Deep learning algorithm for Gaussian noise removal from images

Mickael Aghajarian, John E. McInroy, and Suresh Muknahallipatna
University of Wyoming, College of Engineering and Applied Science,
Department of Electrical and Computer Engineering, Laramie, Wyoming, United States

Abstract. A deep learning algorithm for Gaussian noise removal from both grayscale and color images is developed. As opposed to most existing discriminative methods that train a specific model for each noise level, the proposed method can handle a wide range of noise levels using only two trained models, one for low noise levels and the other for high noise levels. In the proposed algorithm, the training process consists of three successive steps. In the first step, a classifier is trained to classify the noisy and clean images. In the second step, a denoiser network aims to remove the noise in the image features that are extracted by the trained classifier. Finally, a decoder is utilized to map back the denoised images features into images pixels. To evaluate the performance of the model, the Berkeley segmentation dataset of 68 images (BSDS68) and 12 widely used images are used, and the denoising performance for additive white Gaussian noise is compared with several state-of-the-art methods in terms of peak signal-to-noise ratio (PSNR) and visual quality. For grayscale image denoising of BSDS68, our method gives the highest PSNR on all noise levels (significant mean improvement of 0.99). For color image denoising of BSDS68, except for one low noise level, the proposed method gives the highest PSNR on all other noise levels (mean improvement of 0.3). [DOI: 10.1117/1.JEI.29.4.043005]

Keywords: deep learning; image denoising; Gaussian noise; convolutional neural networks.

Paper 200171 received Mar. 2, 2020; accepted for publication Jun. 29, 2020; published online Jul. 10, 2020.

1 Introduction

Image denoising is a fundamental image restoration task that has stimulated considerable attention in both academic and industrial research.[1–3] The goal of image denoising is to recover a clean noise-free image x from a corrupted noisy image y that follows an image degradation model $y = x + N$, where N is measurement noise. One common assumption in the literature, which we also adopt in this work, is that N is additive white Gaussian noise (AWGN) with zero mean and standard deviation σ. Numerous and diverse algorithms have been proposed for the Gaussian noise removal problem. The most effective image denoising algorithms can be generally categorized into two main categories: nonlocal methods [e.g., block matching three-dimensional (BM3D),[4] weighted nuclear norm minimization (WNNM),[5] expected patch log likelihood (EPLL),[6] and multilayer perceptron (MLP)[7]] and local methods [e.g., feed-forward denoising convolutional neural network (DnCNN),[8] trainable nonlinear reaction diffusion (TNRD),[9] and the method proposed by Lefkimmiatis.[10]]. The self-similarities and image priors are widely exploited by effective methods such as BM3D and its modified version for color images (CBM3D).[4] The BM3D method, proposed by Dabov et al.,[4] is based on an enhanced sparse representation in the transform domain; it groups similar two-dimensional (2-D) image fragments into 3-D data arrays and then uses collaborative filtering to deal with these 3-D data arrays. Gu et al.[5] studied standard nuclear norm minimization and proposed the WNNM in which the singular values are assigned different weights. They applied the proposed WNNM algorithm to image denoising by exploiting the image nonlocal self-similarity. As Zoran and Weiss[6] showed, image priors that give high likelihood to data perform better in patch restoration.

*Address all correspondence to Mickael Aghajarian, E-mail: maghajar@uwyo.edu

1017-9909/2020/$28.00

Motivated by the idea that patches in the restored image should be likely under the prior, they proposed the EPLL method that uses patch models for whole image restoration. Burger et al.[7] proposed a plain MLP model that attempts to learn the mapping from a noisy image to a noise-free image. They showed that, by training on large image datasets, MLP is able to compete with other image denoising methods.

However, during the past decade, nonlocal image denoising methods started to converge, and there has been little progress in improving the state-of-the-art denoising performance. Levin et al.[11,12] derived a statistical measure for the best possible denoising results using a fixed support around a denoised pixel and generic natural image prior. They showed that, for small windows, state-of-the-art denoising methods cannot be further improved beyond fractional decibel (dB) values. Moreover, nonlocal image denoising methods give an inferior performance on pseudorandom textures or singular features, where the images have weak self-similarities.[13]

Recently, due to the advent of deep learning algorithms, several effective image denoising methods that could improve the state-of-the-art denoising performance have been proposed.[10,14–16] Nevertheless, most of the existing denoising algorithms usually train a specific model for a certain noise level, so their denoising performance deteriorates dramatically when the noise deviates from the level for which the model was trained.

In this paper, we propose a deep learning algorithm that removes Gaussian noise from both grayscale and color images. As opposed to most of the existing denoising algorithms, which aim to remove the noise by learning the dense mapping from clean to noisy images, the proposed method aims to explicitly remove the noise from images features and then map the denoised features back into images pixels. Moreover, our method is able to handle a wide range of noise levels using only two trained models, one for low noise levels and the other for high noise levels.

The outline of this paper is as follows: Sec. 2 provides a brief survey of related work. The proposed method is described in Sec. 3. Section 4 presents the experimental results and draws a comparison with other state-of-the-art image denoising methods. Finally, Sec. 5 provides the conclusion.

