DeCapsGAN: generative adversarial capsule network for image denoising

Qiongshuai Lyu,[a,b] Min Guo,[a,*] Miao Ma,[a] and Richard Mankin[c]
[a]Shaanxi Normal University, Ministry of Education Key Laboratory of Modern Teaching Technology, School of Computer Science, Xi'an, China
[b]Pingdingshan University, School of Software, Pingdingshan, China
[c]Agricultural Research Service U.S. Department of Agriculture, Center for Medical, Agricultural, and Veterinary Entomology, Gainesville, Florida, United States

Abstract. A convolutional capsule network that characterizes small objects previously has shown remarkable performance in small object classification. We extend the idea of capsules to the image denoising task and combine it with the generative adversarial network to develop a generative adversarial capsule network (DeCapsGAN). Both the generator and discriminator adopt the capsule network architecture. The convolutional capsule network is used to capture richer image features. We introduce deconvolution into the generator and propose a convolutional–deconvolutional capsule block. Skip connections are beneficial to transfer image features to deeper networks. A pretrained residual network (ResNet) is implemented as a feature extractor that captures features from the denoised image and reference image to measure the difference in perceptual information in the feature space. The performance of the proposed model is evaluated on the image with synthetic noise (Gaussian noise and mixed Gaussian with impulse noise) and real noise. Extensive experiments show that our model achieves excellent denoising performance in terms of both visual quality and quantitative metrics. [DOI: 10.1117/1.JEI.30.3.033016]

Keywords: capsule network; image denoising; generative adversarial network; feature extractor.

Paper 200797 received Nov. 19, 2020; accepted for publication May 7, 2021; published online Jun. 2, 2021.

1 Introduction

Image denoising is still an open and challenging problem in low-level computer vision communities. Denoising methods originate from various disciplines, including filtering, Fourier transform, partial differential equations, Bayesian theory, and neural networks, and have applications as well to signal denoising in general. Most of the existing methods attempt to restore the latent noise-free image x from its corrupted image $y = x + v$, where v is additive noise. In general, image denoising is an ill-posed inverse problem due to its irreversibility. To alleviate this issue, model-based and discriminative learning-based methods are two main denoising methods. Model-based methods rely on the properties of the image itself, such as non-local self-similarity (NSS) and sparsity. Block-matching and 3D filtering (BM3D),[1] a highly practical engineering-level method, performs well in different noise levels. By exploiting low-rank minimization to represent the NSS of the image, weight nuclear norm minimization (WNNM)[2] achieves favorable denoising performance. Wen et al.[3] proposed the sparsifying transform learning and low-rank (STROLLR) method, which utilized sparsity and low-rank regularization to achieve promising performance. However, model-based methods involve some manually set hyperparameters, which increase the uncertainty of the denoising process. Moreover, sparse priors and NSS priors of natural images may not accurately characterize complex image textures and structures.

As an alternative, discriminative learning-based methods focus on fast inference and learning of latent image prior from training sets of noisy-clean image pairs. For example, using the powerful learning ability of convolutional neural network (CNN), denoising convolutional neural network (DnCNN)[4] achieves competitive denoising performance. Plötz and Stefan[5] introduced

*Address all correspondence to Min Guo, guomin@snnu.edu.cn

1017-9909/2021/$28.00

a k-nearest neighbor algorithm into a neural network and developed a neural nearest neighbors network (N^3-Net) to achieve different image restoration tasks. Zhang et al.[6] proposed a residual non-local attention network for image denoising and achieved comparable or better results against previous works. Although discriminative learning-based methods enjoy short denoising time and tend to deliver favorable denoising results, these models are purely end-to-end, and there is a lack of discrimination on the denoised images in the training process.

Generative adversarial networks (GANs)[7] can solve this problem through an adversarial process between a generative network and a discriminative network. Most GANs are stacked by standard CNNs, such as SRGAN,[8] Wasserstein GAN and VGG-Net (WGAN-VGG),[9] and self-attention generative adversarial network (SA-GAN).[10] CNNs are mainly composed of convolutional layers that are used to detect important features in image pixels. As the network depth grows, a CNN can learn features such as structures, textures, and edges of images. However, CNNs can neither model the hierarchical relationship of internal knowledge representation within the neural network nor understand how the built-in data distribution for one feature replicates across the entire image. This weakens the learning ability and understanding ability of the convolution operation. As a powerful alternative to CNNs, Sabour et al.[11] proposed a capsule network to compensate for the information lost by the CNN during training. The success of this approach raises a very natural question: Instead of using standard CNNs, can GANs be designed using capsule networks to improve their performance?

In this paper, we combine the capsule network with the GAN and propose a generative adversarial capsule network for image denoising (DeCapsGAN). We extend the idea of the capsule network to the generator and propose a convolutional–deconvolutional capsule block (CDCB). The dynamic routing algorithm implements the transfer of activation information between convolutional and deconvolutional capsule layers. Moreover, the loss function of our model consists of adversarial loss, feature loss, pixel loss, and multi-scale structural similarity (SSIM) loss. Experiments on synthetic noise (Gaussian noise and mixed Gaussian with impulse noise) and real noise show that the proposed model surpasses the comparison methods in denoising performance. Overall, the main contributions of this paper are summarized as follows.

(1) The architecture of the capsule network is extended to image denoising applications for the first time.
(2) We propose a generative adversarial capsule network, which combines the advantages of the capsule network and GAN. The idea of the capsule network is introduced into both the generator and discriminator; we also propose a CDCB in the generator.
(3) A pretrained residual network (ResNet) is incorporated into the proposed model to improve denoising performance.

The reminder of this paper is organized as follows. After reviewing relevant work in Sec. 2, we introduce the proposed model in Sec. 3, including the network architecture, CDCB, loss function, and discussion. In Sec. 4, we present extensive experiments and results, and we provide concluding remarks in Sec. 5.

2 Related Work

2.1 *Generative Adversarial Network*

GANs[7] have made significant advances to reduce the difficulty of modeling probability distributions of data. Initially, GANs were mainly used in the field of image generation. Inspired by GAN,[7] Mirza and Osindero[12] proposed a conditional version of GAN that generates data conditioned on class labels. Zhang et al.[10] proposed a SA-GAN to generate images with fine detail. This model introduces an attention mechanism into the structure of GANs, which alleviates the defect of simply using stacked convolution operations. To ensure the stability of the training process of large-scale GANs, Brock et al.[13] proposed a large-scale GAN to synthesize high-fidelity natural images. This model performs fine balance control between the authenticity and diversity of samples. In addition to image generation, GANs are also widely used in the field of image denoising. To eliminate unknown noise, Chen et al.[14] proposed a two-step blind image

denoising framework and used GAN to estimate the noise distribution in the image. This model overcomes the lack of paired training data in blind image denoising. Furthermore, increasing the number of training samples also helps to suppress noise. Tripathi et al.[15] used GANs to generate virtual samples from training samples. Yang et al.[9] proposed a GAN with Wasserstein distance and perceptual loss for low-dose CT image denoising. Although GANs have achieved great success in the field of computer vision, most of the generators and discriminators adopt standard CNN architectures. Note that, different from these models, the proposed model integrates the idea of the capsule network into GAN to deal with image application problems.

2.2 *Capsule Network*

Inspired by the working mechanism of human visual neurons, Hinton et al.[16] first proposed the concept of the capsule as the main computing unit of neural networks. The capsule performs complex internal calculations on the input data and encodes the results into an output vector with abundant information. In the neural network structure, each capsule layer is composed of several capsule units, and each capsule unit outputs a vector instead of a scalar activation. Sabour et al.[11] proposed a dynamic routing algorithm to realize the interaction between capsule layers. This algorithm is very effective for segmenting highly overlapping objects. Moreover, higher-level capsules can represent richer feature information than those at lower levels. Building on the work of Ref. 11, Hinton et al.[17] further proposed a new matrix capsules system that involved an iterative routing procedure based on an expectation–maximization algorithm that improves accuracy. In the algorithm, the correlation coefficient between capsules is represented by a matrix, which reduces the numbers of calculations and parameters. Extending the capsule network, Kosiorek et al.[18] proposed a staked capsule autoencoder network to achieve unsupervised image feature learning. This model uses the geometric relationship between parts to derive objects, which reduces calculation time compared with unsupervised classification. Motivated by Ref. 19, instead of simply using convolutional capsules, we introduce deconvolution operations and propose CDCBs to capture and reconstruct more complete context information.

