Chapter 4
Convolutional Neural Networks for Image Denoising and Restoration

Wangmeng Zuo, Kai Zhang and Lei Zhang

Abstract With the tremendous progress of convolutional neural networks (CNNs), recent years have witnessed a dramatic upsurge of exploiting CNN toward solving image denoising. Compared to traditional model-based methods, CNN enjoys the principal merits of fast inference and good performance. In this chapter, brief survey and discussions are also given to CNN-based denoising methods from the aspects of effectiveness, interpretability, modeling ability, efficiency, flexibility, and applicability. Then, we provide a gentle introduction of CNN-based denoising methods by presenting and answering the following three questions: (i) can we learn a deep CNN for effective image denoising, (ii) can we learn a single CNN for fast and flexible non-blind image denoising, and (iii) can we leverage CNN denoiser prior to versatile image restoration tasks. Finally, we point out that image denoising remains far from solved. The real image noise is much more sophisticated than additive white Gaussian noise, making the existing CNN denoisers generally perform poorly on real noisy images. As a result, it is still very challenging and valuable to study the issues such as noise modeling, acquisition of noisy-clean image pairs and unsupervised CNN learning for real image denoising.

W. Zuo (✉) · K. Zhang
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
e-mail: wmzuo@hit.edu.cn

K. Zhang
e-mail: cskaizhang@gmail.com

L. Zhang
Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
e-mail: cslzhang@comp.polyu.edu.hk

M. Bertalmío (ed.), *Denoising of Photographic Images and Video*,
Advances in Computer Vision and Pattern Recognition,
https://doi.org/10.1007/978-3-319-96029-6_4

4.1 Introduction

During past decade, driven by the easy access to large-scale dataset, efficient training implementation on modern powerful GPUs and the advances in deep learning methods such as Rectified Linear Unit (ReLU) [22], network initialization [14], stochastic gradient-based optimization [20], batch normalization [17] and residual learning [15], the convolutional neural networks have shown great success in handling various low-level vision tasks. Concurrently, several attempts have been made to exploit CNN for image denoising due to the following reasons. First, the inference can be very efficient due to the parallel computation ability of GPU. Second, deep CNN exhibits powerful modeling capacity and can be trained without knowing the explicit degradation model when abundant noisy/clean image pairs are provided. Thus, the denoising performance can be easily boosted. Third, CNN exploits the external prior which is complementary to the internal prior of many existing denoisers such as nonlocal similarity (NSS). In other words, its combination with NSS is expected to further improve the performance. Fourth, great progress in designing CNN can facilitate the flexibility and practicability in real applications.

The goal of image denoising is to recover a latent clean image $\mathbf{x}$ from its noisy observation $\mathbf{y}$ which follows the image degradation model $\mathbf{y} = \mathbf{x} + \mathbf{v}$. One common assumption is that $\mathbf{v}$ is the additive white Gaussian noise (AWGN) with known noise level σ. Due to the ill-posed nature of denoising, regularization needs to be imposed to constrain the solution. From a Bayesian perspective, the solution $\hat{\mathbf{x}}$ can be obtained by solving the Maximum A Posteriori (MAP) model,

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}) \tag{4.1}$$

where $\log p(\mathbf{y}|\mathbf{x})$ represents the log-likelihood term, and $\log p(\mathbf{x})$ models the prior of $\mathbf{x}$ which is independent of $\mathbf{y}$. More formally, Eq. (4.1) can be reformulated as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{y}\|^2 + \lambda\Phi(\mathbf{x}) \tag{4.2}$$

where $\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{y}\|^2$ is the data fidelity term, $\Phi(\mathbf{x})$ is the regularization term (or prior term) and λ is the trade-off parameter. It is worth noting that in practice λ governs the compromise between noise reduction and detail preservation. When it is too small, some noise will be retained in the denoising result; in contrast, when λ is too large, small-scale details will also be smoothed out along with the suppression of noise.

Generally, the methods to solve Eq. (4.2) can be divided into two main categories, model-based optimization methods and discriminative learning methods. Model-based optimization directly solves Eq. (4.2) with some optimization algorithms which usually involve a time-consuming iterative inference. On the contrary, discriminative learning methods try to learn a compact inference through an optimization of a loss function on a training set containing degraded-clean image pairs [4, 8, 38, 43]. The CNN-based denoising methods belong to this category. Specifically, CNN-based

denoising methods can be further divided into MAP-CNN based and generic CNN-based denoising methods.

MAP-CNN based denoising methods refer in particular to the MAP inference based approaches which involve a series of convolution operations. Instead of first learning the prior (e.g., high-order MRF priors) and then performing the inference, this category of methods aims to learn the prior parameters along with a compact unrolled inference through solving a bi-level optimization problem [4]. With a slight abuse of notation, the objective can be given by

$$\min_{\Theta} \ell(\hat{\mathbf{x}}(\Theta), \mathbf{x}) \quad s.t. \quad \hat{\mathbf{x}}(\Theta) = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{x}\|^2 + \lambda\Phi(\mathbf{x}) \tag{4.3}$$

where Θ denotes the trainable parameters in the inference, $\ell(\hat{\mathbf{x}}, \mathbf{x})$ measures the similarity between estimated clean image $\hat{\mathbf{x}}$ and ground-truth clean image $\mathbf{x}$. While such kind of approaches are not directly connected with CNN, their unrolled inferences actually can be viewed as CNN variants with stagewise architecture. Following the pioneer work of fields of experts [34], Barbu [4] trained a discriminative Markov random field (MRF) model together with a gradient descent inference for image denoising. Samuel and Tappen [36] independently proposed a compact gradient descend inference learning framework, and discussed the advantages of discriminative learning over model-based optimization. Sun and Tappen [42] proposed a novel nonlocal range MRF (NLR-MRF) framework, and employed the gradient-based discriminative learning method to train the model. Chen and Pock [8] further proposed a trainable nonlinear reaction–diffusion (TNRD) model through discriminative learning of a compact gradient descent inference. Lefkimmiatis [25] adopted a proximal gradient-based denoising inference from a variational model to incorporate the NSS prior.

Discriminative inference learning methods enjoy some merits. First, they are efficient due to much fewer inference steps. Second, they have better interpretability because the discriminative architecture is derived from optimization algorithms such as half quadratic splitting and gradient descent [4, 8, 36, 38, 42]. However, the interpretability may come at the expense of performance since the learned priors and inference procedure are limited by the form of the MAP model [50]. In addition, the unrolled inference actually can be viewed as a network with stagewise architecture, which restricts the dataflow in each immediate output layer [51].

Instead of modeling image priors explicitly, the generic CNN-based denoising methods learn a predefined nonlinear function consisting of various CNN building blocks to model image prior implicitly and can be modeled by

$$\min_{\Theta} \ell(\hat{\mathbf{x}}, \mathbf{x}) \quad s.t. \quad \hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \sigma; \Theta). \tag{4.4}$$

The use of CNN for image denoising can be traced back to [18], where a five-layer network with sigmoid nonlinearity was developed. During the past few years, various generic CNN-based denoising methods have been proposed and the denoising

performance has been greatly boosted in comparison to [18]. In this chapter, we focus on this type of denoising methods, and assume that the noise type is AWGN with the known noise level σ.

4.1.1 Recent Advances

In addition to DnCNN and FFDNet described in this chapter, there have been several other attempts to design CNN for image denoising. Some works propose to improve convolutional filters or nonlinearities for better trade-off between effectiveness and efficiency. Wang et al. [47] developed a dilated residual CNN for fast Gaussian denoising. The main idea is to enlarge receptive field by dilated filter with different dilation factors. They showed that the expansion of receptive field can boost the denoising performance. In another denoising work by Wang et al. [46], they proposed a denoising network which uses exponential linear unit (ELU) as the nonlinearity. To better accommodate ELU and batch normalization layer, they further designed a novel structure by incorporating 1×1 convolutional layer. Kligvasser et al. [21] proposed a learnable nonlinear function with spatial connections as activation unit to replace the widespread per-pixel activation units such as ReLUs and sigmoids. They showed that the activation unit can enable CNN to capture much more complex features, thus leading to better denoising results.

While CNN-based denoising methods are effective, they lack the explicit ability to handle images with regular and repetitive patterns. Some researchers resort to incorporate the nonlocal self-similarity prior into CNN. Ahn and Cho [1] proposed a block matching convolutional neural network which combines nonlocal self-similarity prior and CNN for image denoising. The main idea is to group similar local patches into a tensor and then feed it to CNN for denoising. Bae et al. [2] proposed a new denoising network motivated from a novel persistent homology analysis on residual learning for image processing tasks. Specifically, they showed that the residual manifold is topologically simpler than the original image manifold and the wavelet transform can provide topologically simpler manifold structures. Yang and Sun [48] proposed a BM3D-inspired convolutional neural network (BM3D-Net) for image denoising. BM3D-Net directly builds the convolutional neural network by faithfully implementing the transform domain collaborative filtering in the BM3D framework. Lefkimmiatis [26] proposed a novel network architecture which involves convolutional layers as a core component and nonlocal filtering layers to exploit the inherent nonlocal self-similarity property of natural images. By training the denoiser with a wide range of different noise levels, the networks do not need to know the noise level of the noisy image and are very robust when the noise model does not match the statistics of the one used during training.

To design a principled network architecture, an interesting line of denoising methods has focused on incorporating CNN building blocks into MAP-based unrolled inference. Kim et al. [19] proposed a deeply aggregated alternating minimization (DeepAM) based on two of the steps in the conventional AM algorithm: proximal

mapping and β-continuation. The DeepAM framework enables the convolutional neural networks to operate as a regularizer in the AM algorithm for image denoising. Vogel and Pock [45] proposed a primal-dual network for image denoising that leverages the algorithmic structure provided by energy optimization techniques into learning a generalized optimization algorithm.

There also exist some works that focus on class-aware image denoising, generic image denoising and boosting-based image denoising. Remez et al. [31] pointed out a denoiser is aware of the type of image content and proposed a new fully convolutional deep neural network architecture for image denoising. They further showed that the performance can be significantly improved by fine-tuning the denoiser for images belonging to a specific semantic class. Santhanam et al. [37] developed a recursively branched deconvolutional network (RBDN) for image denoising as well as generic image-to-image regression. To integrate multiple weak denoisers with different capabilities to denoise complex scenes, Choi et al. [9] proposed a consensus neural network (ConsensusNet) which comprises a weighting stage to weigh the relevance of the individual denoisers and a boosting neural network to recover the lost features as well as improve contrast. They studied ConsensusNet on various scenarios, including the integration of denoisers with different noise levels, different image classes, and different denoiser types. Experimental results show that ConsensusNet can consistently improve denoising performance for both deterministic denoisers and neural network denoisers.