2 Related Work

Recently, deep learning algorithms, particularly convolutional neural networks (CNNs), have shown great success in image restoration tasks. Jain and Seung[17] introduced an approach that combined the use of CNNs as an image processing architecture and an unsupervised learning procedure that synthesized training samples from specific noise models. They achieved comparable performance to wavelet and Markov random field (MRF) methods. They also showed how CNNs are mathematically related to MRF approaches while CNNs avoid computational difficulties in MRF approaches that arise from probabilistic learning and inference. Schmidt and Roth[18] proposed a deep network that offered both computational efficiency and high restoration quality for a certain noise level. They proposed a cascade of shrinkage fields (CSF), a random field-based architecture that combines the image model and the optimization algorithm in a single unit. Chen and Pock[9] proposed a discriminative training-based method (TNRD) that uses a dynamic nonlinear reaction-diffusion model with time-dependent parameters. In their proposed method, all of the parameters are simultaneously learned from training data through a loss-based approach. Although CSF and TNRD achieve promising denoising results, their priors are based on the analysis model, which is limited in capturing the full characteristics of image structures, and, therefore, their performance is inherently restricted to the specified forms of prior. Moreover, many parameters that need to be learned are involved. Another limitation of these methods is that they train a specific model for a known noise level, so their performance deteriorates dramatically when the noise deviates from the levels for which the models are trained.[8]

Zhang et al.[8] proposed deep CNN models that were able to handle Gaussian denoising with known and unknown noise levels. Their model for Gaussian denoising with known specific noise level is called DnCNN-S while the model for blind Gaussian denoising tasks is called DnCNN-B. They utilized batch normalization and residual learning to speed up the training process and boost the denoising performance. Lefkimmiatis[10] presented a novel network architecture applicable to a wide range of noise levels for grayscale and color image denoising. Inspired by local

and nonlocal variational methods, he utilized both convolutional layers and nonlocal filtering layers. His model has a local operator that uses convolutional layers as a core component and is referred to as $UNet_5$; the second model that relies on nonlocal filtering layers is referred to as $UNLNet_5$. The architecture was considerably shallower than state-of-the-art deep CNN-based networks and achieved excellent results under AWGN. Zhang et al.[16] proposed a fast and flexible denoising network (FFDNet) with a tunable noise level map as the input. The proposed FFDNet is able to deal with different noise levels and spatially variant noise.

3 Proposed Network

The input of the network is a noisy observation $y = x + N$. Discriminative denoising methods aim to learn a mapping function $F(y) = x$ to predict the latent clean image. The residual learning strategy, on the other hand, implicitly removes the latent clean image in the hidden layers by learning a residual mapping $R(y) \sim N$, and then the clean image is obtained by $x = y - R(y)$. In this work, the network removes the noise in the hidden layers explicitly. The training process of the network consists of three successive steps that are explained in detail in the following sections.

3.1 *Classifier*

In the first step, a CNN classifier is used to classify the noisy and clean images. It is worth noting that the main purpose of this step is not classification per se but rather training a classifier that is able to extract discriminative image features. The extracted features (i.e., noisy and clean images features) are used as the training and test data in the next step in which the denoiser removes the noise from the image features. Figure 1 shows the architecture of the CNN classifier. The activation function of all convolution layers is rectified linear unit (ReLU)[19] except for the last convolution layer, which is followed by the sigmoid activation function. The third layer consists of two parallel convolution layers,[20] each of which has 64 convolution filters of different sizes (3×3 versus 5×5). Using 5×5 convolution filters, the size of the receptive fields of the convolution filters is increased, so convolution filters are connected to larger local regions. The parallel convolution layers are concatenated and fed into a batch normalization (BN) layer[21] that is used to speed up the training process. The reason for using two separate max-pooling layers in series is that the output of the first max-pooling layer (with pooling filter of size 2×2 and stride of 2) is used as the extracted image features that are fed into the next step (i.e., denoiser) while the output of the second max-pooling layer (with pooling filter of size 8×8 and stride of 8) is used for the classification purpose. Three extracted image features from clean and noisy images are shown for the visualization purpose in Fig. 2. It is worth noting that the width and height of the extracted images features are half of the width and height of the input images because of the first max-pooling layer. Since these extracted image features with reduced size are fed into the next step (i.e., denoiser), the number of operations in the next step is decreased. Unlike most CNN classifiers that use fully connected layers as the last layers, we use just one convolution layer instead. Since the input of the last convolution layer is of size 20×30, the size of the last convolution filter is chosen to be 20×30. By replacing the fully connected layers by the convolution layer, the number of parameters decreases dramatically while still achieving very high classification accuracies. The number of parameters in the last convolution layer is 601 $[20 \times 30 + 1(\text{bias})]$. If only one FC layer containing 512 neurons is used, the number of parameters in the FC layer (not counting biases) would be as follows: the number of parameters using only one FC layer $= 20 \times 30 \times 190 \times 512 = 58{,}368{,}000$ parameters. Moreover, the local information of the activation maps is preserved, which would not be the case for fully connected layers due to vectorization.