3 Generative Adversarial Capsule Network

Several generative adversarial capsule networks[20,21] have emerged, but the idea of a capsule network is limited to the discriminator. How to extend the idea of a capsule network to a generator remains partly unresolved. Moreover, the original capsule network focuses on small-size image classification and digital recognition and rarely considers image reconstruction tasks. In the proposed model, we introduce the idea of a capsule network to constitute the generator (denoiser). Our complete model includes a denoiser R, a discriminator D, and a feature extractor (see Fig. 1). The denoiser R takes a noisy image y to generate the denoised image $\hat{x}$. The discriminator D is used to determine whether $\hat{x}$ is a noise-free image or not. Through the game between the denoiser R and the discriminator D, the denoiser R realizes the mapping from the noisy image to the noise-free image. The feature extractor captures the features of the reference image and denoised image and assists the model training by calculating the feature loss.

3.1 *Network Architecture*

Denoiser R consists of a feature extraction block (FEB), CDCBs, and a reconstruction block (RB). The input to our denoiser R is a noisy image y, and the output is a denoised image $\hat{x}$. The FEB R^{FEB} contains a convolutional layer and two convolutional capsule layers, which enable the denoiser not only to extract shallow image features but also to learn rich hierarchical relationships. Specifically, the convolutional layer is used to extract the shallow feature F_0 from the noisy image y

$$F_0 = R_{\mathrm{conv}}^{\mathrm{FEB}}(y), \tag{1}$$

where $R_{\mathrm{conv}}^{\mathrm{FEB}}$ denotes convolution operation. F_0 is sent to the convolutional capsule layer for learning hierarchical features:

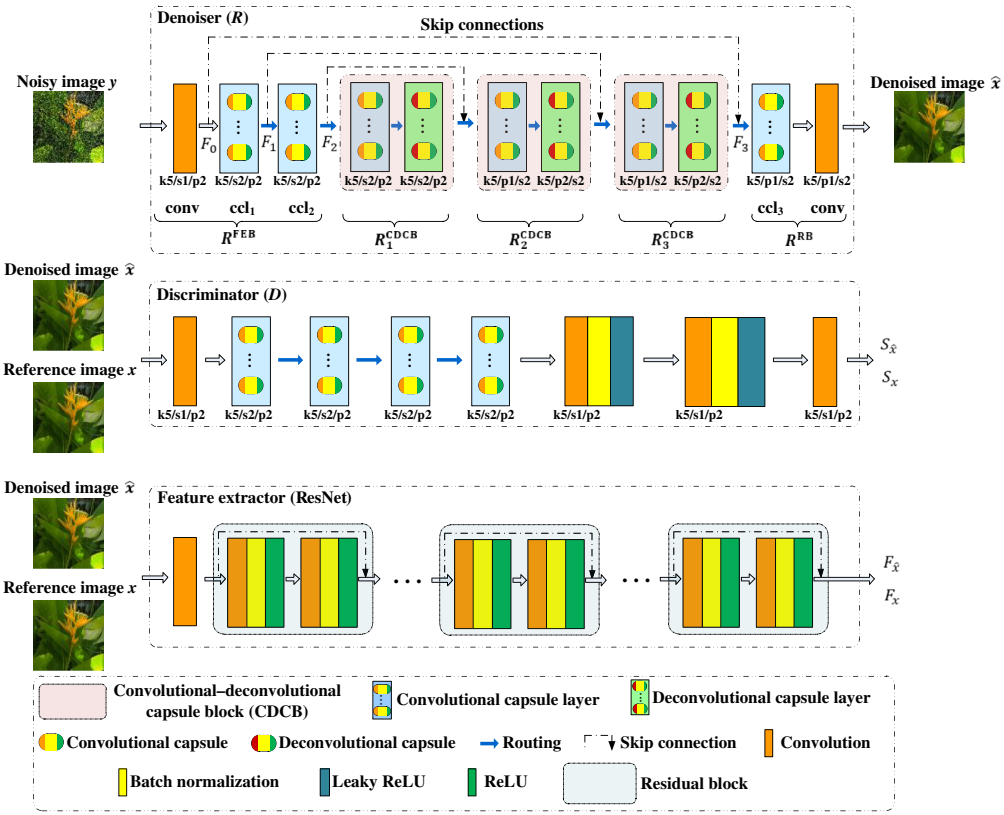

Fig. 1 Architecture of DeCapsGAN where **k** stands for the convolutional kernel size, **s** for stride (the number of pixels moved between operations on the input), and **p** for padding around the border of the input so that **k5**/**s1**/**p2**, for example, means kernel size = 5 × 5, stride = 1, and padding = 2. Note that, although only a color image example is given here, the proposed model is suitable for both gray image denoising and color image denoising.

$$F_2 = R_{\text{ccl2}}^{\text{FEB}}(F_1) = R_{\text{ccl2}}^{\text{FEB}}(R_{\text{ccl1}}^{\text{FEB}}(F_0)), \tag{2}$$

where $R_{\text{ccl1}}^{\text{FEB}}$ and $R_{\text{ccl2}}^{\text{FEB}}$ denote the convolutional capsule operation. F_2 is sent to the CDCBs R^{CDCB}. Three CDCBs are used to capture and reconstruct complete context information. At the same time, skip connections help to carry more feature information. The output F_i, $i \in \{1,2,3\}$ of the i'th CDCB is obtained by

$$F_3 = R_3^{\text{CDCB}}(R_2^{\text{CDCB}}(R_1^{\text{CDCB}}(F_2) + F_2) + F_1)) + F_0, \tag{3}$$

where R_i^{CDCB} denotes the i'th convolutional–deconvolutional capsule operation. Each CDCB contains a convolutional capsule layer and a deconvolutional capsule layer. Finally, instead of directly using a convolution operation, our denoiser uses the RB R^{RB} to generate a denoised image $\hat{x}$. The RB contains a convolutional capsule layer and a convolutional layer. The denoised image $\hat{x}$ is obtained by calculating

$$\hat{x} = R_{\text{conv}}^{\text{RB}}(R_{\text{ccl3}}^{\text{RB}}(F_3)), \tag{4}$$

where $R_{\text{ccl3}}^{\text{RB}}$ denotes the convolutional capsule operation and $R_{\text{conv}}^{\text{RB}}$ denotes the convolution operation. Additionally, unlike the pure convolutional capsule in the original capsule network, we add batch normalization (BN) and rectified linear unit (ReLU) activation function to the convolutional capsule and deconvolutional capsule. Adding BN can ease the internal covariate shift[22] and enjoys fast training. By incorporating convolution and deconvolution with ReLU, the denoiser can extract and recover image information. Skip connections help to convey abundant hierarchical information and image details and further ease the flow of information.

For discriminator D, the input is first a denoised image $\hat{x}$, and its corresponding output is the tensor $S_{\hat{x}}$. Then the input is a reference image x, and the output is tensor S_x. $S_{\hat{x}}$ and S_x are obtained by calculating

$$S_{\hat{x}} = D(\hat{x}), \quad S_x = D(x), \tag{5}$$

where D_{dis} denotes a composite operation, including convolution, convolutional capsule, and Conv + BN + Leaky ReLU. When training the discriminator D, it is assumed that the label corresponding to the reference image is tensor $\mathbf{1}$ and the label corresponding to the noisy image is tensor $\mathbf{0}$. The mean square error is used to measure the difference between the output of the discriminator D and the corresponding label. By training the denoiser R and the discriminator D, the denoiser R tries to generate a clean image, whereas the discriminator D tries to detect the noisy images. Competition in this game prompts both networks to improve their capabilities until the denoised image is indistinguishable from the clean image. A convolution operation is adopted for the first layer and the last layer. To enhance the ability to discriminate hierarchical relationships in the image, four convolutional capsule layers and two Conv + BN + Leaky ReLU layers form the middle layer.

For the feature extractor, the pretrained ResNet18[23] is used to capture features from the denoised image $\hat{x}$ and reference image x. The input is first a denoised image $\hat{x}$, and its corresponding output is the tensor $F_{\hat{x}}$. Then the input is a reference image x, and the output is tensor F_x. The extracted features $F_{\hat{x}}$ and F_x are expressed as

$$F_{\hat{x}} = \text{ResNet}(\hat{x}), \quad F_x = \text{ResNet}(x), \tag{6}$$

where $\text{ResNet}(.)$ denotes the feature extraction operation. The difference between F_x and $F_{\hat{x}}$ represents the feature loss, which is calculated by Eq. (10).