Apart from single image denoising, multi-image denoising has also attracted considerable interest. Godard et al. [12] proposed a recurrent fully CNN to handle an arbitrary number of noisy input frames for burst denoising. Instead of directly predicting the final denoised pixel values [3], Mildenhall et al. [29] proposed a CNN architecture to predict spatially varying weighting kernels of different frames. They further proposed a synthetic data generation approach based on signal-dependent Gaussian noise model, and an optimization guided by an annealed loss function to avoid undesirable local minima.

The rest of this chapter is organized as follows. We first introduce a simple denoising CNN which embraces the progress in learning and designing CNN in Sect. 4.2. We show that residual learning and batch normalization are particularly beneficial to Gaussian denoising. We also analyze the rationale of residual learning and the modeling capacity of CNN. In Sect. 4.3, we provide a fast and flexible denoising CNN with a tunable noise level map as input and thus can handle a wide range of noise levels as well as spatially variant AWGN via a single model. To demonstrate the wide applications of CNN denoisers, in Sect. 4.4, we show that the CNN denoisers can be plugged into model-based optimization methods as a modular part to solve other image restoration tasks such as image deblurring, singe image super-resolution (SISR), and image inpainting. We provide a short review of recent advances, and discuss the challenges and some possible solutions in Sect. 4.5.

4.2 Learning Deep CNN for Image Denoising

Discriminative model learning for image denoising has been recently attracting considerable attention due to its favorable denoising performance. In this section, we take one step forward by investigating the construction of feed-forward denoising convolutional neural networks (DnCNN) to embrace the progress in deep architecture, learning algorithm, and regularization method into image denoising.

4.2.1 Architecture Design: DnCNN

The architecture of DnCNN is shown in Fig. 4.1. The input of DnCNN is a noisy observation $\mathbf{y} = \mathbf{x} + \mathbf{v}$. Discriminative denoising models such as MLP [5] and CSF [38] aim to directly learn the original mapping function $\mathcal{F}(\mathbf{y}) = \mathbf{x}$ to predict the latent clean image. For DnCNN, the residual learning formulation is instead to train a residual mapping $\mathcal{R}(\mathbf{y}) = \mathbf{v}$, and then $\mathbf{x}$ can be obtained by $\mathbf{x} = \mathbf{y} - \mathcal{R}(\mathbf{y})$.

Following the principle in [40], the size of convolutional filters is set to 3×3. Therefore, the receptive field of DnCNN with depth of d should be $(2d + 1) \times (2d + 1)$. Increasing the network depth can make use of the context information in larger image region at the cost of computational burden. For better trade-off between performance and efficiency, one important issue in architecture design is to set a proper depth for DnCNN. The receptive field size of DnCNN for Gaussian denoising with a certain noise level is set to 35×35 with the corresponding depth of 17.

Given the DnCNN with depth D, there are three types of layers (shown in Fig. 4.1) with three different colors. (i) Conv + ReLU: for the first layer, 64 filters of size $3 \times 3 \times c$ are used to generate 64 feature maps, and rectified linear units (ReLU, $\max(0, \cdot)$) are then utilized for nonlinearity. Here, c represents the number of image channels, i.e., $c = 1$ for grayscale image and $c = 3$ for color image. (ii) Conv + BN + ReLU: for layers $2 \sim (D - 1)$, 64 filters of size $3 \times 3 \times 64$ are used, and batch normalization [17] is added between convolution and ReLU. (iii) Conv: for the last layer, c filters of size $3 \times 3 \times 64$ are used to reconstruct the denoising result.

To sum up, DnCNN has two main features: the residual learning formulation is adopted to learn $\mathcal{R}(\mathbf{y})$, and batch normalization is incorporated to speed up training as well as boost the denoising performance. In particular, it turns out that residual

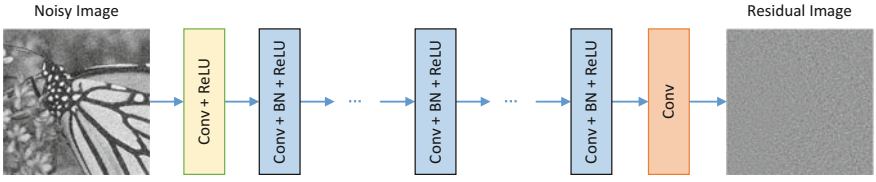


Fig. 4.1 The architecture of the DnCNN network

learning and batch normalization can benefit from each other, and their integration is effective in speeding up the training and boosting the denoising performance.

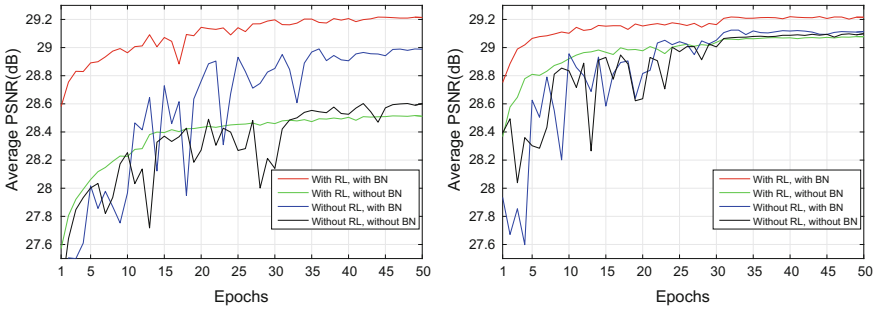
4.2.2 Residual Learning and Batch Normalization

To start with, it is useful to briefly review residual learning and batch normalization. The main idea of residual learning [15] is that the residual mapping is much easier to be learned than the original mapping. Typically, residual network stacks a number of residual units to alleviate the degradation of training accuracy. Benefited from residual network, deep CNN can be easily trained and improved accuracy has been achieved for image classification and object detection [15]. Different from the residual network [15] that uses many residual units (i.e., identity shortcuts), DnCNN employs a single residual unit to predict the residual image. It should be noted that, prior to the residual network [15], the strategy of predicting the residual image has already been adopted in some low-level vision problems such as SISR and image demosaicking. As for batch normalization [17], it was originally proposed to alleviate the internal covariate shift by incorporating a Gaussian normalization step and a scale and shift step. It enjoys several merits, such as fast training, better performance, and low sensitivity to initialization. For further details on batch normalization, please refer to [17].

The DnCNN network can be used to train either the original mapping $\mathbf{y}$ to predict $\mathbf{x}$ or the residual mapping $\mathcal{R}(\mathbf{y})$ to predict $\mathbf{v}$. According to [15], when the original mapping is more like an identity mapping, the residual mapping will be much easier to optimize. Since the noisy observation $\mathbf{y}$ is much more like the latent clean image $\mathbf{x}$ than the residual image $\mathbf{v}$ (especially when the noise level is low), $\mathcal{F}(\mathbf{y})$ would be more close to an identity mapping than $\mathcal{R}(\mathbf{y})$. Thus, the residual learning formulation is more suitable for image denoising.

For Gaussian noise removal, residual learning is also helpful to stabilize the training with batch normalization. Under the residual learning setting, the noise output of a specific noise level should be an ideal Gaussian distribution. Moreover, DnCNN with residual learning implicitly removes the latent clean image with the operations in the hidden layers. This makes that the inputs of each layer are Gaussian-like distributed, less correlated, and less related with image content. As a result, residual learning and batch normalization are particularly beneficial to each other for Gaussian denoising.

Figure 4.2 shows the average PSNR values obtained using these two learning formulations with/without batch normalization under the same setting on gradient-based optimization algorithms and network architecture. Two gradient-based optimization algorithms are adopted: one is the stochastic gradient descent algorithm with momentum (i.e., SGD) and the other one is the ADAM algorithm [20]. One can observe that both the SGD and ADAM optimization algorithms can enable the network with residual learning and batch normalization to have the best results. In other words,



(a) SGD (b) ADAM

Fig. 4.2 The quantitative Gaussian denoising results of four specific models under two gradient-based optimization algorithms, i.e., **a** SGD, **b** ADAM, with respect to epochs. The four specific models are in different combinations of residual learning (RL) and batch normalization (BN) and are trained with noise level 25. The results are evaluated on 68 natural images from Berkeley segmentation dataset

it is the integration of residual learning formulation and batch normalization rather than the optimization algorithms (SGD or ADAM) that lead to the best denoising performance.

4.2.3 Connection with TNRD

To have a further understanding of residual learning for denoising, we analyze its connection with TNRD [8] which is a MAP-CNN based denoising method. The main idea of TNRD is to train a discriminative solution for the following problem:

$$\min_{\mathbf{x}} \Psi(\mathbf{y}-\mathbf{x}) + \lambda \sum_{k=1}^{K} \sum_{p=1}^{N} \rho_k((\mathbf{f}_k * \mathbf{x})_p) \tag{4.5}$$

from an abundant set of degraded-clean training image pairs. Here, N denotes the image size, λ is the regularization parameter, $\mathbf{f}_k * \mathbf{x}$ stands for the convolution of the image $\mathbf{x}$ with the k-th filter kernel $\mathbf{f}_k$, and $\rho_k(\cdot)$ represents the k-th penalty function which is adjustable in the TNRD model. For Gaussian denoising, we set $\Psi(\mathbf{z}) = \frac{1}{2}\|\mathbf{z}\|^2$.

The diffusion iteration of the first stage can be interpreted as performing one gradient descent inference step at starting point $\mathbf{y}$, which is given by

$$\mathbf{x}_1 = \mathbf{y} - \alpha\lambda \sum_{k=1}^{K} (\bar{\mathbf{f}}_k * \phi_k(\mathbf{f}_k * \mathbf{y})) - \alpha \left.\frac{\partial \Psi(\mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{z}=\mathbf{0}} \tag{4.6}$$

where $\bar{\mathbf{f}}_k$ is the adjoint filter of $\mathbf{f}_k$ (i.e., $\bar{\mathbf{f}}_k$ is obtained by rotating 180° the filter $\mathbf{f}_k$), α corresponds to the stepsize and $\rho_k'(\cdot) = \phi_k(\cdot)$. For Gaussian denoising, we have $\frac{\partial \Psi(\mathbf{z})}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\mathbf{0}} = \mathbf{0}$, and Eq. (4.6) is equivalent to the following expression,

$$\mathbf{v}_1 = \mathbf{y} - \mathbf{x}_1 = \alpha\lambda \sum_{k=1}^{K} (\bar{\mathbf{f}}_k * \phi_k(\mathbf{f}_k * \mathbf{y})) \tag{4.7}$$

where $\mathbf{v}_1$ is the estimated residual of $\mathbf{x}$ with respect to $\mathbf{y}$.