3.2 *Denoiser*

As opposed to most of the existing denoising approaches, which aim to remove the noise by learning the dense mapping from clean to noisy images, the denoiser aims to explicitly remove

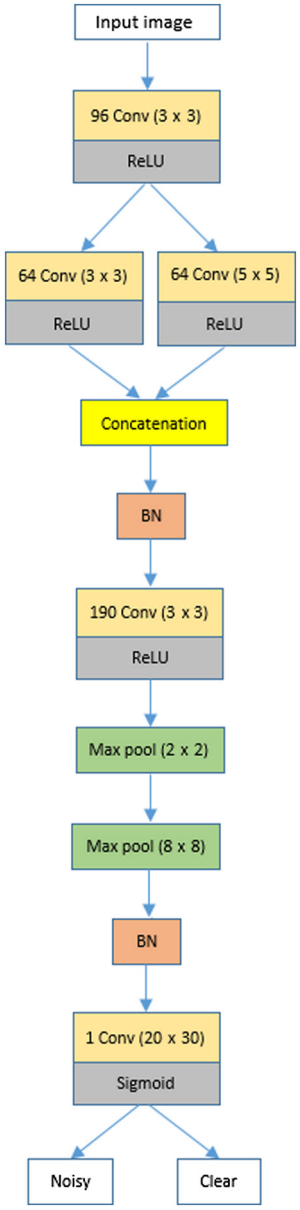

Fig. 1 Classifier architecture.

the noise from image features that are extracted by the trained classifier. The architecture of the denoiser is illustrated in Fig. 3. The denoiser consists of two convolution layers, each of which has 190 convolution filters of size 3×3. The activation function of the first and second layers is ReLU and sigmoid, respectively. It is worth noting that the input and output of the denoiser are the noisy and clean image features of the same size, respectively. In other words, the training set for the denoiser consists of the noisy and clean image features that are extracted by the trained CNN classifier. Since there are 190 convolution filters in the convolution layer right before the first max-pooling layer in the classifier, 190 images features are extracted for clean and noisy images.

3.3 *Decoder*

In the last step of the training process, a decoder is trained to map back the denoised image features into the image pixels. In other words, the input of the decoder is denoised image features while its output is clean images. Figure 4 shows the architecture of the proposed decoder. Since

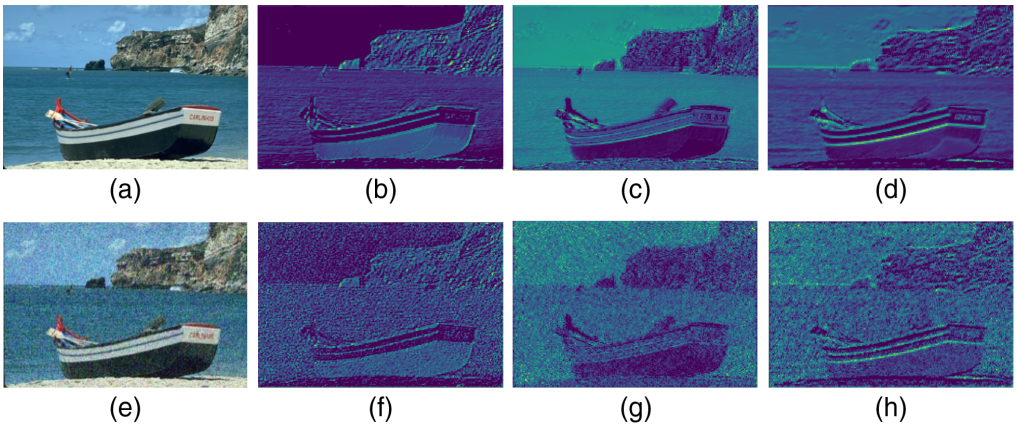

Fig. 2 Clean and noisy image features extracted by the trained classifier: (a) clean image, (b) clean image feature extracted by convolution filter #1, (c) clean image feature extracted by convolution filter #100, (d) clean image feature extracted by convolution filter #190, (e) noisy image, (f) noisy image feature extracted by convolution filter #1, (g) noisy image feature extracted by convolution filter #100, and (h) noisy image feature extracted by convolution filter #190.

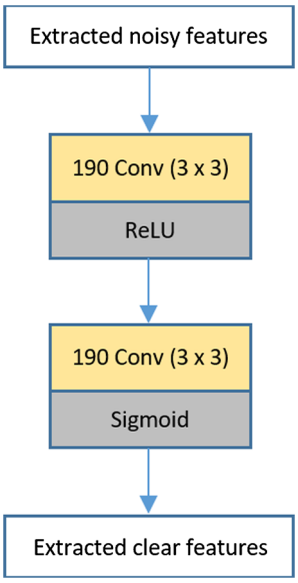

Fig. 3 Denoiser architecture.

the extracted image features are the output of the first max-pooling layer of the classifier, one upsampling layer is used in the decoder to map the size of the features back to the original size. Because there are three channels in color images, the last layer must have three convolution filters to match the color images. Likewise, the last layer of the decoder for grayscale images has one convolution filter. The pixel intensities of input images are normalized so that they have values between zero and one, so the activation function of the last convolution layer is chosen to be sigmoid. To rescale the normalized pixel intensities, they are multiplied by 255 and rounded to the nearest integer number. Figure 5 shows how the three trained networks (i.e., the classifier, denoiser, and decoder) work together.