3.2 *Convolutional–Deconvolutional Capsule Block*

Capsules offer a good candidate model[11] for learning rich hierarchical relationships. The dynamic routing mechanism ensures that the output of a child capsule in the lower level is sent to the appropriate parent capsules in the higher level. In our model, we introduce deconvolutional capsules. The dynamic routing algorithm is not only applied between convolutional capsules but also extended between convolutional capsules and deconvolutional capsules. Specifically, in the CDCB, the convolutional capsule layer contains n convolutional capsules, and the deconvolutional capsule layer contains m deconvolutional capsules. The deconvolutional capsule layer receives a set of "prediction vectors" $\hat{u}_{j|i} = W_{ij}u_i$, $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$, from the convolutional capsule layer, where W_{ij} is a learnable weight matrix and u_i is the output of the i'th capsule type in the convolutional capsule layer. The total input of the j'th deconvolutional capsule type in the deconvolutional capsule layer is defined as $s_j = \sum_i r_{ij}\hat{u}_{j|i}$, where r_{ij} are routing coefficients that are updated by a "routing softmax:"

$$r_{ij} = \frac{e^{b_{i,j}}}{\sum_j e^{b_{i,j}}}, \tag{7}$$

where $b_{i,j}$ indicates the probability that the i'th convolutional capsule should be routed to the j'th deconvolutional capsule. Using a non-linear squashing function,[11] the output of a deconvolutional capsule vector v_j is obtained by

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2}\frac{s_j}{\|s_j\|}. \tag{8}$$

To describe the routing process between the convolutional capsule layer and the deconvolutional capsule layer in more detail, Fig. 2 shows an example of the routing of deconvolutional capsule vectors in the deconvolutional capsule layer from one convolutional capsule vector in the convolutional capsule layer.

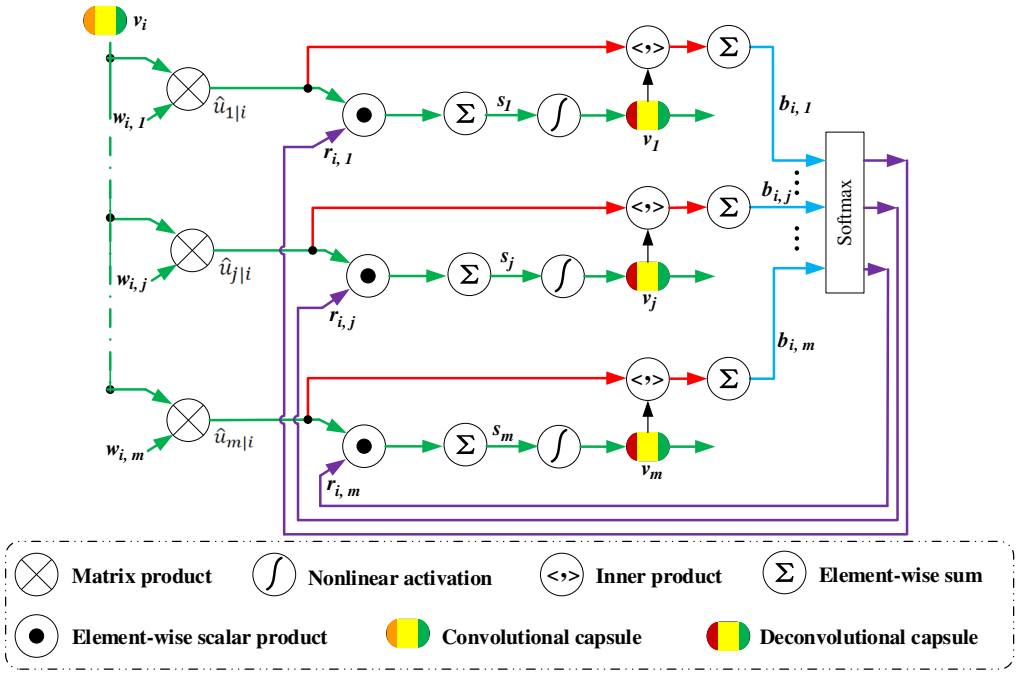

Fig. 2 Routing process between convolutional capsule layer and deconvolutional capsule layer. The green lines indicate the forward propagation of the data flow. The red lines indicate the flow for measuring the agreement between the convolutional capsule layer and deconvolutional capsule layer. The purple lines indicate the flow for updating coupling coefficients. The blue lines denote the updated log prior probabilities.

3.3 *Loss Function*

The design of the loss function is mainly based on the following considerations. First, the margin loss of the capsule network is used for multi-class classification and cannot be applied to the image denoising tasks. Second, the objective function of GAN focuses on the migration of the noise distribution to the image distribution, without considering the details of the generated image. Third, the loss function should ensure not only the denoising performance, that is, realize the statistical distribution of noise from strong to weak, but also the visual quality of the denoised image. Therefore, we propose a new loss function to train our model through supervised learning.

In our model, the denoiser R implements the mapping from noisy image to noise-free image, and the discriminator D is used to determine whether the denoised image is noise-free. The Nash equilibrium is achieved by alternating training between denoiser R and discriminator D. This training process solves the following min–max problem:

$$L_{\text{ad}} = \min_R \max_D L(D,R) = E_{x\sim p_{\text{data}(x)}}[\log\, D(x)] + E_{y\sim p_{\text{noise}(y)}}[\log(1 - D(R(y)))], \tag{9}$$

where L_{ad} denotes the adversarial loss. Although L_{ad} ensures the migration of image noise distribution from strong to weak, it often lacks high-frequency information. This leads to unsatisfying image content perceptually. To alleviate this issue, we introduce feature loss L_F, which is the Euclidean distance between a denoised image and a reference image in the feature space. In our implementation, the pretrained ResNet18[23] is adopted to extract features. Hence, L_F is defined as

$$\begin{aligned} L_F(R) &= E_{(y,x)}\left[\frac{1}{N_F}\|\text{ResNet}(R(y)) - \text{ResNet}(x)\|^2\right] \\ &= E_{(y,x)}\left[\frac{1}{N_F}\|\text{ResNet}(\hat{x}) - \text{ResNet}(x)\|^2\right] \\ &= E_{(y,x)}\left[\frac{1}{N_F}\|F_{\hat{x}} - F_x\|^2\right], \end{aligned} \tag{10}$$

where N_F denotes the number of pixels in the feature space. To further improve the image quality, the pixel-wise loss[8]

$$L_P(R) = E_{(y,x)}\left[\frac{1}{N_P}\|R(y) - x\|^2\right], \tag{11}$$

and the multi-scale SSIM loss[24]

$$L_{\text{ms}}(R) = E_{(y,x)}[\text{MS\_SSIM}(R(y), x)], \tag{12}$$

are also considered, where N_P denotes the number of pixels. Combining Eqs. (9)–(12), the loss function is expressed as

$$\min_R \max_D L_{\text{ad}}(D, R) + L_F(R) + L_P(R) + L_{\text{ms}}(R). \tag{13}$$

3.4 *Discussion*

In addition to our model, two other models are similar to our work: CapsGAN[20] and CapsuleGAN.[21] However, there are four main differences. First, the application areas of the models are different. CapsGAN and CapsuleGAN are applied to image generation, whereas the proposed model is applied to image denoising. Second, due to the different application areas, the loss function is different. The loss function of CapsGAN contains margin loss, reconstruction loss, and binary cross entropy. The loss function of CapsuleGAN adopts margin loss instead of binary cross-entropy loss. The combination of these loss functions cannot be applied to image denoising tasks. To eliminate noise and reconstruct the image, the loss function of our model considers adversarial loss, feature loss, pixel loss, and multi-scale SSIM loss. Third, the architecture of the capsule GAN is different. In CapsGAN and CapsuleGAN, the idea of the capsule network is only reflected in the discriminator, and the generator adopts the standard convolutional network; however, in our model, the idea of the capsule network is reflected not only in the discriminator but also in the generator (denoiser). Fourth, the generator is designed differently. In CapsGAN and CapsuleGAN, the structure of the generator is derived from the DCGAN guidelines,[25] whereas our generator uses a CDCB. In addition, CapsGAN and CapsuleGAN handle small-size images (such as 28×28, 32×32, and 64×64), whereas our model can operate on a larger image size (256×256).

4 Experiment

4.1 *Datasets and Training Setting*

We tested the proposed model on synthetic noise, such as additive white Gaussian noise (AWGN), mixed AWGN and salt-and-pepper impulse noise (SPIN), and real noise in real-world images. For AWGN, we set the range of the noise level as $\sigma \in [0,70]$ for training. For AWGN + SPIN, we set the range of the noise level as $\sigma \in [0,40]$, $s \in [0,40\%]$ for training. In the case of AWGN and AWGN + SPIN, two single models were trained for different noise levels of corruption. We selected the first 5000 images from the VOC 2007 dataset[26] (train/validation data) to train the proposed model. Four popular benchmarks were used for evaluation, including classic5,[4] Set12,[5] Kodak24,[6] and BSD68.[27,28] The classic5 and Set12 contain 5 and 12 classic test images, respectively. The Kodak24 contains 24 images of different life scenes. The BSD68 contains 68 images of different landscapes and people. For synthetic noise (AWGN and AWGN + SPIN), we verified the performance of the model on grayscale images and converted color images to grayscale images during training and testing. For real noisy image denoising, we used 100 images from the PolyU[29] dataset to train a real noise denoising model. Following Ref. 30, we used the mean image of each scene as the reference image to form noisy-clean image pairs. Due to the extreme shortage of real noisy image datasets, to better train the proposed model, we adopted data augmentation technology (e.g., flipping and rotating) to expand the training dataset to 1000. The PolyU dataset contains 40 different scenes captured by five cameras from the three

leading brands of cameras: Canon EOS (5D Mark II, 80D, 600D), Nikon (D800), and Sony (A7 II). The real-noisy cc[31] dataset was used to test the denoising effect of the real noisy images. This dataset contains 15 images and was captured by 3 different cameras (i.e., Canon 5D Mark III, Nikon D600, and Nikon D800) with different ISO settings (1600, 3200, and 6400). Peak signal-to-noise ratio (PSNR), SSIM, and denoising time (s) were used for quantitative analysis.