Since the influence function $\phi_k(\cdot)$ can be regarded as pointwise nonlinearity applied to convolution feature maps, Eq. (4.7) actually is a two-layer feed-forward CNN. DnCNN further generalizes one-stage TNRD from three aspects: (i) replacing the influence function with ReLU to ease CNN training; (ii) increasing the CNN depth to improve the capacity in modeling image characteristics; (iii) incorporating with batch normalization to boost the performance. The connection with one-stage TNRD provides insights in explaining the use of residual learning for CNN-based image restoration.

4.2.4 Understanding the CNN Modeling Capacity

The existing Gaussian denoising methods, such as MLP [5] and TNRD [8], all train a specific model for a fixed noise level. It is interesting to investigate the modeling capacity of CNN for different noise levels via a single model. According to Eq. (4.7), one can see that most of the parameters are derived from the analysis prior term of Eq. (4.5). In this sense, the parameters are mainly representing the image priors which are task-independent. Therefore, CNN has the modeling capacity to deal with multiple degradations via a single model. For example, even the noise is not Gaussian distributed, one still can utilize Eq. (4.6) to obtain $\mathbf{v}_1$ if

$$\frac{\partial \Psi(\mathbf{z})}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\mathbf{0}} = \mathbf{0}. \tag{4.8}$$

Note that Eq. (4.8) holds for many types of noise distributions, e.g., generalized Gaussian distribution. It is natural to assume that it also holds for the noise caused by image downsampling and JPEG compression. Thus, it is possible to train a single CNN model for several general image denoising tasks, e.g., SISR and JPEG deblocking.

To demonstrate the potential of DnCNN in general image denoising, DnCNN is extended for learning a single model for three specific tasks, i.e., blind Gaussian denoising for noise level range of [0, 55], SISR, and JPEG deblocking are considered. In the training stage, the images with AWGN from a wide range of noise levels, downsampled images with multiple upscaling factors, and JPEG images with

(a) Input Image (b) Output Residual Image (c) Restored Image

Fig. 4.3 An example to show the capacity of DnCNN for three different tasks. The input image is composed by noisy images with noise level 15 (upper left) and 25 (lower left), bicubically interpolated low-resolution images with upscaling factor 2 (upper middle) and 3 (lower middle), JPEG images with quality factor 10 (upper right) and 30 (lower right). Note that the white lines in the input image are just used for distinguishing the six regions, and the residual image is normalized into the range of [0, 1] for visualization

different quality factors are utilized. Experimental results show that the learned single DnCNN model is able to yield excellent results for any of the three general image denoising tasks. Figure 4.3 shows an example of DnCNN for these tasks. As one can see, even the input image is corrupted with different distortions in different regions, the restored image looks natural and does not have obvious artifacts.

4.2.5 Implementation and Experiments

Following [8], 400 images of size 180×180 are used for training the Gaussian denoiser with known noise level. It has been found that using a larger training dataset can only bring negligible improvements, especially on BSD68 test set. Three noise levels, i.e., $\sigma = 15, 25$ and 50, are considered and thus three models are trained. The patch size is set as 40×40. $128 \times 1{,}600$ patches are used to train the model and the mini-batch size is set to 128. The mean squared error between the desired residual images and estimated ones from noisy input

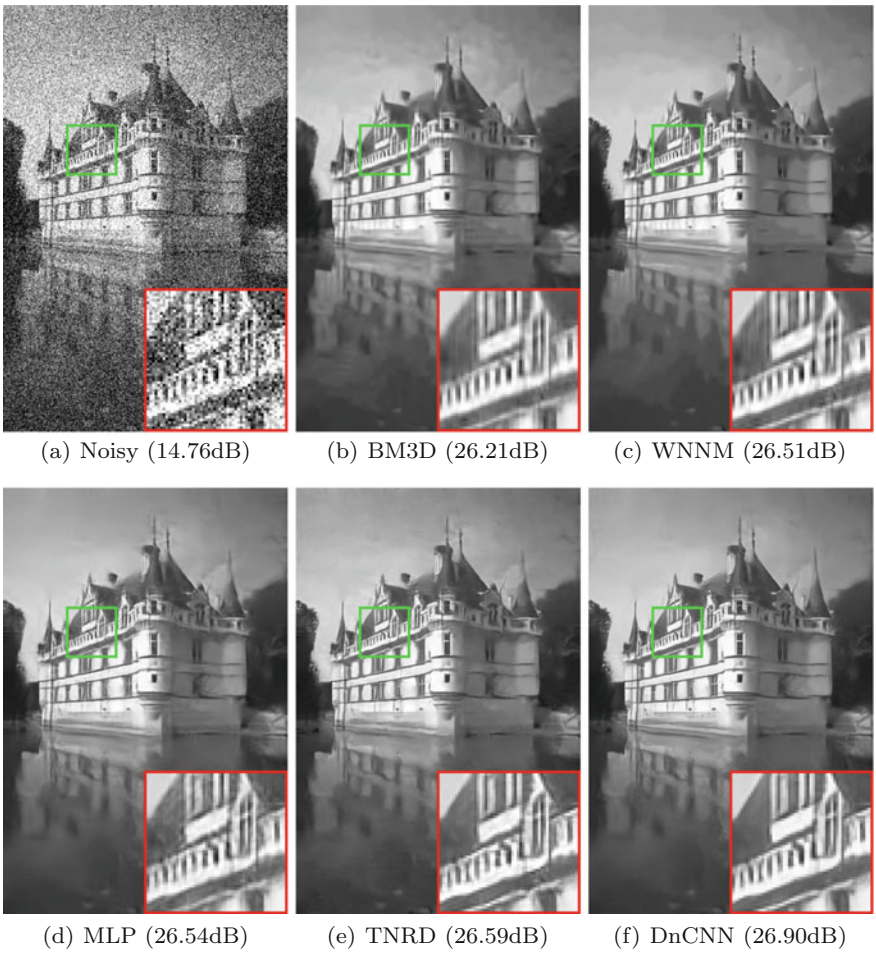
$$\mathcal{L}(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \|\mathcal{R}(\mathbf{y}_i; \Theta) - (\mathbf{y}_i - \mathbf{x}_i)\|^2 \tag{4.9}$$

is adopted as the loss function to learn the trainable parameters Θ. Here $\{(\mathbf{y}_i, \mathbf{x}_i)\}_{i=1}^{N}$ represents N noisy-clean training image patch pairs. The learning rate was decayed exponentially from 10^{-1} to 10^{-4} for 50 epochs. It takes about 6 hours to train a DnCNN model.

To evaluate the model, a dataset containing 68 natural images from Berkeley segmentation dataset (BSD68) [35] is adopted. The average PSNR results of different methods are shown in Table 4.1. As one can see, compared to the benchmark BM3D, the methods MLP and TNRD have a PSNR gain of about 0.35 dB. Notably, DnCNN outperforms BM3D by 0.6 dB on all the three noise levels.

Table 4.1 The average PSNR (dB) results of different methods on the BSD68 dataset. The best results are highlighted in bold

Noise levels	BM3D [5]	WNNM [13]	MLP [5]	TNRD [8]	DnCNN
$\sigma = 15$	31.07	31.37	–	31.42	**31.73**
$\sigma = 25$	28.57	28.83	28.96	28.92	**29.23**
$\sigma = 50$	25.62	25.87	26.03	25.97	**26.23**



(a) Noisy (14.76dB) (b) BM3D (26.21dB) (c) WNNM (26.51dB)

(d) MLP (26.54dB) (e) TNRD (26.59dB) (f) DnCNN (26.90dB)

Fig. 4.4 Denoising results of different methods on noise level 50

Figure 4.4 illustrates the visual results of different methods. It can be seen that BM3D, WNNM and MLP tend to produce over-smooth edges and textures. While preserving sharp edges and fine details, TNRD is likely to generate artifacts in the smooth region. In contrast, DnCNN can not only recover sharp edges and fine details but also yield visually pleasant results in the smooth region.

4.3 CNN for Fast and Flexible Image Denoising

In order to handle practical image denoising problems, a flexible image denoiser is expected to have the following desirable properties: (i) it is able to perform denoising using a single model; (ii) it is efficient, effective, and user-friendly; and (iii) it can handle spatially variant noise. Such a denoiser can be directly deployed to recover the clean image when the noise level is known or can be well estimated. When the noise level is unknown or is difficult to estimate, the denoiser should allow the user to adaptively control the trade-off between noise reduction and details preservation. Furthermore, the noise can be spatially variant and the denoiser should be flexible enough to handle spatially variant noise.

The FFDNet which is shown in Fig. 4.5 is proposed to meet with such desirable properties. Specifically, one of the major differences of FFDNet is that it can be formulated as $\mathbf{x} = \mathcal{F}(\mathbf{y}, \mathbf{M}; \Theta)$, where $\mathbf{M}$ is a noise level map. Here, the noise level map $\mathbf{M}$ is modeled as an input and the model parameters Θ are invariant to noise level. Hence, FFDNet provides a flexible way to handle various types of noise with a single network. By introducing a noise level map as input, network design and training methods are required to be further studied for effective and efficient denoising. Another main difference of FFDNet is that it works on downsampled subimages, which largely accelerates the training and testing speed, and enlarges the receptive field as well.

