution, dynamic convolution has distinct properties with spatial-anisotropy and content-adaptiveness, endowing the network with powerful capabilities to restore complex and delicate high-frequency information. The kernel generation module is an essential component of dynamic convolution, which predicts the corresponding kernels based on the input features. Therefore, this module should be

highly efficient and lightweight to ensure dynamic convolution as a fundamental plug-and-play cell. As pioneer works, the kernel computation in (Ha, Dai, and Le 2016; Jia et al. 2016) introduces a large number of parameters and is difficultly extended at deep CNN-based networks. (Li et al. 2021) utilizes the depth-wise separable convolution manner to obtain relatively superiority in computational complexity but fails to use channel-specific information in kernel prediction. (Zhou et al. 2021) further design a channel filter branch to add channel-wise information to kernel generation. However, the absence of spatial information still remains. Furthermore, these works mainly focus on downstream tasks including detection and segmentation. Image denoising task requires removing noise and keeping fine-grained feature representation. Therefore, how to apply dynamic convolution to denoising networks and achieve fast and efficient denoising targets is a crucial issue.

Besides, multi-scale features have played an important role in the image denoising task, including two mainstream designs: global encoder-decoder architectures (Chang et al. 2020) and local multi-scale feature extraction module (Zamir et al. 2021). Both designs can enlarge the receptive field size of networks, thus being more semantics and robustness to noise. The former can extract cross-scale features and gradually recover high-resolution representation from coarse to fine. The latter can capture resolution-specific (within-scale) multi-scale feature representations with the help of various size contexts. However, almost existing studies cannot effectively incorporate these two manners, let alone inherit the merits of dynamic convolution and apply it to extract enriched multi-scale features.

In this work, we propose a novel Adaptive Dynamic Filtering Network (ADFNet) for image denoising, which incorporates dynamic convolution operation to tackle the above issues. To be specific, we propose a spatially enhanced kernel generation (SEKG) module to improve the dynamic convolution, realizing the interaction of spatial context information with slight additional computational complexity. The SEKG consists of spatial context extractor and channel information interaction branches, which can guide the generation of dynamic kernels, improving the high-frequency information reconstruction. Based on this module, we propose a dynamic convolution block (DCB) and a multi-scale dynamic convolution block (MDCB). The former enhances the high-frequency information via dynamic convolution (Fig. 1 (a)) and preserves low-frequency information via skip connections. The latter expands the receptive field size by introducing the idea of dilated convolution and extracts powerful multi-scale features by applying shared adaptive dynamic kernels (Fig. 1 (b)). The shared mechanism aims further to decrease the calculation of the whole network. In addition, a multi-dimension feature integration (MFI) mechanism is proposed to capture multi-scale feature interaction via cross-dimension inter-dependencies, endowing the network with a more powerful representative capability. When performing the dynamic convolution, we introduce the idea of depth-wise separable convolution to improve efficiency and reduce the model complexity. Our main contributions are summarized as follows:

- We propose a spatially enhanced kernel generation (SEKG) module to implement dynamic convolution (DConv), which incorporates the learning of spatial context to kernel generation.

- With the guidance of the SEKG module, we propose a dynamic convolution block (DCB) and a multi-scale dynamic convolution block (MDCB) to enhance high-frequency and multi-scale feature representations. Extended experiments deploying DConv and MDCB to different backbone architectures show the effectiveness of the proposed SEKG scheme.

- We propose a new encoder-decoder network, named adaptive dynamic filtering network, using the proposed modules. Extensive experiments are conducted on real-world and synthetic Gaussian noisy datasets to show effectiveness and efficiency.

Related Work

CNN-based Image Denoising. Recently, many CNN-based methods have been proposed (Shen et al. 2022; Zhang et al. 2022) and achieved state-of-the-art performance. For instance, (Zhang et al. 2017) proposed to apply the residual learning and batch normalization to facilitate the training for image denoising. (Zhang, Zuo, and Zhang 2018) utilized the tunable noise level map to guide the recovery of the noisy images. Many following works further improve the performance by deploying elaborate architecture designs, including dense connections (Zhang et al. 2020), encoder-decoder architecture (Mao, Shen, and Yang 2016), non-local attention (Zhang et al. 2019), dilated convolution (Chang et al. 2020), multi-scale design (Gou et al. 2022), and others (Ren et al. 2021). However, many of these approaches adopt static convolutions to extract features, which can cause over-smoothing artifacts. And most of them have huge network structures, leading to a large amount of computation and low inference speed. Different from the above methods, our ADFNet adopts dynamic convolution to implement high-frequency and multi-scale feature extraction and achieves competitive results with acceptable model complexity.