Each model was trained for 700 epochs with a mini-batch size of 8. We selected the model with the best performance for testing. All images were resized to 256×256. It takes about one and a half days to train a model. Both the denoiser and discriminator adopted Adam optimizer with an initial learning rate of 0.01, decay rate $\gamma = 0.1$. In denoiser R, the first convolutional capsule layer in the FEB contains one convolutional capsule type, and the second convolutional capsule layer contains four convolutional capsule types. In the first CDCB, the convolutional capsule layer contains eight convolutional capsule types, and the deconvolutional capsule layer contains eight deconvolutional capsule types. In the second CDCB, the convolutional capsule layer contains 16 convolutional capsule types, and the deconvolutional capsule layer contains 4 deconvolutional capsule types. In the third CDCB, the convolutional capsule layer contains eight convolutional capsule types, and the deconvolutional capsule layer contains four deconvolutional capsule types. There are three convolutional capsule types in the convolutional capsule layer of the reconstruction module. In discriminator D, the numbers of capsule types contained in the four convolutional capsule layers are 1, 2, 4, and 2, respectively. The structures of the denoiser and discriminator are listed in Table 1. ResNet18 used the default parameters. We implemented our model with Pytorch and ran it on a Dell T640 server with a Tesla K80 GPU. For AWGN and AWGN + SPIN, the parameters and model size of the two cases are the same due to the denoising of the grayscale image. The model size is 22.2 M, and the number of parameters is 5,835,905. For real noisy image denoising (color image), the model size is 22.3 M, and the number of parameters is 5,837,507. Note that here we only give the number of parameters and

Table 1 Structure of denoiser and discriminator.

Denoiser (R)	Module	Layer
	FEB	Convolutional layer
		Convolutional capsule layer (1 × capsule type)
		Convolutional capsule layer (4 × capsule type)
	3 × CDCB (3 × CDCB)	Convolutional capsule layer (8 × capsule type)
		Deconvolutional capsule layer (8 × capsule type)
		Convolutional capsule layer (16 × capsule type)
		Deconvolutional capsule layer (4× capsule type)
		Convolutional capsule layer (8 × capsule type)
		Deconvolutional capsule layer (4 × capsule type)
	RB	Convolutional capsule layer (3 × capsule type)
		Convolutional layer
Discriminator (D)	Layer	
	Convolutional layer	
	Convolutional capsule layer (1 × capsule type)	
	Convolutional capsule layer (2 × capsule type)	
	Convolutional capsule layer (4 × capsule type)	
	Convolutional capsule layer (2 × capsule type)	
	3 × Conv + BN + leaky ReLU	
	Convolutional layer	

the model size of the denoiser R because, in the test, we only need a trained denoiser and do not need discriminator D or feature extractor.

4.2 *Results*

4.2.1 *Additive white Gaussian noise removal*

We generated the noisy images by adding AWGN with different standard deviations ($\sigma = 20, 40, 60$) to the test images. From Fig. 3, one can see that AWGN blurs the image content and smooths the histogram of image data distribution. At the same time, two weak peaks appear at both ends of the noisy image histogram. To evaluate the denoising performance, the proposed model was compared with BM3D,[1] WNNM,[2] DnCNN,[4] N^3-Net,[5] xUnit,[32] WGAN-VGG,[9] and attention-guided denoising convolutional neural network (ADNet)[30] on the gray image. Table 2 lists the average PSNR/SSIM results on each benchmark dataset for different models. From Table 2, we can see that the proposed model surpasses all comparison models in the PSNR/SSIM metrics. As can be seen from the visual effects shown in Fig. 4, all models suppressed the noise effectively. The proposed model improved the restoration of the texture and details of the image. WNNM, DnCNN, N^3-Net, and xUnit tended to blur local details and produce over-smooth edges. Although preserving sharp edges, WGAN-VGG produced blur in the local detail region.

4.2.2 *Mixed Gaussian and impulse noise removal*

For testing the effects of mixed Gaussian-impulse noise, we focused on mixed AWGN and SPIN. The AWGN standard derivations σ were set to 35 and 25, and the SPIN ratios s were set to 20% and 35%. When an image is corrupted by AWGN + SPIN, some of its pixel values are [0, 255], and its other pixel values are either 0 or 255. From Fig. 5, one can see that AWGN + SPIN corrupts the image content seriously and over-smooths the histogram of image data distribution. At the same time, two very strong peaks appear at both ends of the noisy image histogram. To verify the effectiveness in eliminating mixed noise, the proposed model was compared with BM3D*,[1] ADNet,[30] weighted encoding with sparse nonlocal regularization (WESNR),[33] weighted joint sparse representation (WJSR),[34] Laplacian scale mixture modeling and nonlocal low-rank regularization (LSM-NLR),[35] M-Laplacian scale mixture modeling and nonlocal low-rank regularization (MLSM-NLR),[36] and CNN[37] on the gray image. To deal with AWGN + SPIN, the adaptive median filter was applied to remove the SPIN in the BM3D*. Table 3 lists the average PSNR/SSIM results on each benchmark dataset. The denoising performance of the proposed model surpasses all comparison models. The performance of LSM-NLR and MLSM-NLR is very close, the performance of WESNR and WJSR is very close, BM3D* is slightly better than WESNR and WJSR, and the performance of CNN is worse than our model. Figure 6 shows the visual effect of the denoised image. WESNR produces a serious blur. WJSR over-smooths the details of the image. LSM-NLR and MLSM-NLR cause some jitter on the local details of the image. CNN and DeCapsGAN achieve good visual effects, and our model refines the details.

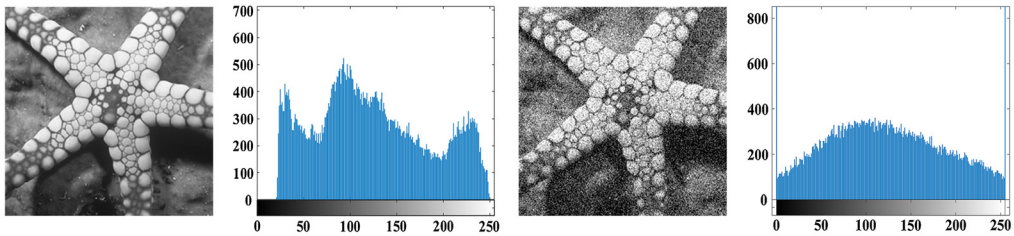

Fig. 3 Effect of AWGN on an image "starfish" from Set12[5] and its histogram: (a) noise-free image; (b) histogram of the noise-free image; (c) noisy image, noise level $\sigma = 40$; and (d) histogram of the noisy image.

Table 2 Average PSNR (dB)/SSIM results of AWGN removal on each benchmark dataset.

Dataset	Noise level	BM3D	WNNM	DnCNN	N^3-Net	xUnit	WGAN-VGG	ADNet	DeCapsGAN
Classic5	$\sigma = 20$	31.19/0.8467	30.65/0.8628	30.92/0.9302	31.00/0.8810	30.93/0.9300	34.22/0.9664	31.40/0.9285	**35.38/0.9725**
	$\sigma = 40$	27.96/0.7557	27.26/0.7567	27.63/0.8662	27.83/0.7963	27.67/0.8688	32.77/0.9525	28.36/0.8753	**34.68/0.9680**
	$\sigma = 60$	26.35/0.6938	25.52/0.6911	26.05/0.8961	26.15/0.7371	25.93/0.8234	28.29/0.8868	25.66/0.7961	**34.10/0.9633**
Set12	$\sigma = 20$	31.01/0.8719	30.89/0.8788	31.13/0.9412	31.58/0.8893	21.14/0.9414	33.91/0.9665	31.47/0.9411	**35.07/0.9769**
	$\sigma = 40$	27.65/0.7944	27.51/0.7974	27.79/0.8920	28.42/0.8271	27.81/0.8935	32.42/0.9529	28.34/0.8991	**34.31/0.9731**
	$\sigma = 60$	25.91/0.7418	25.65/0.7411	26.22/0.8459	26.63/0.7813	26.00/0.8552	27.60/0.8893	25.80/0.8320	**33.70/0.9684**
Kodak24	$\sigma = 20$	30.93/0.8409	30.57/0.8451	30.91/0.9246	31.03/0.8691	30.92/0.9253	33.74/0.9631	31.42/0.9261	**34.85/0.9722**
	$\sigma = 40$	27.85/0.7458	27.40/0.7470	27.71/0.8639	27.93/0.7836	27.79/0.8663	32.54/0.9491	28.44/0.8701	**34.08/0.9674**
	$\sigma = 60$	26.30/0.6888	25.73/0.6857	26.08/0.8017	26.28/0.7257	26.12/0.8225	27.85/0.8795	25.76/0.7884	**33.50/0.9622**
BSD68	$\sigma = 20$	29.62/0.8343	29.78/0.8428	30.17/0.9207	30.27/0.8683	30.19/0.9213	33.66/0.9654	30.16/0.9164	**34.44/0.9723**
	$\sigma = 40$	26.47/0.7246	26.62/0.7341	27.00/0.8530	27.16/0.7749	27.05/0.8549	32.20/0.9521	27.12/0.8491	**33.64/0.9672**
	$\sigma = 60$	24.97/0.6577	25.01/0.6656	25.33/0.7745	25.55/0.7118	25.41/0.8039	27.30/0.8790	24.81/0.7765	**33.06/0.9621**