4 Experimental Results

In this work, the Berkeley segmentation dataset (BSDS),[22] which consists of 500 natural images, was used to train and test our model. The training and validation data contained 400 and 100

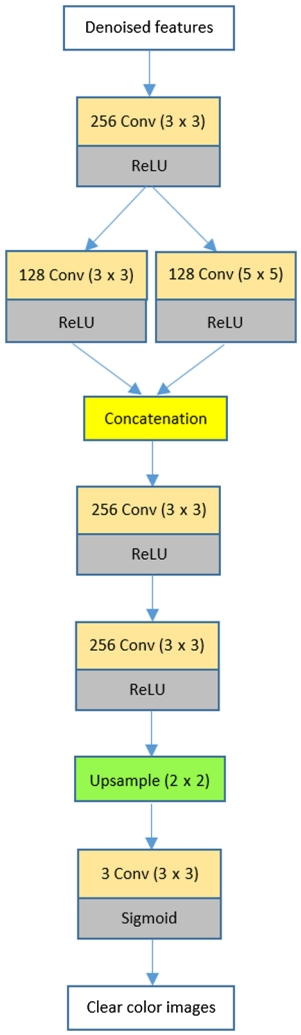

Fig. 4 Decoder architecture.

images of size 320×480, respectively. For low noise levels, images were corrupted with AWGN with a standard deviation that varied from $\sigma = 15$ to $\sigma = 25$ with increments of 5 while for high noise levels, the standard deviation varied from $\sigma = 35$ to $\sigma = 55$ with increments of 10. Although we use the same networks architecture for both high and low noise levels, the networks are trained separately for each case. By doing so, its denoising performance is better than a single network trained for the entire noise densities. The reason for this could be that the optimal weights for denoising images corrupted by high and low noise might be different. By training a single model for both cases, some weights might be overwritten, which could deteriorate the denoising performance.

To draw a comparison with other state-of-the-art algorithms, two widely used datasets were utilized. The first one was the standard evaluation dataset of 68 images (BSDS68),[23] and the other dataset contained 12 widely used grayscale images (Set12 dataset). It is worth mentioning that the BSDS68 images of Ref. 23 were strictly excluded from the training data and were only used in the validation data. Our model was trained on an NVIDIA Quadro RTX 6000 GPU. We used the Keras neural network library to train and test the model. To train the classifier and decoder, we used Adam optimizer[24] with a learning rate of 0.00015 and default parameters while for the denoiser, we used the default learning rate of 0.001.

Table 1 shows the classification accuracies for color and grayscale images corrupted by low and high noise levels. It is worth noting that the main purpose of the classifier is not classification per se but rather extracting discriminative image features that are fed into the denoiser. The very high classification accuracies show that the extracted images features are discriminative.

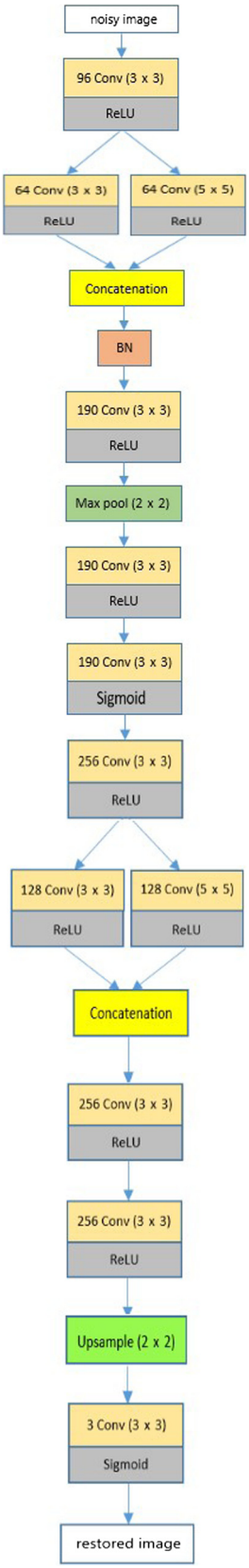

Fig. 5 The three trained networks together.

4.1 *Compared Denoising Methods*

In this work, we made a detailed comparison with several state-of-the-art denoising algorithms, including two nonlocal similarity-based methods (BM3D[4] and WNNM[5]), one generative method (EPLL[6]), three discriminative training-based methods (MLP,[7] CSF,[18] and TNRD[9]), and five CNN-based methods ($UNet_5$,[10] $UNLNet_5$,[10] DnCNN-S,[8] DnCNN-B,[8] and FFDNet[16]). All methods are described in either in Secs. 1 or 2. Implementation codes of the methods with default parameter settings are downloaded from the authors' websites for visual comparison. The peak signal-to-noise ratio (PSNR) results of other methods are taken from Refs. 8 and 10.

4.2 *BSDS68 Grayscale Images Denoising*

Table 2 demonstrates the average PSNR results of different methods on the BSDS68 test images, where the best PSNR result for each noise level is highlighted in bold. As can be seen, our model achieved the highest PSNR on all noise levels with large margins. The proposed method outperformed DnCNN, the current state-of-the-art method, by 0.99 in terms of average PSNR (dB), which is a dramatic improvement.

4.3 *BSDS68 Color Images Denoising*

The PSNR results of different methods for color image denoising are shown in Table 3. Except for one low noise level with $\sigma = 15$, our method achieved the highest PSNR on all other noise levels. Like grayscale images denoising, our method outperformed the current state-of-the-art method (CDnCNN) by 0.12 in terms of average PSNR. Figure 6 shows the average PSNR improvement of our method over DnCNN-S and CDnCNN-B for grayscale and color image denoising, respectively.

Figures 7 and 8 show the visual comparison of our method's color image denoising results with CBM3D and CDnCNN-B for $\sigma = 20$ and $\sigma = 45$, respectively. It can be seen that our method can recover corrupted images with natural colors that yield visually pleasant results while preserving sharp edges and fine details of the images. It is worth noting that the visual

Table 1 Classification accuracies for grayscale and color images.

	Color images		Grayscale images	
	Low noise	High noise	Low noise	High noise
Classification accuracy	97%	99.5%	99%	99.5%

Table 2 Grayscale denoising comparison for different noise levels over the BSDS68 images in terms of average PSNR (dB).