Dynamic Filtering. In contrast to static convolution with fixed kernels, dynamic convolution considers the image content and spatial position and can be categorized into two types. One kind (Lin et al. 2020; Chen et al. 2020; Li, Zhou, and Yao 2022) is to predict the coefficients of different convolution kernels to combine them dynamically. The other is to utilize a separate network branch to predict kernels applied to the target features. The former only increases additional computational complexity to original static convolution kernels and is therefore difficult to apply to deep networks. Regarding the latter category, (Li et al. 2021) utilized the efficient depth-wise separable convolution manner to implement the dynamic convolution operator but failed to encode channel-specific information. (Zhou et al. 2021) developed a channel filter branch, adding channel-wise information via squeeze-and-excitation structure to dynamic convolution kernels in an efficient way. However, the spatial context information cannot be fused effectively into kernel generation. Therefore, the proposed spatially enhanced ker-
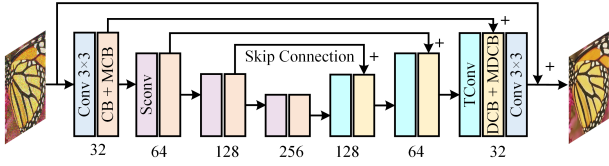
Figure 2: Illustration of our proposed adaptive dynamic filtering network (ADFNet).

nel generation module fully collects the spatial and channel context information to guide the kernel generation.

Method

Network Structure

The architecture of our proposed ADFNet is illustrated in Fig. 2. Following (Chang et al. 2020), our ADFNet adopts an encoder-decoder framework to pursue an effective and efficient target. To be detailed, the proposed ADFNet consists of four scales, each except the lowest scale has a residual connection between the encoder and decoder. In the initial stage, we utilize one 3×3 convolution (Conv) to extract shallow features from the noisy input. Considering that dynamic convolution is very sensitive to noise, in the encoder, we employ a convolution block (CB) and a multi-scale convolutional block (MCB) to extract features at four scales and filter noise. Then, in the decoder, we reconstruct the features in each scale using a dynamic convolution block (DCB) and a multi-scale dynamic convolution block (MDCB) from coarse to fine. Here, we employ one 3×3 strided Conv (SConv) and one 6×6 transposed Conv (TConv) for feature downsampling and upsampling, respectively. The number of channels in the encoder and decoder from the first scale to the fourth scale is 32, 64, 128, and 256.

Without bells and whistles, we optimize the proposed ADFNet using the L_2 loss for Gaussian image denoising and the Charbonnier loss (Charbonnier et al. 1994) for real-world image denoising.

Spatially Enhanced Kernel Generation (SEKG)

As discussed in previous sections, most of previous works discard the spatial information and fail to effectively aggregate channel-specific information. This work attempts to adopt a cost-effective way to achieve kernel generation.

As shown in Fig. 3 (a), our SEKG module is designed with a *spatial context extractor* branch and a *channel information interaction* branch. To reduce the amount of computation and the number of network parameters as much as possible, and to enhance the representation of context information. Inspired by (Shen and Zhao 2020) and (Wang et al. 2020), the spatial context extractor branch is implemented with a simple 3×3 depthwise separable convolution and the channel information interaction branch contains an average pooling followed by a convolution mapping layer. These two branches effectively exploit the relationship of inter-channel and intra-channel, respectively. In order to make full use of them, we adopt the element-wise addition operation to fuse them. Finally, we utilize one 1×1 convolution to output fea-
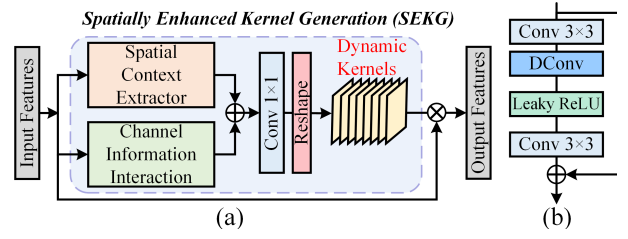

Figure 3: Illustration of (a): dynamic convolution (DConv) and (b): dynamic convolution block (DCB).

ture maps whose dimension is $(k \times k \times c) \times h \times w$. To formulate spatially-varying kernels, we reshape them into a series of per-pixel kernels $W_{i,j} \in \mathbb{R}^{k^2 \times c}$, where $i \in \{1, 2, \cdots, h\}$, $j \in \{1, 2, \cdots, w\}$. In this way, we learn a different high-pass filtering kernel for each location.

Dynamic Convolution Block (DCB)

It is well-known that high-order operators are more sensitive to disturbance, while the first-order operators are not. In other words, dynamic and plain static convolution can produce complementary representations of various degrees. Therefore, we integrate the proposed dynamic convolution (DConv) into the residual block to take advantage of their own meritc fully. As shown in Fig. 3 (b), the DConv is added after the frst convolution in the residual block. Unlike static convolution, which has fixed kernels once trained, dynamic kernels are generated by independent network branches and can be adjusted based on the input.

Dynamic Convolution (DConv). We take W as the predicted kernels to perform dynamic convolutional operators on the corresponding input feature maps. It is worth noting that each kernel in W has only c channels, not $c \times c_{out}$, since we utilize the idea of depth-wise separable convolution to decrease the number of parameters further. This manner is completely different from the existing methods (Jia et al. 2016; Zhang et al. 2021b; Ha, Dai, and Le 2016) because it actually involves channel-wise adjustment and the operation of each channel is independent.