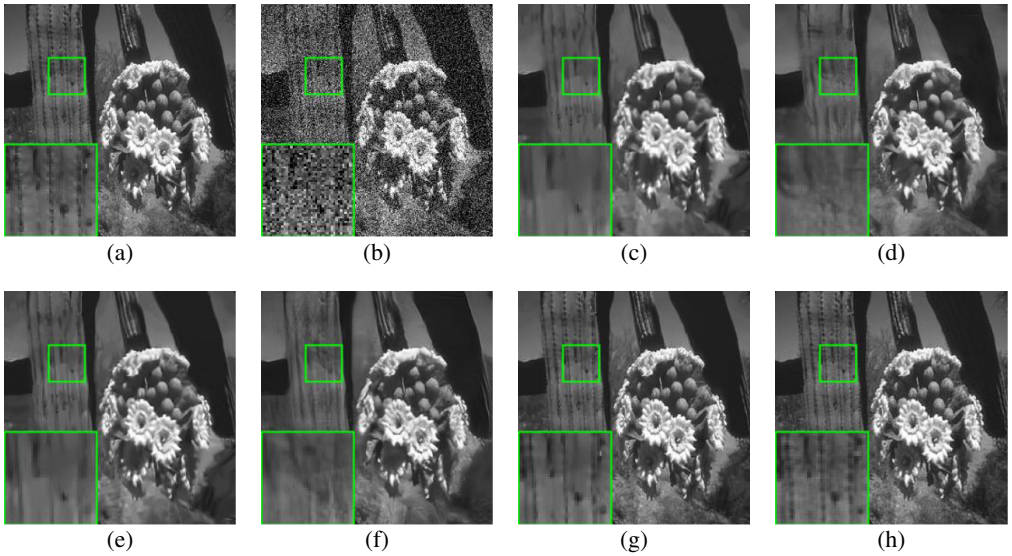

Fig. 4 Denoising results on image "test035" (from BSD68[28]) by different models ($\sigma = 40$): (a) ground truth; (b) noisy image (16.06/0.2693); (c) WNNM (25.92/0.7237); (d) DnCNN (26.07/0.8354); (e) N^3Net (26.36/0.7572); (f) xUnit (26.20/0.8420); (g) WGAN-VGG (32.58/0.9584); and (h) DeCapsGAN (32.90/0.9669).

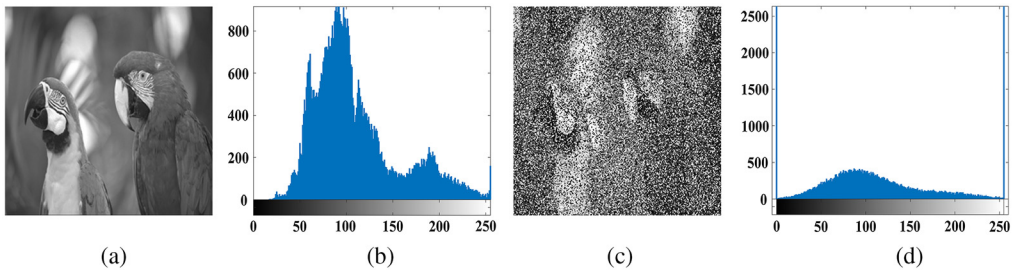

Fig. 5 Effect of AWGN + SPIN on an image "kodak23" from Kodak24[6] and its histogram: (a) noise-free image; (b) histogram of the noise-free image; (c) noisy image, noise level $\sigma = 25$, $s = 35\%$; and (d) histogram of the noisy image.

4.2.3 *Real noisy image denoising*

In real scenarios, since we can only obtain the noisy image, the distribution of real noise is unknown and complex, so we cannot draw the histogram of its data distribution. To better visualize the latent noise hidden in the complex background, Fig. 7 shows a real noisy image and its thermodynamic image. One can see that the noise in Fig. 7(d) is more obvious than that in Fig. 7(b). For real noisy image denoising, the proposed model was compared with CBM3D,[38] WNNM,[2] NI,[39] cascade of shrinkage field (CSF),[40] DnCNN,[4] target image denoising (TID),[41] and ADNet.[30] The results on PSNR/SSIM of different methods are listed in Table 4. (The results of ADNet are copied from the original paper.[30]) One can see that on 6 out of the 15 images, the proposed model achieves the best PSNR. The average PSNR of our model is 0.41 dB higher than the second-best model WNNM and has much higher PSNR gains over the competing models. In particular, our model occupies an absolute advantage in SSIM metrics and achieves the best SSIM on 13 of the 15 images. Figure 8 shows the visual effect of the denoised image for different methods. It can be seen that CBM3D, NI, and DnCNN do not eliminate the noise well and produce some artifacts, whereas WNNM and ADNet smooth the image. CSF and our model produce a visually pleasing result.

Denoising time is another important evaluation index for model assessment. Figure 9 shows the denoising time of all of the comparison models. For different denoising models, BM3D, WNNM, WESNR, WJSR, LSM-NLR, MLSM-NLR, CBM3D, TID, and CSF are executed

Table 3 Average PSNR (dB)/SSIM results of AWGN + SPIN removal on each benchmark dataset.

Dataset	Noise level		BM3D*	WESNR	WJSR	LSM-NLR	CNN	MLSM-NLR	ADNet	DeCapsGAN
Classic5	$\sigma = 35$	$s = 20\%$	26.34/0.6841	25.68/0.6962	25.29/0.6950	27.09/0.7524	27.13/0.8093	27.05/0.7490	27.41/0.8472	**28.88/0.8951**
		$s = 35\%$	25.25/0.6321	24.50/0.6634	24.76/0.6652	26.64/0.7328	26.96/0.7864	26.70/0.7311	26.79/0.8299	**28.14/0.8799**
	$\sigma = 25$	$s = 20\%$	27.75/0.7456	27.37/0.7523	27.19/0.7654	28.74/0.8040	29.02/0.8232	28.80/0.8047	28.90/0.8857	**29.98/0.9168**
		$s = 35\%$	26.70/0.7063	26.85/0.7427	26.52/0.7300	27.92/0.7893	28.87/0.8026	28.17/0.7923	28.15/0.8676	**29.05/0.9008**
Set12	$\sigma = 35$	$s = 20\%$	26.17/0.7375	24.45/0.7299	24.67/0.7301	27.00/0.7794	27.54/0.8641	27.08/0.7807	27.40/0.8785	**28.27/0.9071**
		$s = 35\%$	25.12/0.6838	23.08/0.6912	24.17/0.7108	26.57/0.7761	27.21/0.8491	26.80/0.7787	26.77/0.8654	**27.45/0.8927**
	$\sigma = 25$	$s = 20\%$	27.58/0.7939	26.65/0.7935	26.74/0.7941	28.77/0.8344	28.90/0.8519	28.91/0.8347	28.92/0.9073	**29.38/0.9273**
		$s = 35\%$	26.35/0.7520	25.67/0.7789	26.06/0.7706	27.61/0.8149	28.03/0.8171	28.01/0.8189	28.13/0.8940	**28.39/0.9129**
Kodak24	$\sigma = 35$	$s = 20\%$	26.40/0.6829	24.99/0.6843	23.70/0.6752	27.14/0.7369	27.86/0.8005	27.14/0.7378	27.62/0.8506	**28.55/0.8937**
		$s = 35\%$	25.30/0.6285	23.91/0.6538	24.17/0.6483	26.69/0.7238	27.21/0.7773	26.74/0.7214	27.04/0.8373	**27.78/0.8771**
	$\sigma = 25$	$s = 20\%$	27.73/0.7432	26.87/0.7406	26.60/0.7451	28.64/0.7887	28.99/0.8674	28.68/0.7886	28.98/0.8839	**29.53/0.9145**
		$s = 35\%$	26.63/0.8026	26.03/0.7273	25.98/0.7148	27.74/0.7736	28.33/0.8432	28.04/0.7771	28.21/0.8677	**28.60/0.8980**
BSD68	$\sigma = 35$	$s = 20\%$	25.33/0.6689	24.08/0.6664	23.67/0.6397	26.38/0.7298	27.57/0.8218	26.41/0.7292	26.47/0.8307	**27.98/0.8875**
		$s = 35\%$	24.41/0.6254	22.96/0.6364	23.20/0.6090	25.93/0.7091	26.74/0.8063	25.99/0.7068	25.93/0.8149	**27.17/0.8685**
	$\sigma = 25$	$s = 20\%$	26.56/0.7292	26.00/0.7209	25.57/0.7184	27.83/0.7819	28.33/0.8462	27.88/0.7803	27.84/0.8709	**29.01/0.9105**
		$s = 35\%$	25.65/0.6948	25.28/0.7077	25.04/0.6869	26.98/0.7667	27.68/0.8249	27.23/0.7674	27.10/0.8517	**28.04/0.8915**