	Noise level (standard deviation σ)						
Method	15	20	25	35	45	55	Average
BM3D	31.06	29.60	28.55	27.07	25.99	25.26	27.92
EPLL	31.18	29.73	28.67	27.16	26.09	25.34	28.03
WNNM	31.31	29.83	28.73	27.28	26.26	25.49	28.15
DnCNN-B	31.60	30.19	29.15	27.66	26.62	25.80	28.50
$UNet_5$	31.38	29.93	28.84	27.38	26.38	25.53	28.24
$UNLNet_5$	31.47	30.04	28.96	27.50	26.48	25.64	28.35
Our method	**32.20**	**31.45**	**30.25**	**28.64**	**27.68**	**26.70**	**29.49**

Table 3 Color image denoising comparison for different noise levels over the BSDS68 images in terms of average PSNR (dB).

	Noise level (standard deviation σ)						
Method	15	20	25	35	45	55	Average
CBM3D	33.49	31.88	30.68	28.86	27.82	26.95	29.95
CDnCNN-B	33.88	32.36	31.22	29.57	28.39	27.45	30.48
$CUNet_5$	33.78	32.21	31.03	29.37	28.23	27.27	30.32
$CUNLNet_5$	**33.90**	32.34	31.17	29.53	28.37	27.44	30.46
Our method	33.11	**32.39**	**31.47**	**29.67**	**28.94**	**28.03**	**30.60**

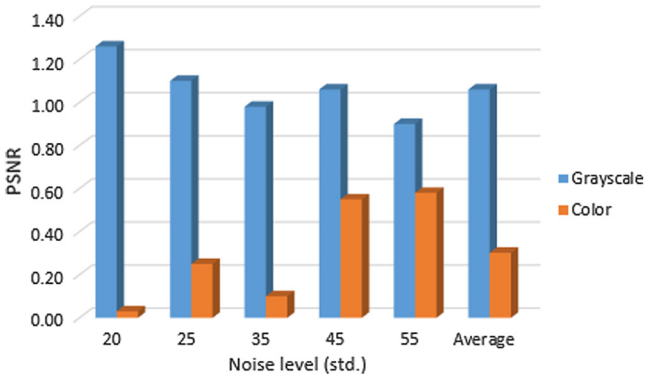

Fig. 6 Average PSNR improvement of our method over DnCNN-S and CDnCNN-B methods.

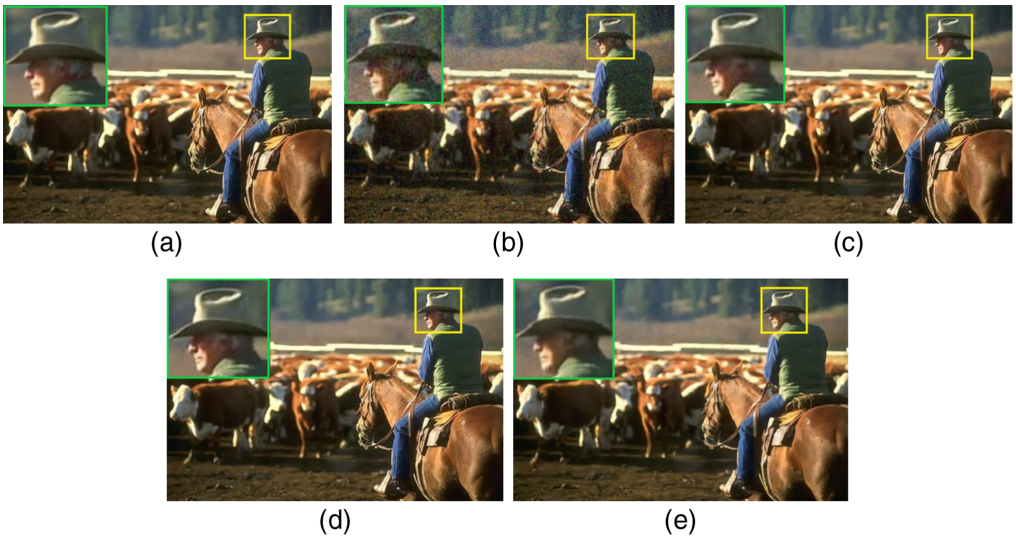

Fig. 7 Color image denoising results of one image from the BSDS68 dataset corrupted by AWGN of $\sigma = 20$. (a) Clear, (b) noisy (25.09 dB), (c) CBM3D (32.88 dB), (d) CDnCNN-B (33.51 dB), and (e) our method (33.04 dB).

comparison of denoising methods might not be useful in some cases because the human eye is more sensitive to edges compared to smooth regions of images. Taking this into account, some denoising methods might do a slightly better job at the edges of images, but it does not necessarily mean that they do a better job for the entire image including smooth regions containing no edges. That is why the quantitative criteria (e.g., PSNR) have been used in the literature to compare the performance of different denoising algorithms.