Multi-Scale Dynamic Convolution Block (MDCB)

The above-designed DConv utilizes the content-adaptive properties and spatial context information to guide the dynamic kernel generation locally, thus processing high-frequency information better. Inspired by multi-scale convolution block (MCB) (Fig. 4 (b)) and dynamic convolution, the MDCB can (1) enlarge the receptive field size; (2) utilize adaptive dynamic kernels to filter multi-scale features globally, thus endowing the model with capabilities to generalize well on diverse noisy input. To this end, the generated kernels should be shared among multi-scale features to reduce computational complexity, which is the main distinction between MCB and our MDCB.

Multi-Scale Dynamic Feature Extraction. There are two steps to extract multi-scale dynamic features, the first step, named shared adaptive kernel generation, is to generate
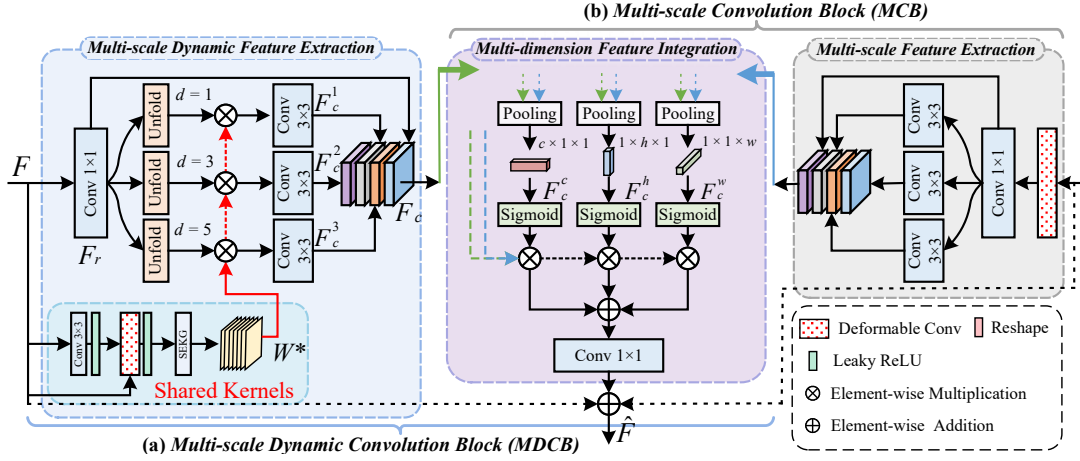
Figure 4: The detailed design of the proposed multi-scale dynamic convolution block.

shared dynamic convolutional kernels. The second aims to aggregate multi-scale features.

(1) Shared Adaptive Kernel Generation. As shown in Fig. 4 (a), we first use a modulated deformable Conv (MD-Conv) for powerful feature representation. Then, the SEKG module is embedded for the transformation between features and per-pixel kernels. As we all know, the MDConv can adaptively set the offset of each element and gather values for each component of the local feature patch, thus providing sufficient guidance for features at different positions. In our implementation, the offset is calculated by one 3×3 Conv followed by an activation function,

$$\{\Delta x, \Delta y, \Delta w\}_{(x,y)} = H(F), \qquad (1)$$

where $H(\cdot)$ denotes the convolutional operation, $\Delta x, \Delta y$ is the offset with respect to the positions (x, y), and Δw is the learnable modulation scalar. Then, the MDConv takes the input features F and learnable offsets as its input to generate the enhanced feature F_m,

$$F_m(x, y) = \sum_{j=1}^{k} w_j * F(x + \Delta x_j, y + \Delta y_j) * \Delta w_j, \quad (2)$$

where k denotes the number of samples, and w is the learnable weights. In this way, we can obtain the fine-grained features and filter unpleasant noises to avoid noise interference with the following kernel generation. Next, we utilize the proposed SEKG module to generate the shared dynamic kernels and denote as W^*.

(2) Multi-Scale Feature Aggregation. Before transforming features, we selectively to reduce the channel numbers and obtain the reduced features F_r. Then, we utilize the Unfold operation to extract sliding local 3D patches with patch-size $k = 3$, stride $s = 1$, and various dilation rates $(d = 1, 3, 5)$ from input features $F_r \in \mathbb{R}^{c \times h \times w}$, and then reshape these 3D patches to obtain three groups of new features, denoted as $F_t^i (i = 1, 2, 3) \in \mathbb{R}^{c \times k^2 \times h \times w}$. As done in the above DConv, we adopt the same process manner to aggregate each scale features. Hence, after concatenating scale-specific features $\{F_c^i\}_{i=1}^{3}$ and initial features F_r, the resulted F_c can achieve powerful multi-scale feature repre-

sentations. The entire process can be formulated as

$$\begin{aligned} F_c^i &= C_{3\times3}^i (F_t^i \otimes W^*), \\ F_c &= Concat(F_r, F_c^1, F_c^3, F_c^3), \end{aligned} \qquad (3)$$

where $\otimes$ denotes the element-wise multiplication operation, $C_{3\times3}^i (i = 1, 2, 3)$ is three independent 3×3 Convs which are used to interwove with the dynamic multi-scale features, $Concat(\cdot)$ is the channel concatenation function, and F_c denotes the final output.