4.3.1 Network Architecture: FFDNet

As shown in Fig. 4.5, the first layer of FFDNet is a reversible downsampling operator which reshapes an $H \times W$ noisy image $\mathbf{y}$ into four downsampled subimages. Then,

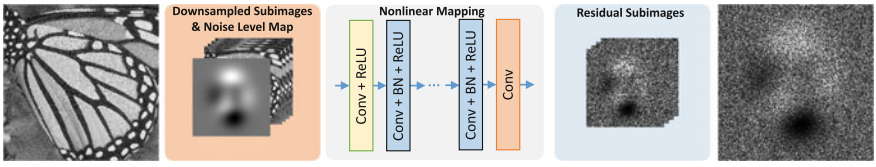


Fig. 4.5 The architecture of the FFDNet network

a tunable downsampled noise level map $\mathbf{M}$ with the downsampled subimages is concatenated to form a tensor $\tilde{\mathbf{y}}$ of size $\frac{H}{2} \times \frac{W}{2} \times (4c+1)$ as the inputs to CNN. The following CNN consists of a series of 3×3 convolutional layers. Each layer is composed of three types of operations: Convolution (Conv), Rectified Linear Units (ReLU) [22], and Batch Normalization (BN) [17]. More specifically, "Conv+ReLU" is adopted for the first convolutional layer, "Conv+BN+ReLU" for the middle layers, and "Conv" for the last convolutional layer. Zero-padding is employed to keep the size of feature maps unchanged after each convolution. After the last convolutional layer, an upscaling operation is applied as the reverse operator of the downsampling operator applied in the input stage to produce the residual noisy image $\tilde{\mathbf{v}}$ of size $H \times W \times c$. The denoised image is then obtained by $\tilde{\mathbf{x}} = \mathbf{y} - \tilde{\mathbf{v}}$. Since FFDNet operates on downsampled subimages, it is not necessary to employ the dilated convolution [49] to further increase the receptive field.

By considering the balance of complexity and performance, the number of convolutional layers are set to 15 for grayscale image and 12 for color image. As for the number of feature maps, they are set to 64 for grayscale image and 96 for color image. The reason of using different settings for grayscale and color images is twofold. First, since there are high dependencies among the R, G, B channels, using a smaller number convolutional layers is good enough to exploit the interchannel dependency for denoising. Second, color image has more channels as input, and hence more feature (i.e., a larger number of feature maps) is required.

4.3.2 Taking Noise Level as CNN Input

Most of the deep learning based denoising methods such as the MLP [5] and convolutional neural network (CNN) based methods [18, 50] are limited in flexibility, and the learned model is usually tailored to a specific noise level. From the perspective of regression, some CNN-based denoisers such as DnCNN aim to learn a mapping function $\mathbf{x} = \mathcal{F}(\mathbf{y}; \Theta_\sigma)$ between the input noisy observation $\mathbf{y}$ and the desired output $\mathbf{x}$. The model parameters Θ_σ are trained for noisy images corrupted by AWGN with a fixed noise level σ, while the trained model with Θ_σ is hard to be directly deployed to images with other noise levels.

From the viewpoint of MAP framework, the solution of Eq. (4.2) actually defines an implicit function $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \sigma, \lambda; \Theta)$ of the noisy image $\mathbf{y}$, noise level σ, and parameter λ. Since λ can be absorbed into σ, the solution $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \sigma, \lambda; \Theta)$ can be rewritten as $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \sigma; \Theta)$. In this sense, setting noise level σ also plays the role of setting λ to control the trade-off between noise reduction and detail preservation.

Since the inputs $\mathbf{y}$ and σ have different dimensions, it is not easy to directly feed them into CNN. To resolve this, a simple dimension stretching strategy can be employed to stretch the noise level σ into a noise level map $\mathbf{M}$. In $\mathbf{M}$, all the elements are σ. Then, the formulation is changed into $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{y}, \mathbf{M}; \Theta)$. For color image, $\mathbf{M}$ can have multiple channels to represent the noise level map of R, G, B channels and can also be extended to degradation map which parameterizes the degradation

process [52]. As such, a trained CNN model is expected to inherit the flexibility of handling images with different noise levels, even spatially variant noises via a single model.

4.3.3 Single Non-blind Model Versus Single Blind Model

So far, we have know that it is possible to learn a single model for blind and non-blind Gaussian denoising, respectively, it is necessary to point out their differences. First, the generalization ability is different. Although the blind model performs favorably for synthetic AWGN removal without knowing the noise level, it does not generalize well to real noisy images. In contrast, the non-blind model with noise level map has the ability to control the trade-off between noise removal and detail preservation, thus it can deal with real noise to some extent. In addition, the non-blind model can handle the out-of-range noise levels, whereas the blind one lacks such an ability. Second, the performance for AWGN removal is different. The non-blind model with noise level map has better performance for AWGN removal than the blind one. This phenomenon has also been recognized in the task of SISR [32]. Third, the application range is different. The non-blind model can be plugged into model-based optimization methods to solve various image restoration tasks, such as image deblurring, SISR and image inpainting. However, the blind model does not have this merit.

4.3.4 Denoising on Subimages

Efficiency is another crucial issue for practical CNN-based denoising. One straightforward idea is to reduce the depth and number of filters. However, such a strategy will sacrifice much the modeling capacity and receptive field of CNN [50]. Shi et al. [39] proposed to extract deep features directly from the low-resolution image for super-resolution, and introduced a sub-pixel convolution layer to improve computational efficiency. In the application of image denoising, we introduce a reversible downsampling layer to reshape the input image into a set of small subimages. Here, the downsampling factor is set to 2 since it can largely improve the speed by slightly reducing modeling capacity. The CNN is deployed on the subimages, and finally a sub-pixel convolution layer is adopted to reverse the downsampling process.

Denoising on downsampled subimages can also effectively expand the receptive field which in turn leads to a moderate network depth. For example, the proposed network with a depth of 15 and 3×3 convolution will have a large receptive field of 62×62. In contrast, a plain 15-layer CNN only has a receptive field size of 31×31. We note that the receptive field of most state-of-the-art denoising methods ranges from 35×35 to 61×61 [50]. Further increase of receptive field actually benefits

little in improving denoising performance [27]. What is more, the introduction of subsampling and sub-pixel convolution is also effective in reducing the memory burden.

4.3.5 Dataset Generation and Network Training

To train the FFDNet model, we need to prepare a training dataset of patches $(\mathbf{y}, \mathbf{M}; \mathbf{x})$. Here, $\mathbf{y}$ is obtained by adding AWGN to latent image $\mathbf{x}$, and $\mathbf{M}$ is the corresponding noise level map. The reasons of using AWGN to generate the training dataset are as follows. First, AWGN is a natural choice when there is no specific prior information on noise source. Second, real-world noise can be locally approximated as AWGN [24]. It is worth noting that FFDNet model is trained on the noisy images $\mathbf{y} = \mathbf{x} + \mathbf{v}$ without quantization to 8-bit integer values. Though the real noisy images are generally 8-bit quantized, we empirically found that the learned model still works effectively on real noisy images. For the noise level of each noisy, it is uniformly sampled from the range of [0, 75].

The ADAM algorithm [20] is adopted to optimize FFDNet by minimizing the mean squared error loss. The learning rate starts from 10^{-3} and reduces to 10^{-4} when the training error stops decreasing. When the training error keeps unchanged in five sequential epochs, we merge the parameters of each batch normalization into the adjacent convolution filters. Then, a smaller learning rate of 10^{-6} is adopted for additional 20 epochs to fine-tune the FFDNet model. As for the other hyper-parameters of ADAM, we use their default settings. The mini-batch size is set as 128, and the rotation and flip based data augmentation is also adopted during training. The FFDNet models are trained in Matlab (R2015b) environment with MatConvNet package [44] and an Nvidia Titan X Pascal GPU. The training of a single model can be done in about one day.

4.3.6 Experiments on Synthetic and Real Images

4.3.6.1 Experiments on AWGN Removal

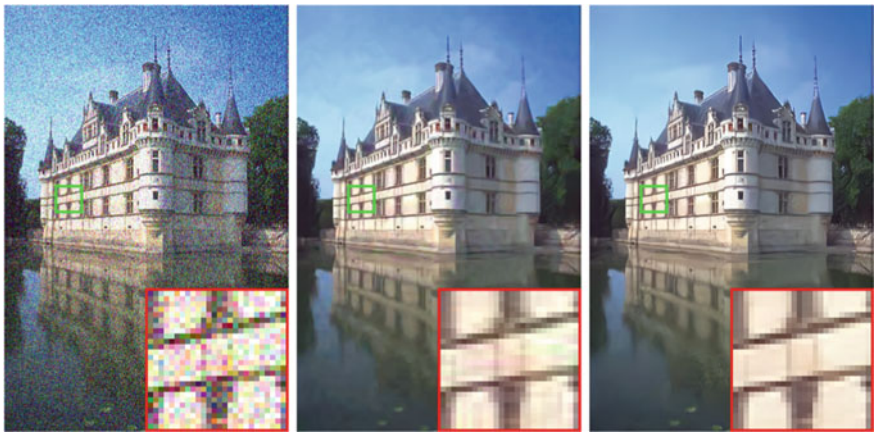
The PSNR results of different methods on BSD68 dataset are reported in Table 4.2, from which we have the following observations. First, FFDNet surpasses BM3D by a large margin and outperforms WNNM, MLP, and TNRD by about 0.2 dB for a wide range of noise levels. Second, FFDNet is slightly inferior to DnCNN when the noise level is low (e.g., $\sigma \leq 25$), but gradually outperforms DnCNN with the increase of noise level (e.g., $\sigma > 25$). This phenomenon may result from the trade-off between receptive field size and modeling capacity. FFDNet has a larger receptive field than DnCNN, thus favoring for removing strong noise, while DnCNN has better modeling capacity which is beneficial for denoising images with lower noise level.

Table 4.2 The average PSNR (dB) results of different methods on BSD68 with noise levels 15, 25 35, 50, and 75. The best results are highlighted in bold

Methods	BM3D [11]	WNNM [13]	MLP [5]	TNRD [8]	DnCNN [50]	FFDNet
$\sigma = 15$	31.07	31.37	–	31.42	**31.72**	31.62
$\sigma = 25$	28.57	28.83	28.96	28.92	**29.23**	29.19
$\sigma = 35$	27.08	27.30	27.50	–	27.69	**27.73**
$\sigma = 50$	25.62	25.87	26.03	25.97	26.23	**26.30**
$\sigma = 75$	24.21	24.40	24.59	–	24.64	**24.78**

Table 4.3 The average PSNR (dB) results of CBM3D and FFDNet on CBSD68 dataset with noise levels 15, 25 35, 50, and 75

Methods	$\sigma = 15$	$\sigma = 25$	$\sigma = 35$	$\sigma = 50$	$\sigma = 75$
CBM3D	33.52	30.71	28.89	27.38	25.74
FFDNet	33.80	31.18	29.57	27.96	26.24



(a) Noisy ($\sigma = 35$) (b) CBM3D (29.90dB) (c) FFDNet (30.51dB)

Fig. 4.6 Color image denoising results by CBM3D and FFDNet

Table 4.3 reports the performance of CBM3D and FFDNet on color version of BSD68 datasets, and Fig. 4.6 presents the visual comparisons. It can be seen that FFDNet consistently outperforms CBM3D on different noise levels in terms of both quantitative and qualitative evaluation.