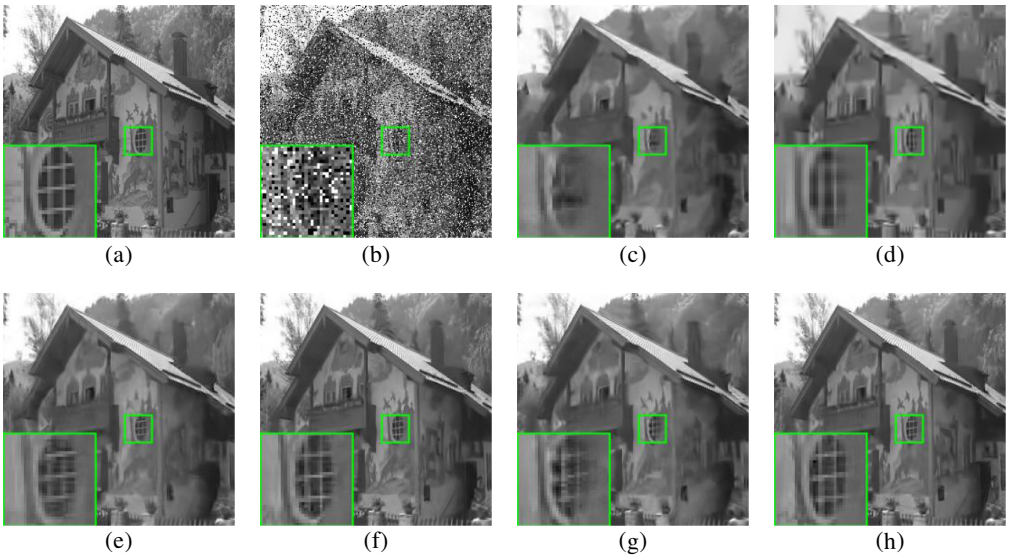

Fig. 6 Denoising images of "kodak24" (from Kodak24[6]) by different models ($\sigma = 25$, $s = 20\%$): (a) ground truth; (b) noisy image (11.80/0.1233); (c)WESNR (25.54/0.6714); (d)WJSR (24.99/0.6920); (e) LSM-NLR (27.04/0.7461); (f) CNN (27.39/0.8219); (g) MLSM-NLR (27.11/0.7418); and (h) DeCapsGAN (27.81/0.8981).

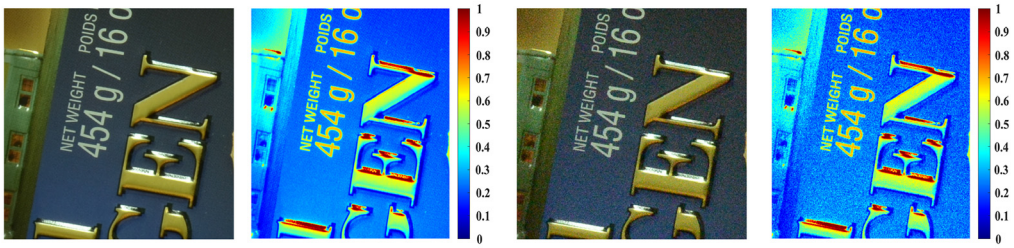

Fig. 7 A real noisy image "d800_iso6400_1" from cc[31] and its thermodynamic image: (a) ground truth; (b) thermodynamic image of ground truth; (c) real noisy image; and (d) thermodynamic image of the noisy image. The scene was shot 500 times with the same camera and camera setting. The mean image of the 500 shots is roughly taken as the "ground truth," with which the PSNR/SSIM is computed.

in the MATLAB platform. DnCNN, N^3-Net, xUnit, WGAN-VGG, and ADNet are implemented under the PyTorch framework. CNN is run under the Tensorflow platform. From Fig. 9, we can see that the denoising time of CNN, DnCNN, xUnit, WGAN-VGG, and ADNet is generally very short. The reason is that these models make full use of the parallel computing power of the GPU in the denoising stage. By contrast, the denoising time of WNNM, WESNR, WJSR, LSM-NLR, MLSM-NLR, and TID is relatively long. These models are designed to solve a complex and non-convex optimization problem, and it is difficult to achieve high performance without sacrificing iteration time and computational efficiency. Moreover, we can further observe that the proposed model achieves a very appealing denoising time; BM3D and CBM3D achieve promising computational efficiency, even though it runs on a CPU.

4.3 *Ablation Studies*

4.3.1 *FEB and CDCB*

To further explore the effect of the capsule network on improving model performance, we seek to investigate FEB and CDCB through some ablation studies. We mainly consider the following

Table 4 Average PSNR (dB)/SSIM results of different methods on real noisy images.

Camera settings	CBM3D	WNNM	NI	CSF	DnCNN	TID	ADNet	DeCapsGAN
Canon 5D Mark III ISO = 3200	**39.76**/0.9778	37.51/0.9673	37.68/0.9600	35.68/0.9434	37.26/0.9389	37.22/0.9515	35.96/—	35.74/**0.9858**
	36.40/0.9552	33.86/0.9210	34.87/0.9308	34.03/9.9011	34.13/0.8989	34.54/0.9041	36.11/—	**37.02/0.9821**
	36.37/0.9660	31.43/0.9110	34.77/0.9463	32.63/0.9037	34.09/0.9182	34.25/0.9354	34.49/—	**36.47/0.9854**
Nikon D600 ISO = 3200	34.18/0.9330	33.46/0.9281	34.12/0.9413	31.78/0.8792	33.62/0.9123	32.99/0.8913	33.94/—	**35.71/0.9792**
	35.07/0.9168	**36.09/**0.9432	35.36/0.9251	35.16/0.9261	34.48/0.8932	34.20/0.8605	34.33/—	35.83/**0.9719**
	37.13/0.9313	39.86/0.9737	38.68/0.9481	**39.89/0.9763**	35.41/0.8708	35.58/0.8632	38.87/—	36.93/0.9638
Nikon D800 ISO = 1600	36.81/0.9339	36.35/0.9471	37.34/0.9506	34.84/0.9148	35.79/0.9060	34.94/0.8832	37.61/—	**38.41/0.9854**
	37.76/0.9383	**39.99/**0.9748	38.57/0.9615	38.42/0.9674	36.08/0.8943	35.15/0.8772	38.24/—	39.14/**0.9871**
	37.51/0.9277	37.15/0.9311	**37.87/**0.9229	35.79/0.9035	35.48/0.8735	35.26/0.8451	36.89/—	37.19/**0.9742**
Nikon D800 ISO = 3200	35.05/0.8866	**38.60/**0.9656	36.95/0.9101	38.36/0.9654	34.08/0.8463	33.70/0.8356	37.20/—	37.68/**0.9756**
	34.07/0.8926	36.04/0.9416	35.09/0.9194	35.53/0.9354	33.07/0.8755	31.04/0.7761	35.67/—	**36.85/0.9786**
	34.42/0.8430	39.73/0.9664	36.91/0.9001	**40.05/0.9712**	33.31/0.7204	33.07/0.7882	38.09/—	36.85/0.9666
Nikon D800 ISO = 6400	31.13/0.7952	33.29/0.9188	31.28/0.7781	**34.08**/0.9259	29.83/0.7847	29.40/0.7118	32.24/—	33.32/**0.9496**
	31.22/0.8613	31.16/0.9050	31.38/0.8649	32.13/0.9172	30.55/0.8259	29.86/0.7995	**32.59/—**	31.81/**0.9551**
	30.97/0.8363	31.98/0.8818	31.40/0.8295	31.52/0.8494	30.09/0.7939	29.21/0.7717	33.14/—	**33.67/0.9549**
Average PSNR	35.19/0.9063	35.77/0.9381	35.49/0.9126	35.33/0.9250	33.86/0.8635	33.36/0.8463	35.69/—	**36.18/0.9733**