Multi-Dimension Feature Integration. Regarding feature integration, inspired by the visual attention mechanism, we perform integration through a multi-dimension attention-guided mechanism. Specifically, we obtain three groups of contextual information by average-pooling operation on different dimensions of feature maps. We use channel-wise attention $((h, w)$ dimension) as an instance. An average-pooling operation along the channel axis is firstly employed on F_c to generate an efficient feature descriptor. Based on the feature descriptor, we then apply a sigmoid function to generate a spatial attention map ranging from 0 to 1, which is used to encode where to highlight or suppress. Finally, we perform an multiplication operation between F_c and descriptor to obtain spatial-wise attentive features F_c^c. Likewise, we can obtain another two attention maps from (c, h) and (c, w) dimensions. To fully take advantage of them and explore the cross-dimension interaction between channel and spatial contexts, we take the above obtained F_c^c as the calibrated features and then multiply them with other two attention maps, respectively, to acquire two new attentive features F_c^w and F_c^h. Formally, we have

$$\begin{aligned} F_c^c &= Sigmoid(AvgPool(F_c)) \otimes F_c, \\ F_c^w &= Sigmoid(AvgPool(F_c)) \otimes F_c^c, \\ F_c^h &= Sigmoid(AvgPool(F_c)) \otimes F_c^c. \end{aligned} \qquad (4)$$

Finally, we add three attentive features and employ a convolutional layer to fuse them.

$$\hat{F} = F + C_{1\times1}(F_c^c + F_c^w + F_c^h), \qquad (5)$$

Datasets	Noise Level	DnCNN	FFDNet	RNAN	RIDNet	RDN	SADNet	DeamNet	P3AN	MSANet	**ADFNet**
Kodak24	30	31.39	31.39	31.86	31.64	31.94	31.86	31.88	31.99	31.78	**32.01**
	50	29.16	29.10	29.58	29.25	29.66	29.64	29.70	29.69	29.57	**29.81**
	70	27.64	27.68	28.16	27.94	28.20	28.28	28.30	28.25	28.17	**28.48**
BSD68	30	30.40	30.31	30.63	30.47	30.67	30.64	30.70	30.72	30.67	**30.74**
	50	28.01	27.96	28.27	28.12	28.31	28.32	28.34	28.37	28.36	**28.43**
	70	26.56	26.53	26.83	26.69	26.85	26.93	26.95	26.94	26.96	**27.03**
MCMaster	50	28.62	29.18	29.18	-	29.60	29.72	29.78	-	29.82	**29.96**

Table 1: Average PSNR (dB) results of different methods for color image denoising on various datasets.
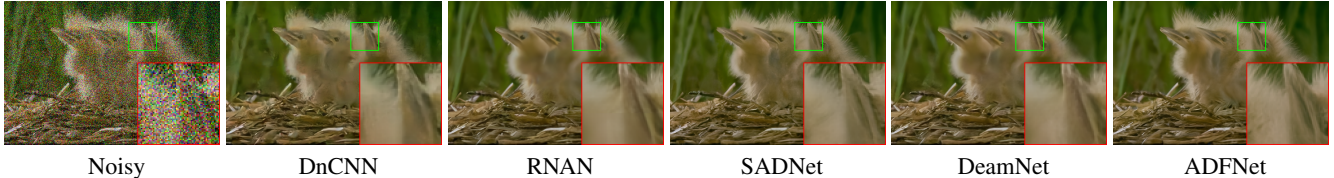

Figure 5: Visual comparisons of various methods for color gaussian image denoising. The noisy image is corrupted by additive white gaussian noise (AWGN) with noise level $\sigma = 50$.

where $C_{1\times 1}$ denotes the 1×1 Conv, $\hat{F}$ is the final output of the MDCB. Here, we adopt local residual learning to further improve the information flow.

By capturing the cross-dimension inter-dependencies, the extracted multi-scale dynamic features can be fused further and show a more powerful representative capability.

Experiments

Experimental Setup

Datasets. In this work, we use 800 images from DIV2K (Timofte et al. 2017) as training data for Gaussian image denoising. For real image denoising, the Smart-phone Image Denoising Data set (SIDD) (Abdelhamed, Lin, and Brown 2018) is chosen to train models. There are 30K real noisy images in the SIDD, 320 image pairs for training, and 1280 images for validation. We randomly crop 320×300 image patches with a size of 256×256 to train the model due to the large size of these primitive real images.

For Gaussian image denoising, we employ BSD68 (Martin et al. 2001), Kodak24 (Russakovsky et al. 2015), and McMaster (Zhang et al. 2011) as test sets. For real image denoising, Darmstadt Noise Dataset (DND) (Plotz and Roth 2017) and SIDD (Abdelhamed, Lin, and Brown 2018) (*e.g.,* benchmark and validation sets) are chosen for testing. It is worth noting that due to the ground truths of the real images are not publicly available at present, we obtain the results of quantitative assessment via the online submission system.