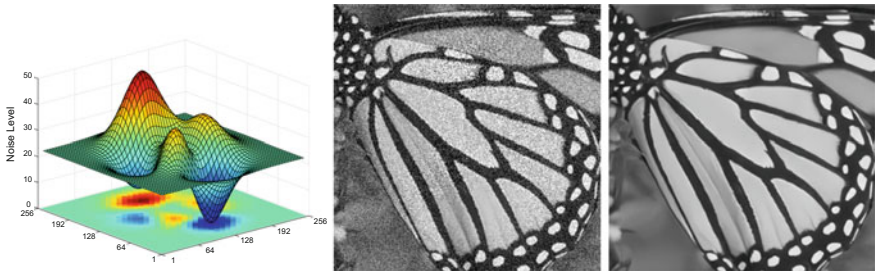


Fig. 4.7 Examples of FFDNet on removing spatially variant AWGN. Left: noise level maps. Middle: noisy images with PSNR 20.55 dB. Right: denoised images (PSNR: 29.97 dB) by FFDNet

4.3.6.2 Experiments on Spatially Variant AWGN Removal

We then test the flexibility of FFDNet to deal with spatially variant AWGN. To synthesize spatially variant AWGN, we first generate an AWGN image $\mathbf{v}$ with the standard deviation 1 and a noise level map $\mathbf{M}$ of the same size. Then, elementwise multiplication is applied on $\mathbf{v}$ and $\mathbf{M}$ to produce the spatially variant AWGN, i.e., $\mathbf{v} \odot \mathbf{M}$. In the denoising stage, we take the bilinearly downsampled noise level map as the input to FFDNet. Figure 4.7 shows the denoising result of FFDNet for a spatially variant AWGN. One can see that FFDNet is flexible and powerful to remove spatially variant AWGN.

4.3.6.3 Experiments on Noise Level Sensitivity

In practical applications, the noise level map may not be accurately estimated from the noisy observation, and mismatch between the input and real noise levels is inevitable. If the input noise level is lower than the real noise level, the noise cannot be completely removed. Therefore, users often prefer to set a higher noise level to guarantee the removal of noise. However, this may also remove too much image details together with noise. In this subsection, we evaluate FFDNet in comparison with benchmark BM3D by varying different input noise levels for a given ground-truth noise level.

Figure 4.8 shows the visual comparisons between BM3D/CBM3D and FFDNet by setting different input noise levels to denoise a noisy image. Four typical image structures, including flat region, sharp edge, line with high contrast, and line with low contrast, are selected for visual comparison to investigate the noise level sensitivity of BM3D and FFDNet. The following observations can be obtained. The best visual quality is obtained when the input noise level matches the ground-truth one. BM3D and FFDNet produce similar visual results with lower input noise levels, while they exhibit certain difference with higher input noise levels. Both of them will smooth out noise in flat regions, and gradually smooth out image structures with the increase in input noise levels. Particularly, FFDNet may wipe out some low contrast line structure, whereas BM3D can still preserve the mean patch regardless of the input

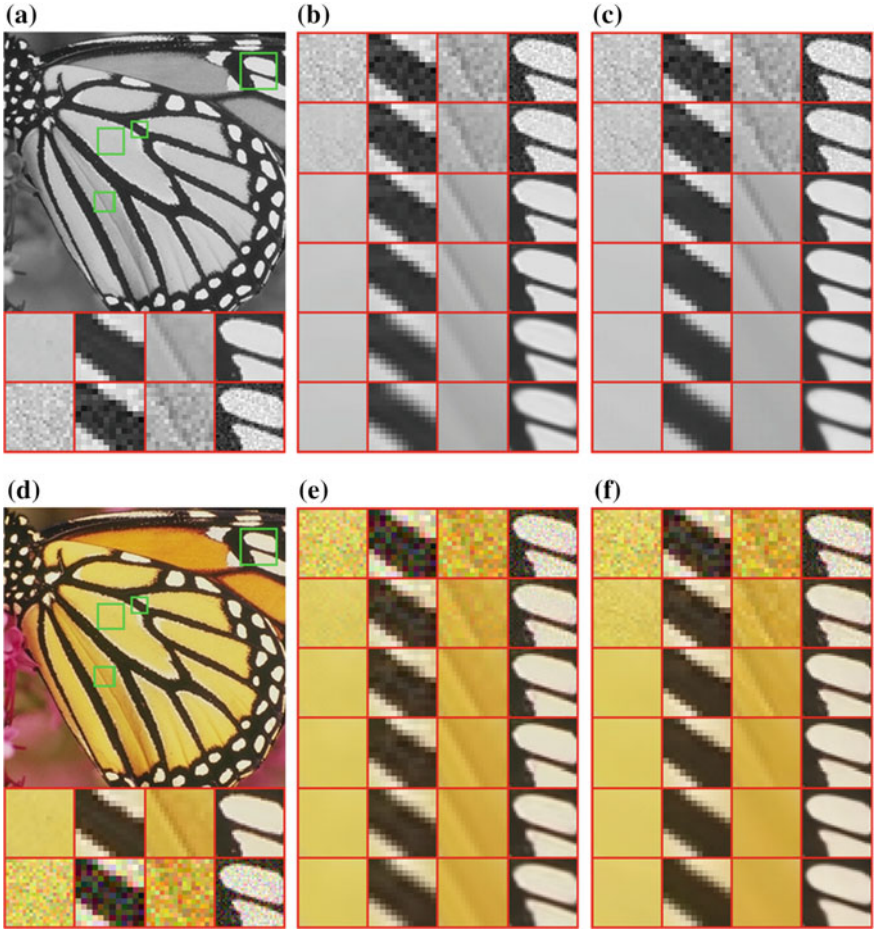


Fig. 4.8 Visual comparisons between FFDNet and BM3D/CBM3D by setting different input noise levels to denoise a noisy image. **a** From top to bottom: ground-truth image, four clean zoom-in regions, and the corresponding noisy regions (AWGN, noise level 15). **b** From top to bottom: denoising results by BM3D with input noise levels 5, 10, 15, 20, 50, and 75, respectively. **c** Results by FFDNet with the same settings as in (**b**). **d** From top to bottom: ground-truth image, four clean zoom-in regions, and the corresponding noisy regions (AWGN, noise level 25). **e** From top to bottom: denoising results by CBM3D with input noise levels 10, 20, 25, 30, 45, and 60, respectively. **f** Results by FFDNet with the same settings as in (**e**)

noise levels due to its use of nonlocal information. Using a higher input noise level can generally produce better visual results than using a lower one. In addition, there is no much visual difference when the input noise level is a little higher than the ground-truth one.

4.3.6.4 Experiments on Real Noisy Images

In this subsection, real noisy images are used to further assess the practicability of FFDNet. However, such an evaluation is difficult to conduct due to the following reasons. (i) Both the ground-truth clean image and noise level are unknown for real noisy image. (ii) The real noise comes from various sources such as camera imaging pipeline (e.g., shot noise, amplifier noise and quantization noise), scanning, lossy compression and image resizing [10, 28], and it is generally non-Gaussian, spatially variant, and signal-dependent. As a result, the AWGN assumption in many denoising algorithms does not hold, and the associated noise level estimation methods may not work well for real noisy images.

Instead of adopting any noise level estimation methods, we adopt an interactive strategy to handle real noisy images. First of all, we empirically found that the assumption of spatially invariant noise usually works well for most real noisy images. We then employ a set of typical input noise levels to produce multiple outputs, and select the one which has best trade-off between noise reduction and details preservation. Second, for spatially variant noise, we sample several typical image patches which represent the distinct regions of different noise levels and apply different input noise levels to them. By observing the denoising results, we then choose the proper noise level for each typical patch. The noise levels at other locations are interpolated from the noise levels of the typical patches. An approximation of nonuniform noise level map can then be obtained. In our following experiments, unless otherwise specified, we assume spatially invariant noise for the real noisy images.

Since there is no ground-truth image for a real noisy image, visual comparison is employed to evaluate the performance of FFDNet. We choose BM3D for comparison because it is widely accepted as a benchmark for denoising applications. Given a noisy image, the same input noise level is used for BM3D and FFDNet.

Figure 4.9 compares the grayscale image denoising results on four noisy images. As one can see, BM3D and FFDNet exhibit similar behaviors to those on denoising grayscale images. In particular, for image "*Building*" which contains some structured noises, BM3D fails to yield visually pleasant results because the structured noises fit the nonlocal self-similarity prior. In contrast, FFDNet removes such noise without losing underlying image textures. Figure 4.10 shows the denoising results of CBM3D and FFDNet on four color noisy images. It can be seen that FFDNet can handle various kinds of noises, including Gaussian-like noise (see image "*Pattern*"), JPEG lossy compression noise (see image "*Audrey Hepburn*"), and low-frequency noise (see image "*Boy*"). Similarly, from the denoising results of "*Boy*", one can see that CBM3D remains the structured low-frequency noise unremoved whereas FFDNet removes successfully such kind of noise. As a result, we can conclude that while the nonlocal self-similarity prior helps to remove random noise, it hinders the removal of structured noise. In comparison, the prior implicitly learned by CNN is able to remove both random noise and structured noise.