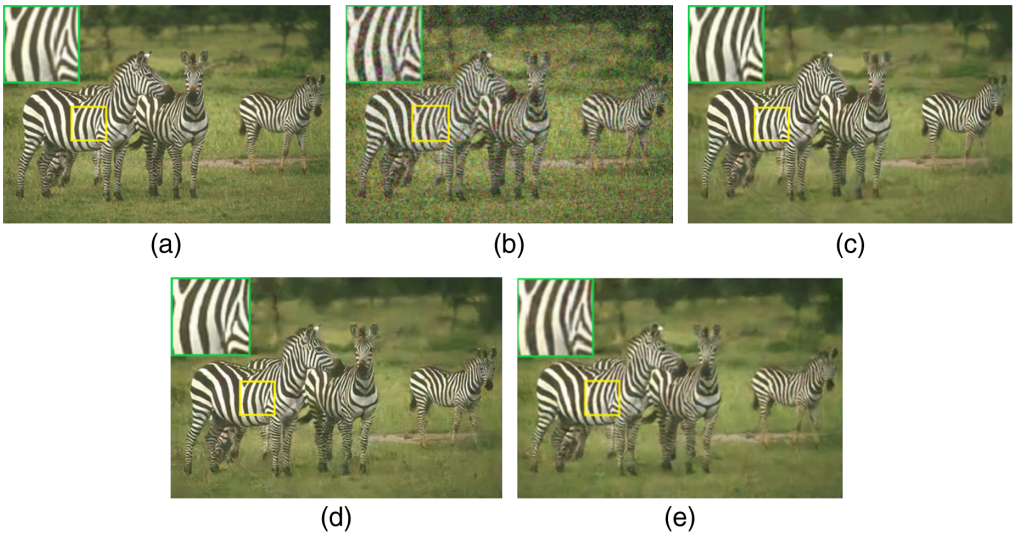

Fig. 8 Color image denoising results of one image from the BSDS68 dataset corrupted by AWGN of $\sigma = 45$. (a) Clear, (b) noisy (15.07 dB), (c) CBM3D (26.97 dB), (d) CDnCNN-B (27.87 dB), and (e) our method (28.36 dB).

4.4 *Blind Denoising*

To further show the capacity of our model, we evaluated its performance for denoising BSDS68 images corrupted by the noise levels for which our model was never trained (i.e., Gaussian noise with a standard deviation of 30, 40, and 50). For $\sigma = 30$, the trained model for low noise level was used while for $\sigma = 40$ and $\sigma = 50$, the trained model for high noise level was utilized. The average PSNR results for grayscale and color image denoising are shown in Tables 4 and 5, respectively. Surprisingly, for all three noise levels of grayscale image denoising, our model outperformed all other methods, some of which were specifically trained for that known noise level. Compared to the current state-of-the-art method (DnCNN), our model led to an average PSNR increase of 0.82 dB, which is a noticeable improvement.

4.5 *Set12 Grayscale Denoising*

The 12 widely used test images (i.e., Set12), used in Ref. 8, are utilized to draw a comparison with other methods. The PSNR results of different denoising algorithms on the Set12 images

Table 4 Grayscale BSDS68 images denoising comparison for noise levels for which our model was never trained.

	Noise level (standard deviation σ)			
Method	30	40	50	Average
BM3D	27.74	26.45	25.60	26.60
EPLL	27.84	26.58	25.71	26.71
WNNM	27.94	26.72	25.85	26.84
DnCNN-B	28.33	27.10	26.21	27.21
$UNet_5$	28.01	26.85	25.95	26.94
$UNLNet_5$	28.13	26.96	26.04	27.04
Our method	**28.74**	**28.10**	**27.24**	**28.03**

Table 5 Color BSDS68 images denoising comparison for noise levels for which our model was never trained.

	Noise level (standard deviation σ)			
Method	30	40	50	Average
CBM3D	29.71	28.06	27.36	28.38
CDnCNN-B	**30.31**	28.94	27.91	29.05
$CUNet_5$	30.06	28.77	27.74	28.86
$CUNLNet_5$	30.24	28.91	27.89	29.01
Our method	30.22	**29.32**	**28.51**	**29.35**

Table 6 The PSNR (dB) results of different methods on the Set12 images corrupted by AWGN of $\sigma = 25$. The best PSNR result for each noise level is highlighted in bold.

Noise level: $\sigma = 25$													
	C.man	House	Peppers	Starfish	Monarch	Airplane	Parrot	Lena	Barbara	Boat	Man	Couple	Average
BM3D	29.45	32.85	30.16	28.56	29.25	28.42	28.93	32.07	30.71	29.9	29.61	29.71	29.97
WNNM	29.64	33.22	30.42	29.03	29.84	28.69	29.15	32.24	**31.24**	30.03	29.76	29.82	30.26
EPLL	29.26	32.17	30.17	28.51	29.39	28.61	28.95	31.73	28.61	29.74	29.66	29.53	29.69
MLP	29.61	32.56	30.30	28.82	29.61	28.82	29.25	32.25	29.54	29.97	29.88	29.73	30.03
CSF	29.48	32.39	30.32	28.8	29.62	28.72	28.90	31.79	29.03	29.76	29.71	29.53	29.84
TNRD	29.72	32.53	30.57	29.02	29.85	28.88	29.18	32.00	29.41	29.91	29.87	29.71	30.06
DnCNN-S	**30.18**	33.06	**30.87**	29.41	30.28	**29.13**	**29.43**	32.44	30.00	30.21	30.1	30.12	**30.44**
DnCNN-B	29.94	33.05	30.84	29.34	30.25	29.09	29.35	32.42	29.69	30.20	30.09	30.10	30.36
FFDNet	30.06	**33.27**	30.79	29.33	30.14	29.05	**29.43**	32.59	29.98	30.23	30.10	30.18	30.43
Our method	29.20	33.02	30.29	**29.82**	**30.68**	28.98	28.77	**32.98**	28.96	**30.45**	**30.44**	**30.33**	30.33

Table 7 The PSNR (dB) results of different methods on the Set12 images corrupted by AWGN of $\sigma = 50$. The best PSNR result for each noise level is highlighted in bold.