Implementation Details. We implement ADFNet in PyTorch framework and conduct all experiments on NVIDIA 2080-Ti GPUs. During training, we employ ADAM (Kingma and Ba 2014) optimizer that the parameters are set as $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ to learn the optimal parameters. We randomly crop 16 patches of size 128×128 as input for each training mini-batch and augment the patches by flipping horizontally or vertically and rotating $90°$. The learning rate is initialized to 10^{-4} and then linearly

Methods	FLOPs	Time	Kodak24	McMaster
SwinIR*	186	0.472	29.79	30.22
MalleNet*	-	-	29.61	**30.23**
ADFNet	**13.9**	**0.027**	29.81	29.96
ADFNet*	46.2	0.057	**29.92**	30.12

Table 2: Comparisons of performance and model complexity between SOTA methods. '*' denotes models trained on DIV2K + Flickr2k + WED + BSD400 datasets.

decreases to half every 2×10^5 iterations.

Color Gaussian Image Denoising

To verify the effectiveness of the proposed ADFNet for Gaussian noisy images, we compare it with the existing methods include DnCNN (Zhang et al. 2017), FFD-Net (Zhang, Zuo, and Zhang 2018), RNAN (Zhang et al. 2019), RIDNet (Tian, Xu, and Zuo 2020), RDN (Zhang et al. 2020), SADNet (Chang et al. 2020), DeamNet (Ren et al. 2021), P3AN (Hu et al. 2021), and MSANet (Gou et al. 2022). The compared PSNR values are listed in Table 1. From the quantitative results, we observe that our ADFNet consistently achieves better PSNR values on all datasets. Taking the Kodak24 dataset as an example, our model obtains a performance gain of 0.13 dB, 0.11 dB, and 0.18 dB over the DeamNet on three different noise levels. More importantly, Deamnet employs an additional BSD400 dataset to train the model. These persistent performance improvements also demonstrate that our ADFNet has powerful capabilities to restore strong corrupted images. Besides, some CNN- and Transformer-based approaches (Liang et al. 2021; Jiang et al. 2022) enrich training sets to boost model performance. As such, we also utilize more data to train a novel model named **ADFNet***, which is built by setting the channel numbers from the first to the last scale as 64, 128, 256, and 512. From Table 2, we can see the SwinIR (Liang et al.

Datasets	DnCNN	CBDNet*	RIDNet	AINDNet*	VDN	MPRNet	DeamNet*	DAGL	MSANet*	**ADFNet**
SIDD	38.56	38.68	38.71	39.08	39.28	39.71	39.47	38.94	39.56	**39.79**
Validation	0.910	0.909	0.951	0.954	0.956	0.958	0.955	0.953	0.912	**0.960**
SIDD	23.66	33.28	-	38.08	39.26	-	39.35	-	-	**39.63**
Benchmark	0.583	0.868	-	0.953	0.955	-	0.955	-	-	**0.958**
DND	32.43	38.06	39.26	39.53	39.38	39.80	39.63	39.77	39.65	**39.87**
	0.790	0.942	0.953	0.956	0.952	0.954	0.953	**0.956**	0.955	0.955

Table 3: Average PSNR (dB) and SSIM results of different methods for real-world image denoising on DND dataset and SIDD benchmark datasets. '*' denotes models trained on additional traing sets.

2021) takes more than $\times \mathbf{8}$ time and $\times \mathbf{4}$ FLOPs to process one 128×128 image compared to our model. However, the PSNR value on Kodak24 is still lower than ours. Although our methods cannot achieve the best results on all benchmarks, the superiority of inference time and computational complexity are apparent.

We visually compare our denoised images with some competitive approaches. As shown in Fig. 5, the image generated by our ADFNet preserves rich details, while RNAN and DeamNet introduce some smoothness.

Real-World Image Denoising

We compare our ADFNet with recent leading methods for real-world image denoising, such as DnCNN (Zhang et al. 2017), CBDNet (Guo et al. 2019), RIDNet (Anwar and Barnes 2019), AINDNet (Kim et al. 2020), VDN (Yue et al. 2019), MPRNet (Zamir et al. 2021), DeamNet (Ren et al. 2021), DAGL (Mou, Zhang, and Wu 2021), and MSANet (Gou et al. 2022). The results of different methods are provided in Table 3. We can conclude that ADFNet achieves the best PSNR results on three benchmarks, including the newly proposed DAGL and MSANet. It is worth noting that some methods train models with more training sets but achieve lower performance. Overall, the performance superiority of ADFNet confirms the validity and rationality of our network design.

Evaluation on Model Complexity. We evaluate the model complexity of some representative methods in the last three years for real-world image denoising. The FLOPs and running time are chosen as the primary evaluation metrics. The FLOPs are calculated on one 128×128 resolution RGB patch. The running time is evaluated on the DND dataset, and each result is obtained by repeating five experiments to ensure a fair comparison. It can be seen from Fig. 6(a) that DANet (Yue et al. 2020) has the lowest FLOPs, but the performance is far worse than our method. MPRNet (Zamir et al. 2021) is close to the performance of our ADFNet, but the FLOPs cost is more than $\times \mathbf{10}$ ours. Next, we study the trade-off between the running time and the performance of various methods. As shown in Fig. 6(b), RIDNet (Anwar and Barnes 2019) has a similar execution time compared to our method, but our ADFNet surpasses its performance by a large margin from **39.23 dB** to **39.87 dB**. Further, compared to the top methods like MPRNet and DAGL (Mou, Zhang, and Wu 2021), we have a slight performance improvement in the PSNR but run nearly $\times \mathbf{4}$ faster. Therefore, compared with these approaches, our ADFNet achieves a better trade-
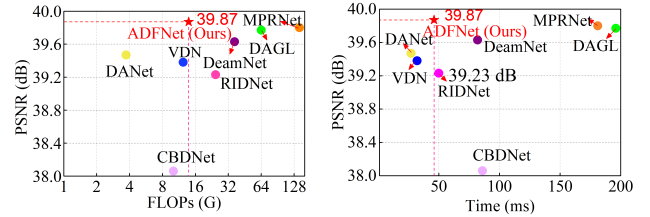