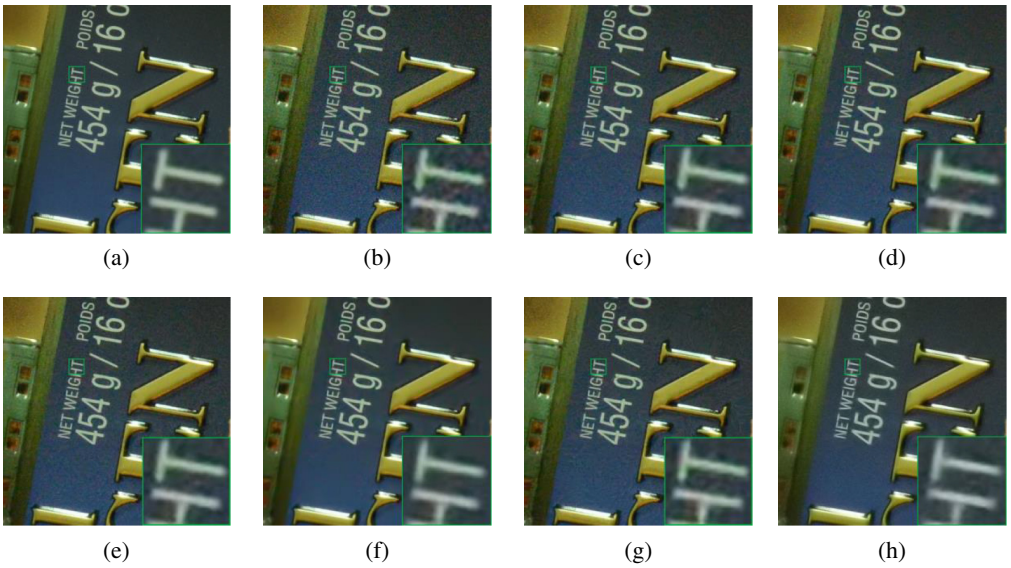

Fig. 8 Denoising images of "d800 _iso6400_1_real" (from cc[31]) by different models: (a) ground truth; (b) noisy image (29.63/0.7107); (c) CBM3D (31.13/0.7952); (d) WNNM (33.29/0.9188); (e) NI (31.28/0.7781); (f) CSF (34.08/0.9259); (g) DnCNN (29.83/0.7847); and (h) DeCapsGAN (33.32/0.9496).

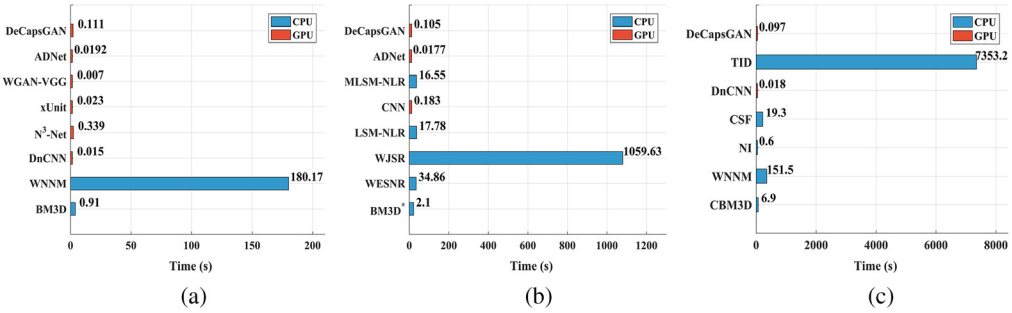

Fig. 9 Comparison of denoising time: (a) average denoising time on Set12[5] ($\sigma = 40$); (b) average denoising time on BSD68[28] ($\sigma = 35$, $s = 35\%$); and (c) average denoising time on cc.[31]

two cases: the validity of (1) FEB and (2) CDCB. For case 1, we only replace the two convolutional capsule layers in FEB with the standard convolutional operation without changing other training configurations (denoted as DeCapsGAN*). For case 2, we replace the convolutional capsule and deconvolution capsule layers in CDCB with standard convolution and deconvolution operations without changing other training configurations (denoted as DeCapsGAN$^{\#}$). As can be observed from Table 5, the capsule network has an influence on the performance of the model. Removing the capsule operation degrades the model. On the one hand, after replacing the capsule network in FEB, the performance of the model decreases significantly. This indicates

Table 5 Average PSNR (dB)/SSIM results on four benchmark datasets corrupted by AWGN + SPIN ($\sigma = 35$, $s = 20\%$).

Model	Classic5	Set12	Kodak24	BSD68
DeCapsGAN*	28.37/0.8829	27.66/0.8936	28.01/0.8782	27.47/0.8719
DeCapsGAN#	28.79/0.8840	**28.31**/0.8912	28.47/0.8771	27.96/0.8779
DeCapsGAN	**28.88/0.8951**	28.27/**0.9071**	**28.55/0.8937**	**27.98/0.8875**

that the capsule operation does extract more abundant features than the standard convolutional operation. On the other hand, CDCB can be used for image denoising, and its advantages can be further exploited when used in conjunction with an FEB with the capsule operation. The main reason is that, although standard convolution operations and capsule operations can both extract image features, capsules are better at extracting the hierarchical relationship of lower features, which is helpful to give full play to the performance of CDCB.

4.3.2 *ResNet feature extractor*

Due to the strong network expression capability of ResNet, how well it performs image feature extraction may arouse concerns. Figure 10 visualizes the feature maps of different layers of ResNet. From Fig. 10, one can see that each feature map contains abundant feature information of the extracted image, e.g., edges, textures, and structures. Therefore, ResNet as a feature extractor offers a better way to capture image features.

To verify the contribution of the ResNet feature extractor to the model, we only remove ResNet from our proposed model without changing other training configurations (denoted as DeCapsGAN^{-R}). Figure 11 shows the averaged PSNR/SSIM results on four benchmark datasets. From Fig. 11, we can see that introducing ResNet is beneficial to improving model performance. The main reason is that ResNet has a very powerful presentation ability, which comes from its previous training on large-scale natural image datasets. By introducing ResNet, DeCapGAN transfers the perceptual information of feature space that is embedded in ResNet to the image quality evaluation. DeCapsGAN^{-R} achieves the data distribution from the noisy image domain to

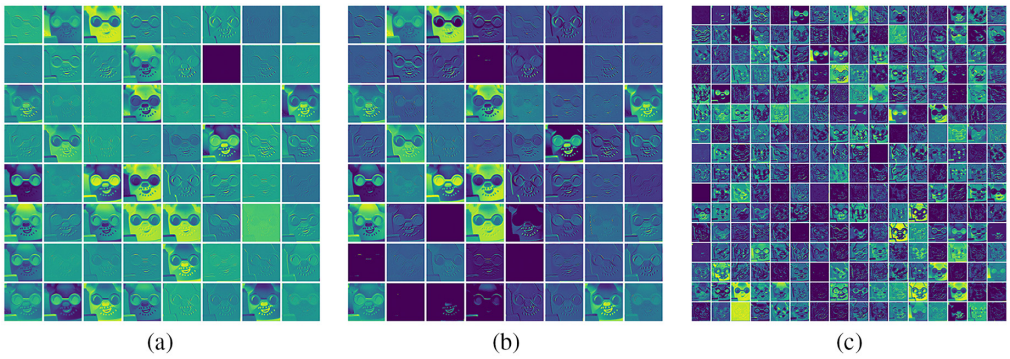

Fig. 10 The feature maps of different layers of ResNet on image "d800_iso3200_1_mean" from cc.[31] (a) Some feature maps generated by the first Conv, (b) some feature maps generated by the first Conv+BN+ReLu, and (c) some feature maps generated by the third residual group.

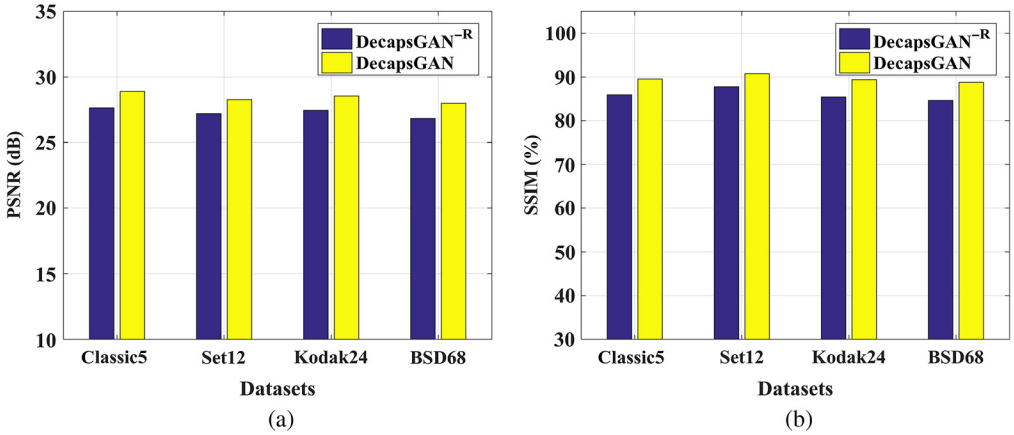

Fig. 11 Average (a) PSNR (dB) and (b) SSIM (%) results on four benchmark datasets corrupted by AWGN + SPIN ($\sigma = 35$, $s = 20\%$).

the free-noisy image domain, which effectively suppresses noise, but DeCapsGAN can further improve PSNR/SSIM metrics by introducing ResNet.