Figure 4.11 shows a more challenging example to demonstrate the advantage of FFDNet for denoising noisy images with spatially variant noise. As one can see, while FFDNet with a small input noise level can recover the details of regions with

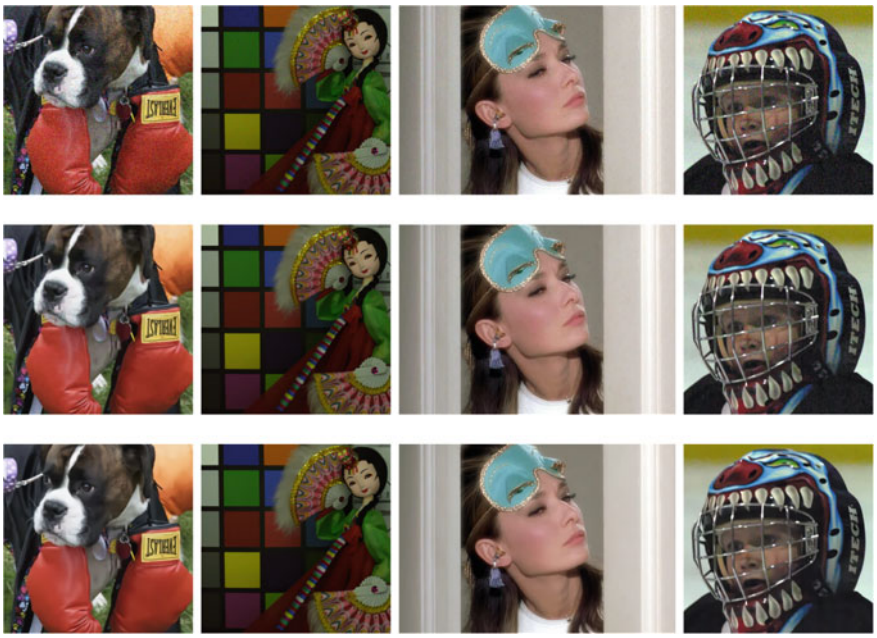
(a) *David Hilbert* (b) *Chupa Chups* (c) *Vinegar* (d) *Building*

Fig. 4.9 Grayscale image denoising results by different methods on real noisy images. From top to bottom: noisy images, denoised images by BM3D, denoised images by FFDNet. **a** $\sigma = 15$; **b** $\sigma = 10$; **c** $\sigma = 20$; **d** $\sigma = 20$

low noise level, it fails to remove strong noise. On the other hand, FFDNet with a large input noise level can remove strong noise but it will also smooth out the details in the region with low noise level. In comparison, the denoising result with a proper nonuniform noise level map not only preserves image details but also removes the strong noise.

4.3.7 Running Time

Table 4.4 lists the running time results of BM3D, DnCNN and FFDNet for denoising grayscale level and color images with size 256×256, 512×512, and $1{,}024 \times 1{,}024$. The evaluation was performed in Matlab (R2015b) environment on a computer with a six-core Intel(R) Core(TM) i7-5820K CPU @ 3.3 GHz, 32 GB of RAM and a



(a) *Dog* (b) *Pattern* (c) *Audrey Hepburn* (d) *Boy*

Fig. 4.10 Color image denoising results by different methods on real noisy images. From top to bottom: noisy images, denoised images by CBM3D, denoised images by FFDNet. **a** $\sigma = 28$; **b** $\sigma = 12$; **c** $\sigma = 10$; **d** $\sigma = 45$

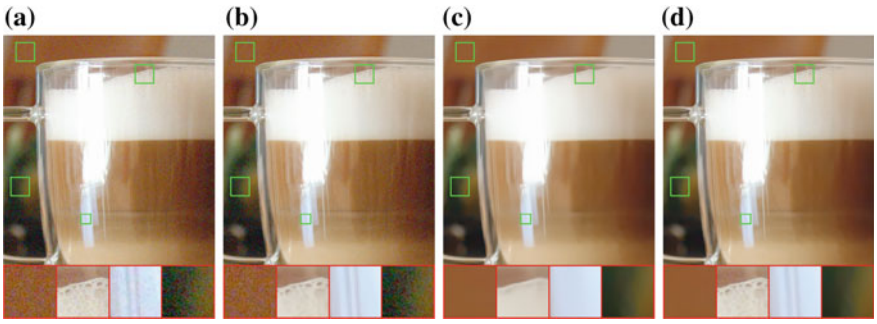


Fig. 4.11 An example of FFDNet on image "*Glass*" with spatially variant noise. **a** Noisy image; **b** denoised image by FFDNet with $\sigma = 10$; **c** denoised image by FFDNet with $\sigma = 35$; **d** denoised image by FFDNet with nonuniform noise level map

Table 4.4 Running time (in seconds) of different methods for denoising images with size 256 × 256, 512 × 512, and 1,024 × 1,024

Methods	Device	256 × 256		512 × 512		1024 × 1024	
		Gray	Color	Gray	Color	Gray	Color
BM3D	CPU(ST)	0.59	0.98	2.52	3.57	10.77	20.15
DnCNN	CPU(ST)	2.14	2.44	8.63	9.85	32.82	38.11
	CPU(MT)	0.74	0.98	3.41	4.10	12.10	15.48
	GPU	0.011	0.014	0.033	0.040	0.124	0.167
FFDNet	CPU(ST)	0.44	0.52	1.81	2.14	7.24	8.51
	CPU(MT)	0.18	0.19	0.73	0.79	2.96	3.15
	GPU	0.006	0.007	0.012	0.016	0.038	0.054

Nvidia Titan X Pascal GPU. For BM3D, we evaluate its running time by denoising images with noise level 25. For DnCNN, the grayscale and color image denoising models have 17 and 20 convolutional layers, respectively. The Nvidia cuDNN-v5.1 deep learning library is used to accelerate the computation of DnCNN and FFDNet. The memory transfer time between CPU and GPU is also counted. Note that DnCNN and FFDNet can be implemented with both single-threaded (ST) and multi-threaded (MT) CPU computations.

From Table 4.4, we have the following observations. First, BM3D spends much more time on denoising color images than grayscale images. The reason is that, compared to gray-BM3D, CBM3D needs extra time to denoise the chrominance components after luminance-chrominance color transformation. Second, while DnCNN can benefit from GPU computation for fast implementation, it has comparable CPU time to BM3D. Third, FFDNet spends almost the same time for processing grayscale and color images. More specifically, FFDNet with multi-threaded implementation is about three times faster than DnCNN and BM3D on CPU, and much faster than DnCNN on GPU. Even with single-threaded implementation, FFDNet is also faster than BM3D. Taking denoising performance and flexibility into consideration, FFDNet is very competitive for practical applications.

4.4 CNN Denoiser Prior Based Image Restoration

Motivated by the impressive achievement on image denoising, it is natural to ask whether CNNs can be applied to more general image restoration tasks. Although CNNs can be directly adopted with promising performance and fast testing speed, their application range is greatly restricted by the specialized task. In contrast, model-based optimization methods are flexible for handling different inverse problems but are usually time-consuming with sophisticated priors for the purpose of good performance. Fortunately, it has been revealed that, with the aid of variable splitting

techniques such as alternating direction method of multipliers (ADMM) algorithm, half quadratic splitting (HQS) algorithm, and the primal-dual algorithm [6], denoiser prior can be plugged in as a modular part of model-based optimization methods to solve other image restoration problems, and particularly, the regularization term only corresponds to a denoising subproblem [7, 16, 33, 41, 53]. Consequently, such an integration induces considerable advantage when the denoiser is based on CNN.

Very recently, various methods have been proposed to incorporate the CNN denoiser prior into model-based optimization methods. In those methods, the CNN denoiser can be either pretrained or jointly trained with data fidelity term for a specific task. In other words, there exist two general CNN denoiser prior based frameworks, i.e., model-based optimization and discriminative learning, for different image restoration tasks. In this section, we focus on the former since once the CNN denoiser is trained, no additional training is needed for other tasks. As for the variable splitting algorithm, we choose half quadratic splitting (HQS) algorithm due to its simplicity.

In the following, we first give a brief review of HQS algorithm and then show how to plug CNN denoiser into the optimization procedure to solve other image restoration problems, including image deblurring, single image super-resolution, and image inpainting.

4.4.1 Half Quadratic Splitting Algorithm

In general, the purpose of image restoration is to recover the latent clean image $\mathbf{x}$ from its degraded observation $\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}$, where $\mathbf{H}$ is a degradation matrix, $\mathbf{v}$ is additive white Gaussian noise of standard deviation σ. By specifying different degradation matrices, one can correspondingly get different image restoration tasks. Three classical IR tasks would be image denoising when $\mathbf{H}$ is an identity matrix, image deblurring when $\mathbf{H}$ is a blurring operator, and image super-resolution when $\mathbf{H}$ is a composite operator of blurring and downsampling.

Due to the ill-posed nature of general image restoration problems, regularization needs to be imposed to constrain the solution. Mathematically, the latent clean image of a degraded image $\mathbf{y}$ can be estimated by solving the following MAP problem:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \lambda\Phi(\mathbf{x}) \tag{4.10}$$

where the solution minimizes an energy function composed of a data fidelity term $\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$, a regularization term $\Phi(\mathbf{x})$ and a trade-off parameter λ.

In HQS, by introducing an auxiliary variable $\mathbf{z}$, Eq. (4.10) can be reformulated as a constrained optimization problem which is given by

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \lambda\Phi(\mathbf{z}) \quad s.t. \quad \mathbf{z} = \mathbf{x} \tag{4.11}$$

Then, HQS tries to minimize the following cost function:

$$\mathcal{L}_{\mu}(\mathbf{x}, \mathbf{z}) = \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \lambda\Phi(\mathbf{z}) + \frac{\mu}{2}\|\mathbf{z} - \mathbf{x}\|^2 \tag{4.12}$$

where μ is a penalty parameter which varies iteratively in a non-descending order. Equation (4.12) can be solved via the following iterative scheme:

$$\begin{cases} x_{k+1} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \mu\sigma^2\|\mathbf{x} - \mathbf{z}_k\|^2 & \text{(4.13a)} \\ \mathbf{z}_{k+1} = \arg\min_{\mathbf{z}} \frac{\mu}{2}\|\mathbf{z} - \mathbf{x}_{k+1}\|^2 + \lambda\Phi(\mathbf{z}). & \text{(4.13b)} \end{cases}$$

As one can see, the data fidelity term and regularization term are decoupled into two individual subproblems. Specifically, the data fidelity term is associated with a quadratic regularized least-squares problem (Eq. (4.13a)) which has various fast solutions for different degradation matrices. A direct solution is given by

$$\mathbf{x}_{k+1} = (\mathbf{H}^T\mathbf{H} + \mu\sigma^2\mathbf{I})^{-1}(\mathbf{H}^T\mathbf{y} + \mu\sigma^2\mathbf{z}_k) \tag{4.14}$$

The regularization term is involved in Eq. (4.13b) which can be rewritten as

$$\mathbf{z}_{k+1} = \arg\min_{\mathbf{z}} \frac{1}{2(\sqrt{1/\mu})^2}\|\mathbf{x}_{k+1} - \mathbf{z}\|^2 + \lambda\Phi(\mathbf{z}) \tag{4.15}$$

Equation (4.15) corresponds to denoising the image $\mathbf{x}_{k+1}$ by a CNN-based Gaussian denoiser with noise level $\sqrt{1/\mu}$. As a consequence, any CNN-based Gaussian denoisers can be acted as a modular part to solve Eq. (4.10). To address this, Eq. (4.15) can be rewritten as

$$\mathbf{z}_{k+1} = \mathcal{F}(\mathbf{x}_{k+1}, \sqrt{1/\mu}) \tag{4.16}$$

We point out that the CNN denoiser from Eq. (4.15) should be designed for AWGN removal and the noisy image in the training should not be quantized to 8-bit integer values.