Figure 6: Trade-off between the performance, FLOPs and running time for real-world image denoising. The PSNR scores are evaluated on the DND dataset and the FLOPs are obtained by calculating one 128×128 patch.

off between performance and FLOPs cost.

Extension: DConv and MDCB Used in Backbones

To further demonstrate the effectiveness of the proposed DConv and MDCB, we conduct a group of experiments by plugging them into existing well-designed network architectures. Here, we mainly choose two representative methods, DnCNN (Zhang et al. 2017) and RDN (Zhang et al. 2020), as the backbone networks for color Gaussian image denoising. Specifically, we manually build novel networks based on the original two network architectures by adjusting the number of channels and network depth. For instance, we design lightweight and effective DnCNN by setting the depth = (5, 10, 15) and channel = (32, 64). Likewise, we implement deep network RDN by controlling block numbers as (3, 6, 9) and channels as (16, 32). As comparisons, we reimplement the DConv-enhanced DnCNN by replacing one regular static 3×3 convolution with one DConv in the middle of the network. The MDCB-enhanced RDN is developed by replacing one residual dense block with three MDCBs in the middle of the backbone network.

As shown in Fig. 7, the DConv-enhanced DnCNN achieves performance gains by 0.60 dB, 0.24 dB, and 0.15 dB, respectively, with the increases in the network depth. According to these observations, we can draw the following conclusions. First, the DConv- and MDCB- enhanced corresponding versions can achieve better performance in the lightweight and deep networks. Second, the performance superiority is decreased with the increases of the channels and depth, but stacking more modules may bring adequate performance improvements further. Third, the SEKG-guided modules enhance the representational capabilities of networks and can serve as a normal layer or an essential feature extraction module, mainly attributed to the generated kernels fully utilizing the spatial context information.
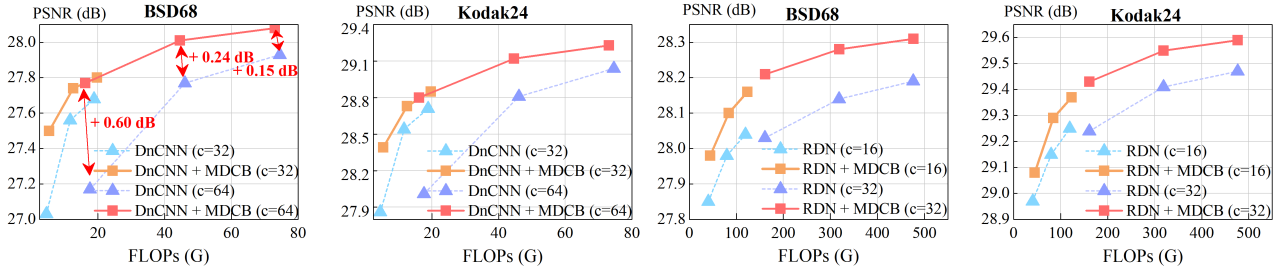
Figure 7: Applying DConv and MDCB to different backbones. The DConv-enhanced DnCNN is built by setting depth = (5, 10, 15) and channel = (32, 64). The MDCB-enhanced RDN is built by setting block number = (3, 6, 9) and channel = (16, 32).

Model	Params	Kodak24	BSD68
w/o (DCB and MDCB)	7.71	29.61	28.33
w/o DCB	7.64	29.69	28.36
w/o MDCB	7.72	29.66	29.37
w/o MFI	7.65	29.77	28.40
w/ (DCB & MDCB) (all)	**7.30**	29.77	28.38
w/ (DCB & MDCB) (encoder)	7.65	29.73	28.38
w/ (DCB & MDCB) (decoder) ADFNet	7.65	**29.81**	**28.43**

Table 4: Ablation study on different components.

Ablation Studies

We study the impact of each component on the performance of our ADFNet. All experiments are performed on the color Gaussian image denoising with noise level $\sigma = 50$. The ablation models are trained on image patches of the size of 128×128 with the same number of iterations.