5 Conclusions

In this paper, we propose a generative adversarial capsule network for image denoising. The proposed model consists of a denoiser, discriminator, and feature extractor. To make full use of the advantages of the capsule network in learning hierarchical relationships, the capsule network is used not only in the discriminator but also in the generator. Capsule networks are used to capture abundant image features. Moreover, we also introduce deconvolutional capsules and elaborate the efficient CDCB. CDCB can further exploit its advantages when used in conjunction with FEB. To improve the denoising performance of the model, we use ResNet to extract the perceptual information of features in both the denoised and reference images in the feature space for image quality evaluation. Extensive experimental results show that the proposed model not only achieves very good performance in eliminating synthetic noise but also has certain advantages in eliminating real noise.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61877038) and the Fundamental Research Funds for the Central Universities (No. GK202105006). We appreciated the help of the Henan Intelligent Traffic Safety Engineering Technology Research Center and Henan Multimodal Data Intelligent Traffic Safety Engineering Technology Research Center.

References

1. K. Dabov et al., "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.* **16**(8), 2080–2095 (2007).
2. S. Gu et al., "Weighted nuclear norm minimization and its applications to low level vision," *Int. J. Comput. Vision* **121**(2), 183–208 (2017).
3. B. Wen, Y. Li, and Y. Bresler, "Image recovery via transform learning and low-rank modeling: the power of complementary regularizers," *IEEE Trans. Image Process.* **29**, 5310–5323 (2020).
4. 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).
5. T. Plötz and R. Stefan, "Neural nearest neighbors networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, pp. 1087–1098 (2018).
6. Y. Zhang et al., "Residual non-local attention networks for image restoration," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, New Orleans (2019).
7. I. Goodfellow et al., "Generative adversarial nets," in *Proc. 28th Annu. Conf. Neural Inf. Proc. Syst. (NeurIPS)*, pp. 2672–2680 (2014).
8. C. Ledig et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 105–114 (2017).
9. Q. Yang et al., "Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss," *IEEE Trans. Med. Imaging* **37**(6), 1348–1357 (2018).
10. H. Zhang et al., "Self-attention generative adversarial networks," *Comput. Res. Repository (CoRR)*, arXiv:1805.08318 (2018).
11. S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, pp. 3859–3869 (2017).
12. M. Mirza and S. Osindero, "Conditional generative adversarial nets," *Comput. Res. Repository (CoRR)*, arXiv:1411.1784 (2014).
13. A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Int. Conf. Learn. Represent. (ICLR)*, New Orleans (2019).

14. J. Chen et al. "Image blind denoising with generative adversarial network based noise modeling," in *Proc. 2018 IEEE/CVF Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 3155–3164 (2018).
15. S. Tripathi, Z. C. Lipton, and T. Q. Nguyen, "Correction by projection: denoising images with generative adversarial networks," *Comput. Res. Repository (CoRR)*, arXiv:1803.04477 (2018).
16. G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," *Lect. Notes Comput. Sci.* **6791**, 44–51 (2011).
17. G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proc. Int. Conf. Learn. Represent. (ICLR)* (2018).
18. A. Kosiorek et al., "Stacked capsule autoencoders," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, pp. 15512–15522 (2019).
19. R. LaLonde and U. Bagci, "Capsules for object segmentation," *Comput. Res. Repository (CoRR)*, arXiv: 1804.04241v1 (2018).
20. R. Saqur and S. Vivona, "CapsGAN: using dynamic routing for generative adversarial networks," in *Proc. Adv. Comput. Vision, CVC2019, Adv. Intell. Syst. and Comput.*, Vol. 944, pp. 511–525 (2020).
21. A. Jaiswal et al., "CapsuleGAN: generative adversarial capsule network," *Comput. Res. Repository (CoRR)*, arXiv: 1802.06167 (2018).
22. S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn. (ICML)*, Vol. 37, pp. 448–456 (2015).
23. K. He et al., "Deep residual learning for image recognition," in *Proc. 2016 IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 770–778 (2016).
24. C. You et al., "Structurally-sensitive multi-scale deep neural network for low-dose CT denoising," *IEEE Access* **6**, 41839–41855 (2018).
25. A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *Comput. Res. Repository (CoRR)*, arXiv: 1511.06434 (2015).
26. R. J. Chet, "Pascal VOC dataset mirror (VOC2007)," https://pjreddie.com/projects/pascal-voc-dataset-mirror/ (2007).
27. Y. Tai et al., "MemNet: a persistent memory network for image restoration," in *Proc. 2017 IEEE Int. Conf. Comput. Vision (ICCV)*, pp. 4549–4557 (2017).
28. P. Arbelaez et al., "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(5), 898–916 (2011).
29. J. Xu et al., "Real-world noisy image denoising: a new benchmark," *Comput. Res. Repository (CoRR)*, arXiv:1804.02603 (2018).
30. C. Tian et al., "Attention-guided CNN for image denoising," *Neural Networks* **124**, 117–129 (2020).
31. S. Nam et al., "A holistic approach to cross-channel image noise modeling and its application to image denoising," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 1683–1691 (2016).
32. I. Kligvasser, T. R. Shaham, and T. Michaeli, "xUnit: learning a spatial activation function for efficient image restoration," in *Proc. 2018 IEEE/CVF Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 2433–2442 (2018).
33. J. Jiang, L. Zhang, and J. Yang, "Mixed noise removal by weighted encoding with sparse nonlocal regularization," *IEEE Trans. Image Process.* **23**(6), 2651–2662 (2014).
34. L. Liu et al., "Weighted joint sparse representation for removing mixed noise in image," *IEEE Trans Cybern.* **47**(3), 600–611 (2017).
35. T. Huang et al., "Mixed noise removal via Laplacian scale mixture modeling and nonlocal low-rank approximation," *IEEE Trans. Image Process.* **26**(7), 3171–3186 (2017).
36. M. T. Islam et al., "A variational step for reduction of mixed Gaussian-impulse noise from images," in *Proc. 2018 10th Int. Conf. Electr. and Comput. Eng. (ICECE)*, pp. 97–100 (2018).
37. M. T. Islam et al., "Mixed Gaussian-impulse noise reduction from images using convolutional neural network," *Signal Process.: Image Commun.* **68**, 26–41 (2018).

38. K. Dabov et al., "Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, pp. 313–316 (2007).
39. Neatlab ABSoft, "Neat image," https://ni.neatvideo.com/home (2017).
40. U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *Proc. 2014 IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, pp. 2774–2781 (2014).
41. E. Luo, S. H. Chan, and T. Q. Nguyen, "Adaptive image denoising by targeted databases," *IEEE Trans. Image Process.* **24**(7), 2167–2181 (2015).

Qiongshuai Lyu received his BS degree from the School of Computer at Henan University of Urban Construction and his MS degree from the School of Information Engineering at Zhengzhou University in 2007 and 2011, respectively. Now, he is a PhD candidate at the School of Computer Science, Shaanxi Normal University, Xi'an, China. His main research interests include image restoration, deep learning, and sparse signal representation models.

Min Guo received her MS degree and PhD in electronic technology application from the College of Physics and Information Technology, Shaanxi Normal University, Xi'an, China, in 1990 and 2003, respectively. She worked at the Postdoctoral Research Station of Northwestern Polytechnical University, Xi'an, China, in 2007. Since then, she has been a professor at the School of Computer Science, Shaanxi Normal University, Xi'an, China. Her research interests include pattern recognition, computer vision, and signal and image processing.

Miao Ma received her MS degree from the College of Computer Science and Technology at Xi'an University of Science and Technology and her PhD from the School of Computer Science, Northwestern Polytechnical University, Xi'an, China, in 2002 and 2005, respectively. She is now a professor at the School of Computer Science of Shaanxi Normal University. Her main research interests include image processing, machine learning, and sparse signal representation.

Richard Mankin received his MS degree and PhD from the University of Florida, Florida, USA, in 1976 and 1979, respectively. Now he works at the Agricultural Research Service (ARS), United States Department of Agriculture, Center for Medical, Agricultural, and Veterinary Entomology, Gainesville, FL, USA. His research includes digital signal processing, biological control, and acoustic detection of hidden insect infestations.