So far, we have obtained that the image prior $\Phi(\cdot)$ can be implicitly replaced by a denoiser prior. Such a promising property actually offers several advantages. First, it enables to use fast and effective CNN-based denoisers to solve a variety of inverse problems. Second, the explicit image prior $\Phi(\cdot)$ can be unknown in solving Eq. (4.10). Third, several complementary denoisers which exploit different image priors can be jointly utilized to solve one specific problem.

4.4.2 CNN Denoisers and Parameter Setting

For the architecture of the CNN denoiser, it consists of seven layers with three different blocks, i.e., "Dilated Convolution+ReLU" block in the first layer, five "Dilated Convolution+Batch Normalization+ReLU" blocks in the middle layers, and "Dilated Convolution" block in the last layer. The dilation factors of (3×3) dilated convolutions from first layer to the last layer are set to 1, 2, 3, 4, 3, 2, and 1, respectively. The number of feature maps in each middle layer is set to 64. We trained a set of denoisers on noise level range [0, 50] and divided it by a step size of 2 for each model, resulting in a set of 25 denoisers for each grayscale and color image prior modeling.

Once the denoisers are provided, the subsequent crucial issue would be parameter setting. There involve two parameters, λ and μ, to tune. For the setting of λ, since it is implicitly optimized in the CNN denoiser and can be absorbed into σ, one can instead tune σ to obtain the best results. In practice, this can be achieved by multiplying σ by a scalar around 1. For the setting of μ, it is better to set the noise level of denoiser in each iteration to implicitly determine μ. Note that the noise level of denoiser $\sqrt{1/\mu}$ should be set from large to small. In the following experiments, it is decayed exponentially from 49 to a value in [1, 15] for 30 iterations. Note that all the experimental results are reproducible, and the source code can be downloaded from https://github.com/cszn/IRCNN.

4.4.3 Image Deblurring

For image deblurring, by assuming the convolution is carried out with circular boundary conditions, the fast implementation of Eq. (4.13a) is given by

$$\mathbf{x}_{k+1} = \mathcal{F}^{-1}\left(\frac{\overline{\mathcal{F}(\mathbf{k})}\mathcal{F}(\mathbf{y}) + \mu\sigma^2\mathcal{F}(\mathbf{z}_k)}{\overline{\mathcal{F}(\mathbf{k})}\mathcal{F}(\mathbf{k}) + \mu\sigma^2}\right) \tag{4.17}$$

where the $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ denote the fast Fourier transform (FFT) and inverse FFT, $\overline{\mathcal{F}(\cdot)}$ denotes complex conjugate of $\mathcal{F}(\cdot)$ and $\mathbf{k}$ is a blurring kernel corresponding to the degradation matrix $\mathbf{H}$.

Figure 4.12 gives an example of IRCNN for image deblurring. It can be seen that IRCNN can yield visually pleasant result with sharp edges and fine details.

4.4.4 Single Image Super-Resolution

There exist several degradation settings for single image super-resolution (SISR), among which bicubic degradation (default setting of Matlab function *imresize*) and

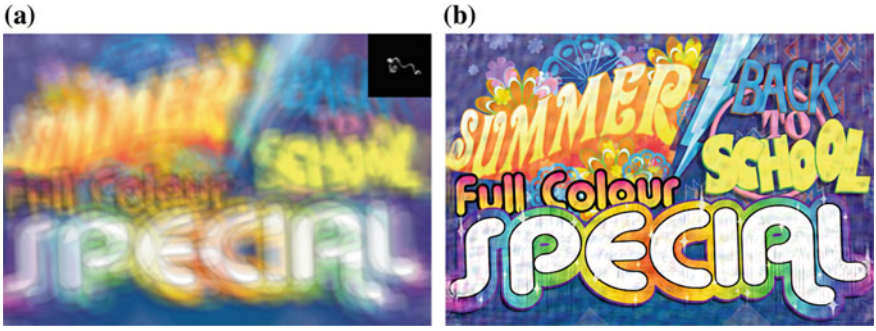


Fig. 4.12 An example of IRCNN for image deblurring. **a** Blurred image with estimated kernel by Pan et al. [30]; **b** Deblurring result

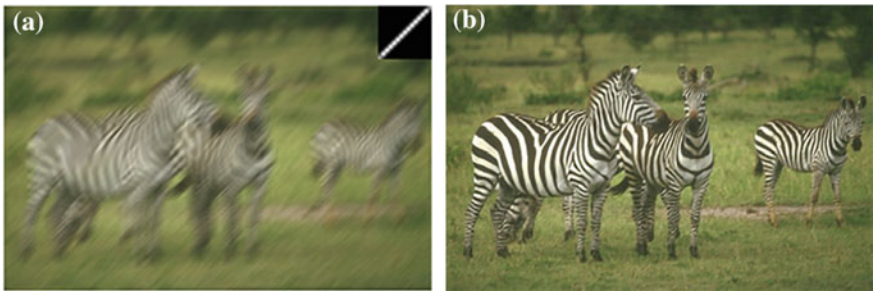


Fig. 4.13 An example of IRCNN for single image super-resolution (the blur kernel is a motion blur, the scale factor is 2). **a** LR image with motion blur kernel; **b** SISR result

Gaussian blurring followed by anti-aliasing downsampling are the two most widely used ones. For these two degradations, we use the following back-projection iteration to solve Eq. (4.13a),

$$\mathbf{x}_{k+1} = \mathbf{z}_k - \alpha(\mathbf{y} - \mathbf{z}_k \downarrow_{sf}) \uparrow^{sf}_{bicubic} \tag{4.18}$$

where $\downarrow_{sf}$ denotes the degradation operator with downscaling factor sf, $\uparrow^{sf}_{bicubic}$ represents bicubic interpolation operator with upscaling factor sf, and α is the step size which is fixed to 1.75.

It is worth noting that when the downsampler is the standard K-fold downsampler (Matlab function *downsample*), Eq. (4.13a) has a fast closed-form solution by benefiting FFT [7]. Furthermore, the blur kernel can go beyond Gaussian blur. Figure 4.13 shows an example of IRCNN for super-resolving LR image degraded by motion blurring and standard K-fold downsampler. It can be seen that the super-resolved image is much more visually pleasing than the LR image.

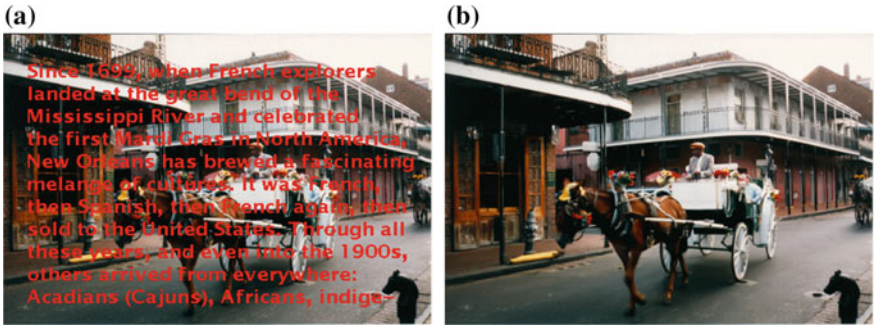


Fig. 4.14 An example of IRCNN for image inpainting. **a** Original image with overlaid text; **b** inpainting result

4.4.5 Image Inpainting

For image inpainting, $\mathbf{Hx}$ can be rewritten as $\mathbf{M} \odot \mathbf{x}$, where $\mathbf{M}$ is matrix with binary elements indicating the missing pixels of $\mathbf{y}$, and $\odot$ denotes elementwise multiplication. A closed-form solution of Eq. (4.13a) is given by

$$\mathbf{x}_{k+1} = (\mathbf{M} \odot \mathbf{y} + \mu\sigma^2\mathbf{z}_k) \oslash (\mathbf{M} + \mu\sigma^2) \tag{4.19}$$

where $\oslash$ denotes elementwise division.

Figure 4.14 shows an example of IRCNN for image inpainting. As one can see, there is no visible artifacts in the inpainted image.

4.5 Challenges and Possible Solutions

While the image denoising for AWGN removal has been well-studied, little work has been done on real image denoising. The main difficulty arises from the fact that real noises are much more sophisticated than AWGN and it is not an easy task to thoroughly evaluate the performance of a denoiser. Figure 4.15 shows four typical noise types in real world. It can be seen that the characteristics of those noises are very different and a single noise level may be not enough to parameterize those noise types. In most cases, a denoiser can only work well under a certain noise model. For example, a denoising model trained for AWGN removal is not effective for mixed Gaussian and Poisson noise removal. This is intuitively reasonable because the CNN-based methods can be treated as general cases of Eq. (4.3) and the important data fidelity term corresponds to the degradation process. In spite of this, the image denoising for AWGN removal is still valuable due to the following reasons. First, it is an ideal test bed to evaluate the effectiveness of different CNN-based denoising models and learning algorithms. Second, in the unrolled inference

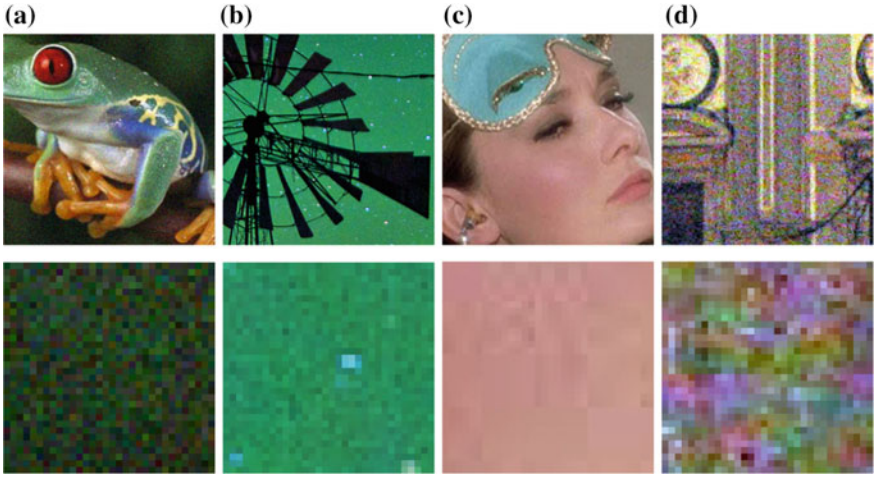


Fig. 4.15 Different noise types. **a** Additive white Gaussian noise; **b** interchannel correlated Gaussian noise; **c** JPEG compression noise; **d** low-frequency noise

via variable splitting techniques, many image restoration problems can be addressed by sequentially solving a series of Gaussian denoising subproblems, which further broadens the application fields.