Effectiveness of SEKG Module. As previously mentioned, the proposed DCB and MDCB are designed based on the SEKG module. To demonstrate the validity of the SEKG, we replaced the DCB with a regular 3×3 convolution block, denoted as *w/o DCB*. From Table 4, we see the model has a significant performance degradation on all benchmarks, but the number of parameters and FLOPs is almost the same as our ADFNet. We also replace the MDCB in the decoder with a static multi-scale convolution block (Fig. 4). This block is composed of a modulated deformable convolution and three parallel 3×3 convolutions with different dilation rates, and the main distinction lies in the kernels of our MDCB generated by an individual kernel generation module. As shown in Table 4, we observe the *w/o MDCB* obtains worse performance, while our ADFNet obtains better results. The performance degrades further when the DCB and MDCB are replaced simultaneously, indicating that the proposed SEKG promotes kernel generation. To have better insight, we visualize the average feature maps after applying the DCB and MDCB. From the Fig. 8, we have two observations. First, the DConv used in DCB is able to highlight the high-frequency information, and the low-frequency information can be incorporated via skip connections. Second, the shared dynamic kernels used in MDCB can enhance the multi-scale feature representations. These indicate that the joint low- and high-frequency information structure is reasonable to deploy in the decoder to reconstruct features.

Since dynamic convolution is very sensitive to noise, we only deploy the proposed modules in the decoder. When the
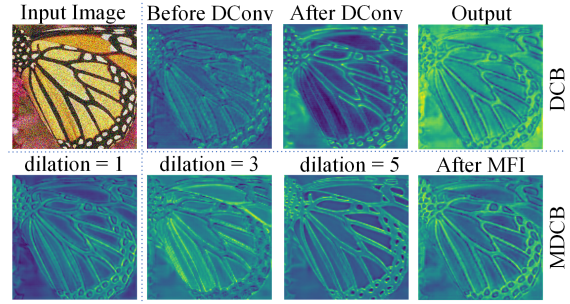

Figure 8: Visualization of feature maps extracted by the first DCB and MDCB and their inner convolutions.

DCB and MDCB are deployed in the phase of encoding, the performance is lower than our proposed scheme. This phenomenon is consistent with our analysis in the previous section, that is, noisy kernels are generated during the encoder period, and more robust kernels are generated in the decoding phase with the removal of noise.

Effectiveness of MFI Mechanism. To evaluate the effectiveness of the module MFI, we remove it from the architecture of our ADFNet for comparison, denoted as *w/o MFI*. Thus, the produced dynamic multi-scale features are concatenated along the channel dimension but omit the fully multi-dimension feature fusion. From Table 4, we see an apparent degradation in terms of PSNR score on three datasets, indicating the MFI can accurately discover the correlation of multi-scale features and complete the information interaction among different feature dimensions to achieve better representations. Fig. 8 shows the feature maps after employing the MFI mechanism in MDCB, and we can see the attention mechanism has a noticeable effect on modulating the activation values. These results demonstrate the effectiveness of the fusion mechanism.

Conclusions

In this work, we propose a spatially enhanced kernel generation (SEKG) module to improve the dynamic convolution while maintaining low computational costs. With the proposed module, we further present a dynamic convolution block and a multi-scale dynamic convolution block to improve the representations of high-frequency and multi-scale features. Therefore, benefiting these modules, the proposed ADFNet can effectively remove noise and preserve fine-grained image details as verified in experiments.

Acknowledgments

References

Abdelhamed, A.; Lin, S.; and Brown, M. S. 2018. A high-quality denoising dataset for smartphone cameras. In *CVPR*, 1692–1700.

Aharon, M.; Elad, M.; and Bruckstein, A. 2006. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *TSP*, 54(11): 4311–4322.

Anwar, S.; and Barnes, N. 2019. Real image denoising with feature attention. In *ICCV*, 3155–3164.

Chang, M.; Li, Q.; Feng, H.; and Xu, Z. 2020. Spatial-adaptive network for single image denoising. In *ECCV*, 171–187.

Charbonnier, P.; Blanc-Feraud, L.; Aubert, G.; and Barlaud, M. 1994. Two deterministic half-quadratic regularization algorithms for computed imaging. In *ICIP*, 168–172.

Chen, Y.; Dai, X.; Liu, M.; Chen, D.; Yuan, L.; and Liu, Z. 2020. Dynamic convolution: Attention over convolution kernels. In *CVPR*, 11030–11039.

Gou, Y.; Hu, P.; Lv, J.; and Peng, X. 2022. Multi-Scale Adaptive Network for Single Image Denoising. *arXiv preprint arXiv:2203.04313*.

Guo, S.; Yan, Z.; Zhang, K.; Zuo, W.; and Zhang, L. 2019. Toward convolutional blind denoising of real photographs. In *CVPR*, 1712–1722.

Ha, D.; Dai, A.; and Le, Q. V. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Hu, X.; Chen, X.; Ni, B.; Li, T.; and Liu, Y. 2022. Bi-volution: A Static and Dynamic Coupled Filter. In *AAAI*, 960–968.

Hu, X.; Ma, R.; Liu, Z.; Cai, Y.; Zhao, X.; Zhang, Y.; and Wang, H. 2021. Pseudo 3D Auto-Correlation Network for Real Image Denoising. In *CVPR*, 16175–16184.

Jia, X.; De Brabandere, B.; Tuytelaars, T.; and Gool, L. V. 2016. Dynamic filter networks. *NeurIPS*, 29.

Jiang, Y.; Wronski, B.; Mildenhall, B.; Barron, J.; Wang, Z.; and Xue, T. 2022. Fast and High-Quality Image Denoising via Malleable Convolutions. *arXiv preprint arXiv:2201.00392*.