Noise level: $\sigma = 50$													
	C.man	House	Peppers	Starfish	Monarch	Airplane	Parrot	Lena	Barbara	Boat	Man	Couple	Average
BM3D	26.13	29.69	26.68	25.04	25.82	25.10	25.9	29.05	27.22	26.78	26.81	26.46	26.72
WNNM	26.45	30.33	26.95	25.44	26.32	25.42	26.14	29.25	**27.79**	26.97	26.94	26.64	27.05
EPLL	26.10	29.12	26.8	25.12	25.94	25.31	25.95	28.68	24.83	26.74	26.79	26.30	26.47
MLP	26.37	29.64	26.68	25.43	26.26	25.56	26.12	29.32	25.24	27.03	27.06	26.67	26.78
TNRD	26.62	29.48	27.10	25.42	26.31	25.59	26.16	28.93	25.70	26.94	26.98	26.50	26.81
DnCNN-S	**27.03**	30.00	27.32	25.70	26.78	25.87	26.48	29.39	26.22	27.20	27.24	26.90	27.18
DnCNN-B	**27.03**	30.02	27.39	25.72	26.83	25.89	26.48	29.38	26.38	27.23	27.23	26.91	27.21
FFDNet	**27.03**	30.43	**27.43**	25.77	26.88	**25.90**	**26.58**	29.68	26.48	27.32	27.30	**27.07**	**27.32**
Our method	26.66	**30.48**	27.16	**26.35**	**27.17**	25.97	26.01	**29.94**	25.10	**27.37**	**27.49**	27.04	27.23

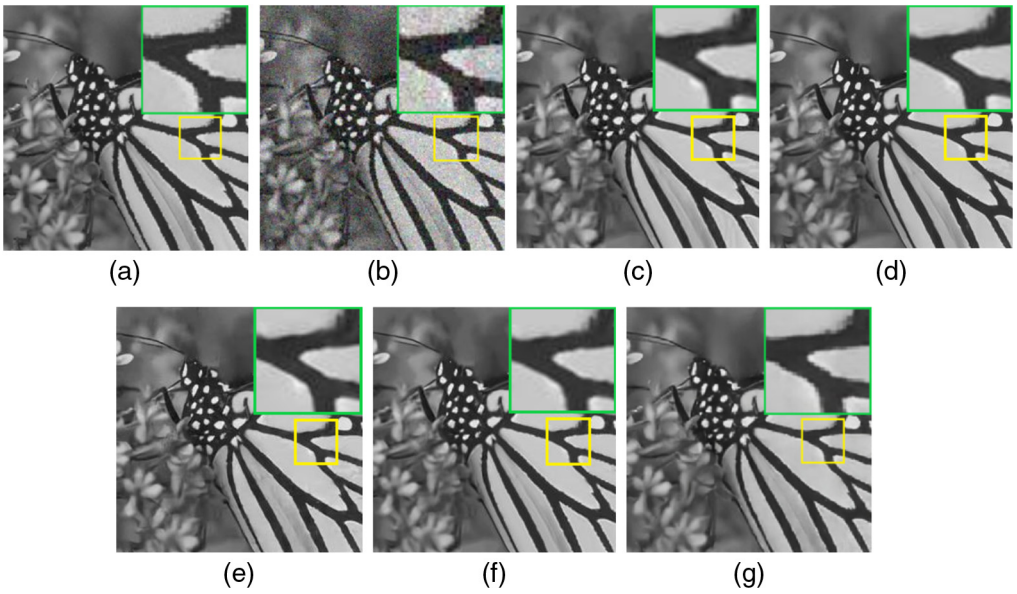

Fig. 9 Denoising results of the image "monarch" corrupted by AWGN of $\sigma = 25$: (a) original image, (b) noisy image, (c) denoised image using BM3D (PSNR = 29.25), (d) denoised image using DnCNN-S (PSNR = 30.23), (e) denoised image using TNRD (PSNR = 29.86), (f) denoised image using WNNM (PSNR = 29.84), and (g) denoised image using our method (PSNR = 30.68).

corrupted by AWGN with a standard deviation of 25 and 50 are given in Tables 6 and 7, respectively. The best PSNR result for each noise level is highlighted in bold. For $\sigma = 25$, our model achieved the highest PSNR on half of the images, while for the other half it yielded comparable results to other methods. On average, the proposed method outperformed all other algorithms except for DnCNN-S and DnCNN-B. Our achieved average PSNR on all 12 images is 0.11 and 0.03 dB lower than those of DnCNN-S and DnCNN-B, respectively.

For $\sigma = 50$, the proposed method yielded the highest PSNR on six images. On average, the proposed method outperformed all other algorithms except for FFDNet. Our achieved average PSNR on all 12 images is 0.09 lower than that of FFDNet. It should be noted that our model was never trained for $\sigma = 50$ and yet outperformed other methods, some of which were specifically trained for that noise level. It is also worth mentioning that for image "house," which is dominated by repetitive structures, our method outperformed nonlocal similarity-based methods such as BM3D and WNNM that usually work better on images with regular and repetitive structures. Figure 9 shows the visual comparisons between our method and the BM3D, DnCNN-S, TNRD, and WNNM methods.