To improve the practicability of a CNN denoiser, perhaps the most straightforward way is to capture adequate amounts of real noisy-clean training pairs for training so that the real degradation space can be covered. This solution has advantage that there is no need to know the complex degradation process. However, deriving the corresponding clean image of a noisy one is not a trivial task due to the need of careful postprocessing steps, such as spatial alignment and illumination correction. Alternatively, one can simulate the real degradation process to synthesize noisy images for a clean one. However, it is not easy to accurately model the complex degradation process. In particular, the noise model can be different across different cameras. Nevertheless, it is practically preferable to roughly model a certain noise type for training and then use the learned CNN model for type-specific denoising.

Besides the training data, the robust architecture and robust training also play vital roles for the success of a CNN denoiser. For the robust architecture, designing a deep multiscale CNN which involves a coarse-to-fine procedure is a promising direction. Such a network is expected to inherit the merits of multiscale [23]: (i) the noise level decreases at larger scales; (ii) the ubiquitous low-frequency noise can be alleviated by multiscale procedure; and (iii) downsampling the image before denoising can effectively enlarge the receptive field. For the robust training, the effectiveness of the denoiser trained with generative adversarial networks (GAN) for real image denoising still remains uninvestigated. The main idea of GAN-based denoising is to introduce an adversarial loss to improve the perceptual quality of denoised image. A distinctive advantage of GAN is that it can do unsupervised learning, and thus

is expected to be helpful in training denoising CNNs without ground-truth clean images.

Acknowledgement This work is partially supported by the National Natural Scientific Foundation of China (NSFC) under Grant No. 61671182 and 61471146, and the HK RGC GRF grant (under no. PolyU 152124/15E).

References

1. Ahn B, Cho NI (2017) Block-matching convolutional neural network for image denoising. arXiv:1704.00524
2. Bae W, Yoo J, Ye JC (2017) Beyond deep residual learning for image restoration: persistent homology-guided manifold simplification. In: The IEEE conference on computer vision and pattern recognition (CVPR) workshops, pp 145–153
3. Bako S, Vogels T, McWilliams B, Meyer M, Novák J, Harvill A, Sen P, DeRose T, Rousselle F (2017) Kernel-predicting convolutional networks for denoising Monte Carlo renderings. ACM Trans Gr 36(4):97
4. Barbu A (2009) Training an active random field for real-time image denoising. IEEE Trans Image Process 18(11):2451–2462
5. Burger HC, Schuler CJ, Harmeling S (2012) Image denoising: can plain neural networks compete with BM3D? In: IEEE conference on computer vision and pattern recognition, pp 2392–2399
6. Chambolle A, Pock T (2011) A first-order primal-dual algorithm for convex problems with applications to imaging. J Math Imaging Vis 40(1):120–145
7. Chan SH, Wang X, Elgendy OA (2017) Plug-and-Play ADMM for image restoration: fixed-point convergence and applications. IEEE Trans Comput Imaging 3(1):84–98
8. Chen Y, Pock T (2017) Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration. IEEE Trans Pattern Anal Mach Intell 39(6):1256–1272
9. Choi JH, Elgendy O, Chan SH (2017) Integrating disparate sources of experts for robust image denoising. arXiv:1711.06712
10. Colom M, Lebrun M, Buades A, Morel JM (2014) A non-parametric approach for the estimation of intensity-frequency dependent noise. In: IEEE international conference on image processing, pp 4261–4265
11. Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Trans Image Process 16(8):2080–2095
12. Godard C, Matzen K, Uyttendaele M (2017) Deep burst denoising. arXiv:1712.05790
13. Gu S, Zhang L, Zuo W, Feng X (2014) Weighted nuclear norm minimization with application to image denoising. In: IEEE conference on computer vision and pattern recognition, pp 2862–2869
14. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: ICCV, pp 1026–1034
15. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition, pp 770–778
16. Heide F, Steinberger M, Tsai YT, Rouf M, Pajak D, Reddy D, Gallo O, Liu J, Heidrich W, Egiazarian K et al (2014) FlexISP: a flexible camera image processing framework. ACM Trans Gr 33(6):231
17. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pp 448–456
18. Jain V, Seung S (2009) Natural image denoising with convolutional networks. In: Advances in neural information processing systems, pp 769–776

19. Kim Y, Jung H, Min D, Sohn K (2017) Deeply aggregated alternating minimization for image restoration. In: IEEE conference on computer vision and pattern recognition, pp 6419–6427
20. Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In: International conference for learning representations
21. Kligvasser I, Shaham TR, Michaeli T (2017) xUnit: learning a spatial activation function for efficient image restoration. arXiv:1711.06445
22. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
23. Lebrun M, Colom M, Morel JM (2015) The noise clinic: a blind image denoising algorithm. Image Process On Line 5:1–54. http://demo.ipol.im/demo/125/
24. Lee JS (1981) Refined filtering of image noise using local statistics. Comput Gr Image Process 15(4):380–389
25. Lefkimmiatis S (2017) Non-local color image denoising with convolutional neural networks. In: IEEE conference on computer vision and pattern recognition, pp 3587–3596
26. Lefkimmiatis S (2018) Universal denoising networks: a novel CNN-based network architecture for image denoising. In: IEEE conference on computer vision and pattern recognition
27. Levin A, Nadler B (2011) Natural image denoising: optimality and inherent bounds. In: IEEE conference on computer vision and pattern recognition, pp 2833–2840
28. Liu C, Szeliski R, Kang SB, Zitnick CL, Freeman WT (2008) Automatic estimation and removal of noise from a single image. IEEE Trans Pattern Anal Mach Intell 30(2):299–314
29. Mildenhall B, Barron JT, Chen J, Sharlet D, Ng R, Carroll R (2017) Burst denoising with kernel prediction networks. arXiv:1712.02327
30. Pan J, Hu Z, Su Z, Yang MH (2014) Deblurring text images via L0-regularized intensity and gradient prior. In: IEEE conference on computer vision and pattern recognition, pp 2901–2908
31. Remez T, Litany O, Giryes R, Bronstein AM (2017) Deep class-aware image denoising. In: International conference on sampling theory and applications, pp 138–142
32. Riegler G, Schulter S, Ruther M, Bischof H (2015) Conditioned regression models for non-blind single image super-resolution. In: IEEE international conference on computer vision, pp 522–530
33. Romano Y, Elad M, Milanfar P (2016) The little engine that could: Regularization by denoising (RED). SIAM J Imaging Sci (submitted)
34. Roth S, Black MJ (2005) Fields of experts: a framework for learning image priors. In: IEEE computer society conference on computer vision and pattern recognition, vol 2, pp 860–867
35. Roth S, Black MJ (2009) Fields of experts. Int J Comput Vis 82(2):205–229
36. Samuel KG, Tappen MF (2009) Learning optimized MAP estimates in continuously-valued MRF models. In: IEEE conference on computer vision and pattern recognition, pp 477–484
37. Santhanam V, Morariu VI, Davis LS (2017) Generalized deep image to image regression. In: IEEE conference on computer vision and pattern recognition, pp 5609–5619
38. Schmidt U, Roth S (2014) Shrinkage fields for effective image restoration. In: IEEE conference on computer vision and pattern recognition, pp 2774–2781
39. Shi W, Caballero J, Huszár F, Totz J, Aitken AP, Bishop R, Rueckert D, Wang Z (2016) Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: IEEE conference on computer vision and pattern recognition, pp 1874–1883
40. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference for learning representations
41. Sreehari S, Venkatakrishnan S, Wohlberg B, Drummy LF, Simmons JP, Bouman CA (2015) Plug-and-play priors for bright field electron tomography and sparse interpolation. arXiv:1512.07331
42. Sun J, Tappen MF (2011) Learning non-local range Markov random field for image restoration. In: IEEE conference on computer vision and pattern recognition, pp 2745–2752
43. Sun J, Tappen MF (2013) Separable Markov random field model and its applications in low level vision. IEEE Trans Image Process 22(1):402–407
44. Vedaldi A, Lenc K (2015) MatConvNet: convolutional neural networks for matlab. In: ACM conference on multimedia conference, pp 689–692

45. Vogel C, Pock T (2017) A primal dual network for low-level vision problems. In: German conference on pattern recognition. Springer, pp 189–202
46. Wang T, Qin Z, Zhu M (2017) An ELU network with total variation for image denoising. In: International conference on neural information processing. Springer, pp 227–237
47. Wang T, Sun M, Hu K (2017) Dilated residual network for image denoising. arXiv:1708.05473
48. Yang D, Sun J (2018) Bm3d-net: a convolutional neural network for transform-domain collaborative filtering. IEEE Signal Process Lett 25(1):55–59
49. Yu F, Koltun V (2016) Multi-scale context aggregation by dilated convolutions. In: International conference on learning representations
50. Zhang K, Zuo W, Chen Y, Meng D, Zhang L (2017) Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. IEEE Trans Image Process 26(7):3142–3155
51. Zhang K, Zuo W, Gu S, Zhang L (2017) Learning deep CNN denoiser prior for image restoration. In: IEEE conference on computer vision and pattern recognition, pp 3929–3938
52. Zhang K, Zuo W, Zhang L (2018) Learning a single convolutional super-resolution network for multiple degradations. In: IEEE conference on computer vision and pattern recognition
53. Zoran D, Weiss Y (2011) From learning models of natural image patches to whole image restoration. In: IEEE international conference on computer vision, pp 479–486