Kim, Y.; Soh, J. W.; Park, G. Y.; and Cho, N. I. 2020. Transfer learning from synthetic to real-noise denoising with adaptive instance normalization. In *CVPR*, 3482–3492.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, C.; Zhou, A.; and Yao, A. 2022. Omni-dimensional dynamic convolution. In *ICLR*.

Li, D.; Hu, J.; Wang, C.; Li, X.; She, Q.; Zhu, L.; Zhang, T.; and Chen, Q. 2021. Involution: Inverting the Inherence of Convolution for Visual Recognition. In *CVPR*, 12316–12325.

Liang, J.; Cao, J.; Sun, G.; Zhang, K.; Van Gool, L.; and Timofte, R. 2021. Swinir: Image restoration using swin transformer. In *ICCVW*, 1833–1844.

Lin, X.; Ma, L.; Liu, W.; and Chang, S.-F. 2020. Context-gated convolution. In *ECCV*, 701–718.

Liu, Y.; Qin, Z.; Anwar, S.; Ji, P.; Kim, D.; Caldwell, S.; and Gedeon, T. 2021. Invertible denoising network: A light solution for real noise removal. In *CVPR*, 13365–13374.

Mao, X.; Shen, C.; and Yang, Y.-B. 2016. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NeurIPS*, 2802–2810.

Martin, D.; Fowlkes, C.; Tal, D.; and Malik, J. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 416–423.

Mou, C.; Zhang, J.; and Wu, Z. 2021. Dynamic attentive graph learning for image restoration. In *ICCV*, 4328–4337.

Plotz, T.; and Roth, S. 2017. Benchmarking denoising algorithms with real photographs. In *CVPR*, 1586–1595.

Ren, C.; He, X.; Wang, C.; and Zhao, Z. 2021. Adaptive consistency prior based deep network for image denoising. In *CVPR*, 8596–8606.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV*, 115(3): 211–252.

Shen, H.; and Zhao, Z.-Q. 2020. Mid-Weight Image Super-Resolution with Bypass Connection Attention Network. In *ECAI*, 2760–2767.

Shen, H.; Zhao, Z.-Q.; Liao, W.; Tian, W.; and Huang, D.-S. 2022. Joint operation and attention block search for lightweight image restoration. *Pattern Recognition*, 132: 108909.

Tian, C.; Xu, Y.; and Zuo, W. 2020. Image denoising using deep CNN with batch renormalization. *Neural Networks*, 121: 461–473.

Timofte, R.; Agustsson, E.; Van Gool, L.; Yang, M.-H.; and Zhang, L. 2017. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPRW*, 114–125.

Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; and Hu, Q. 2020. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. In *CVPR*, 11531–11539.

Yue, Z.; Yong, H.; Zhao, Q.; Meng, D.; and Zhang, L. 2019. Variational denoising network: Toward blind noise modeling and removal. *NeruIPS*, 32.

Yue, Z.; Zhao, Q.; Zhang, L.; and Meng, D. 2020. Dual adversarial network: Toward real-world noise removal and noise generation. In *ECCV*, 41–58.

Zamir, S. W.; Arora, A.; Khan, S.; Hayat, M.; Khan, F. S.; and Yang, M.-H. 2021. Multi-stage progressive image restoration. In *CVPR*, 14821–14831.

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

Zhang, K.; Zuo, W.; and Zhang, L. 2018. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. *TIP*, 27(9): 4608–4622.

Zhang, L.; Wu, X.; Buades, A.; and Li, X. 2011. Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. *Journal of Electronic imaging*, 20(2): 023016.

Zhang, Y.; Li, D.; Law, K. L.; Wang, X.; Qin, H.; and Li, H. 2022. IDR: Self-Supervised Image Denoising via Iterative Data Refinement. In *CVPR*, 2098–2107.

Zhang, Y.; Li, K.; Li, K.; Sun, G.; Kong, Y.; and Fu, Y. 2021a. Accurate and fast image denoising via attention guided scaling. *TIP*, 30: 6255–6265.

Zhang, Y.; Li, K.; Li, K.; Zhong, B.; and Fu, Y. 2019. Residual Non-local Attention Networks for Image Restoration. In *ICLR*, 1–18.

Zhang, Y.; Tian, Y.; Kong, Y.; Zhong, B.; and Fu, Y. 2020. Residual dense network for image restoration. *TPAMI*, 43(7): 2480–2495.

Zhang, Y.; Wei, D.; Qin, C.; Wang, H.; Pfister, H.; and Fu, Y. 2021b. Context reasoning attention network for image super-resolution. In *CVPR*, 4278–4287.

Zhou, J.; Jampani, V.; Pi, Z.; Liu, Q.; and Yang, M.-H. 2021. Decoupled Dynamic Filter Networks. In *CVPR*, 6643–6652.

Zoran, D.; and Weiss, Y. 2011. From learning models of natural image patches to whole image restoration. In *ICCV*, 479–486.

Zou, X.; Xiao, F.; Yu, Z.; Li, Y.; and Lee, Y. J. 2022. Delving deeper into anti-aliasing in convnets. *IJCV*, 1–15.