5 Conclusion

In this paper, we proposed a deep learning algorithm for grayscale and color image denoising. Unlike most existing discriminative models that train a specific model for each noise level, our model can handle a wide range of noise levels using two trained models only, one for low noise levels (i.e., AWGN with a standard deviation of 15, 20, and 25) and the other for high noise levels (i.e., AWGN with a standard deviation of 35, 45, and 55). To evaluate the performance of our model, the BSDS of 68 images (BSDS68) and 12 widely used images were used and the denoising performance was compared with other state-of-the-art methods in terms of PSNR and visual quality. In the vast majority of the cases, our proposed method gave the highest PSNR and outperformed other state-of-the-art models. The results clearly verify the validity of both the design of the networks (e.g., number of convolutional filters and types of layers) and the idea behind the algorithm (i.e., removing noise from image features instead of image pixels).

Acknowledgments

The authors certify that they have no affiliations with or involvement in any organization or entity with any financial or nonfinancial interest in the subject matter or materials discussed in this paper.

References

1. M. Aghajarian, J. E. McInroy, and C. H. G. Wright, "Salt-and-pepper noise removal using modified mean filter and total variation minimization," *J. Electron. Imaging* **27**(1), 013002 (2018).
2. A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *Proc. of the IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognit.*, pp. 1692–1700 (2018).
3. P. Milanfar, "A tour of modern image filtering: new insights and methods, both practical and theoretical," *IEEE Signal Process. Mag.* **30**(1), 106–128 (2012).
4. K. Dabov et al., "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.* **16**(8), 2080–2095 (2007).
5. S. Gu et al., "Weighted nuclear norm minimization with application to image denoising," in *Proc. of the IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognit.*, pp. 2862–2869 (2014).
6. D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *IEEE Int. Conf. on Comput. Vision*, pp. 479–486 (2011).
7. H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: can plain neural networks compete with BM3D?" in *Proc. of the IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognit.*, pp. 2392–2399 (2012).
8. K. Zhang et al., "Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.* **26**(7), 3142–3155 (2017).
9. Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(6), 1256–1272 (2017).
10. S. Lefkimmiatis, "Universal denoising networks: a novel CNN architecture for image denoising," in *Proc. of the IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognit.*, pp. 3204–3213 (2018).
11. A. Levin et al., "Patch complexity, finite pixel correlations and optimal denoising," *Eur. Conf. Comput. Vis.* **7576**(5), 73–86 (2012).
12. A. Levin and B. Nadler, "Natural image denoising: optimality and inherent bounds," in *Proc. of the IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognit.*, pp. 2833–2840 (2011).
13. C. Cruz et al., "Nonlocality-reinforced convolutional neural networks for image denoising," *IEEE Signal Process. Lett.* **25**(8), 1216–1220 (2018).
14. K. Zhang et al., "Learning deep CNN denoiser prior for image restoration," in *Proc. of the IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognit.*, pp. 2808–2817 (2017).
15. R. Vemulapalli, O. Tuzel, and M. Y. Liu, "Deep Gaussian conditional random field network: a model-based deep network for discriminative denoising," in *IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognit.*, pp. 4801–4809 (2016).
16. K. Zhang, W. Zuo, and L. Zhang, "FFDNet: toward a fast and flexible solution for CNN-based image denoising," *IEEE Trans. Image Process.* **27**(9), 4608–4622 (2018).
17. V. Jain and S. Seung, "Natural image denoising with convolutional networks," *Adv. Neural Inf. Process. Syst.* **21**, 769–776 (2009).
18. U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *Proc. of the IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognit.*, pp. 2774–2781 (2014).
19. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Int. Conf. on Artif. Intell. and Sta.*, pp. 315–323 (2011).
20. C. Szegedy et al., "Going deeper with convolutions," in *Proc. of the IEEE Comput. Soc. Conf. on Comput. Vision and Pattern Recognit.*, pp. 1–9 (2015).

21. S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Int. Conf. on Mach. Learn.*, pp. 448–456 (2015).
22. D. Martin et al., "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. of the IEEE Int. Conf. on Comput. Vision*, pp. 416–423 (2001).
23. S. Roth and M. J. Black, "Fields of experts," *Int. J. Comput. Vis.* **82**(2), 205–229 (2009).
24. D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," presented at the 2015 *Int. Conf. for Learn. Represent.*, arXiv:1412.6980 (2015).

Mickael Aghajarian is a PhD candidate of electrical engineering at the University of Wyoming. His current research interests include computer vision and image processing, machine/deep learning algorithms, data science, robotics, and control.

John E. McInroy received his PhD from Rensselaer Polytechnic Institute, Troy, New York, in 1991. During his graduate study, he was a Rensselaer Fellow, Xerox Fellow, and NASA Graduate Student Researchers Fellow. Following graduation, he joined the University of Wyoming, where he is currently professor and head of Electrical and Computer Engineering. He has held visiting appointments at the Air Force Research Lab Space Vehicles Directorate, University of Pennsylvania, NASA Jet Propulsion Laboratory, NASA Marshall Space Flight Center, and Xerox Corporation.

Suresh Muknahallipatna received his BE degree in electrical engineering and master's degree of Engineering from the University of Bangalore, India, in 1988 and 1991, respectively. He completed his PhD at the University of Wyoming in 1995, with an emphasis on neural networks. He is currently working as a professor in the Department of ECE at the University of Wyoming. His current areas of expertise are GPGPU application in computer vision, radio propagation maps in mobile ad-hoc networks, and machine learning.