Applied Intelligence
https://doi.org/10.1007/s10489-019-01623-0

The classification and denoising of image noise based on deep neural networks

Fan Liu[1] · **Qingzeng Song**[1] · **Guanghao Jin**[1]

Abstract

Currently, image denoising is a challenge in many applications of computer vision. The existing denoising methods depend on the information of noise types or levels, which are generally classified by experts. These methods have not applied computational methods to pre-classify the image noise types. Furthermore, some methods assume that the noise type of the image is a certain one like Gaussian noise, which limits the ability of the denoising in real applications. Different from the existing methods, this paper introduces a new method that can classify and denoise not only a certain type noise but also mixed types of noises for real demand. Our method utilizes two types of deep learning networks. One is used to classify the noise type of the images and the other one performs denoising based on the classification result of the first one. Our framework can automatically denoise single or mixed types of noises with these efforts. Our experimental results show that our classification network achieves higher accuracy, and our denoising network can ensure higher PSNR and SSIM values than the existing methods.

Keywords Image denoising · Deep learning · Classification · PSNR · SSIM

1 Introduction

Image denoising is a process of removing interference from images and restoring clean images [1]. It is one of the most basic operations in image processing, video processing and low-level computer vision. In many real applications, the obtained images may contain noise, which means there is useless information. The noise is popular in many applications such as medical CT images, satellite remote sensing images and so on. The noise in these applications may reduce the quality of the image and make it impossible to output the correct judgment. Thus the noise needs to be preprocessed before the following processes.

Since the concept of deep learning [2] was introduced in 2006, it has made outstanding achievements in classification and related fields. One of the deep learning methods that is called Convolutional Neural Network, (CNNS) achieved a remarkable result at the 2012 ImageNet Large-Scale Visual Identity Competition (ILSVRC) [3]. It has attracted many scholars to invest in the relevant researches including image denoising, which has achieved good results by the deep learning methods. Cagdas designed a Complete Convolutional Network (CCN) to map noise-perfused weighted images from clean images and their subtraction (residual), thereby it can improve the image quality of ASL [4]. G. Chierchia used a residual learning strategy to design a convolutional neural network to detect the noise and then subtracted the noisy component to obtain a high quality SAR images [5]. Non-local self-similar models (NNS) [6, 7], gradient models [8, 9], sparse models [10, 11], convolutional neural network model [12–14] are also proposed to perform denoising. Although these methods have achieved good denoising effects, they also retain many defects like the type of noise should be manually decided by experts or assume there is certain type of noise, which leads to slow processing and poor denoising effect in real applications.

Compared with the existing methods, the innovation of our method is shown in Fig. 1. The existing methods manually recognize the noise type by experts, or assume the type is a certain one like Gaussian noise and then choose the corresponding methods to denoise the image. Actually,

✉ Guanghao Jin
jingh_research@163.com

Fan Liu
bluesliuf@163.com

[1] Tianjin Polytechnic University, Tianjin, 300387, China

Published online: 02 March 2020

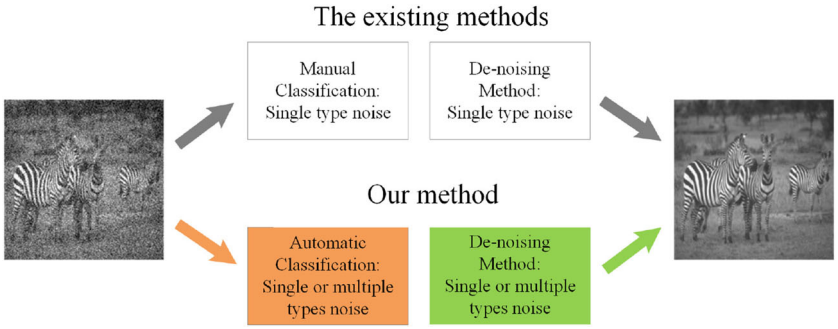

Fig. 1 The difference between the existing methods and our method

in the real applications, there may be various noise types in an image that are not only Gaussian noise, but also salt and pepper, Poisson, speckle noise [15], etc., or a mixture of these.

In this paper, our framework does not assume the image noise type a certain one like Gaussian and then denoises it. Our framework mainly consists of four processes: data preprocessing, noise classification, denoising and testing. First of all, our framework generates noisy images by adding single, two or multiple types of noises to each clean image in preprocessing to build the datasets. Secondly, these noisy images are utilized to train and evaluate some classification networks. Then our framework select the best network to classify the noise type of image for the following processes. Thirdly, our framework trained some models of a denoising network, so that each of them can denoise corresponding type of noise. Finally, it uses denoising models to remove the noise based on the classification result. The experimental results show that both our classification and denoising models have achieved better performance among the compared methods.

The rest of this article is organized as follows. Section 2 introduces related work including the development and application of denoising methods. Section 3 introduces data preprocessing and the structure of our method. Section 4 explains how to train the classification and denoising models on the datasets and analyzes the experimental results. Finally, Section 5 discusses the conclusions and future work.

2 Related work

Some existing methods perform noise identification or estimation based on statistical analysis of natural images. Gupta [16] devised an efficient blind image noise estimator based on a framework built on the analysis of DCT filtered natural images. Zhai [17] developed a framework for estimating image noise using two important characteristics of kurtosis and scale invariance of natural images on some transformed domains. Recently, many researchers have applied deep learning methods to solve the problem of image denoising. Jain and Seung [18] proposed a new method that uses convolutional neural networks (CNN) for image denoising and claimed that CNN has similar or even better capability than MRF models. Vincent [19] used the method of stacked denoising self-encoder (SDA) for image denoising and obtained good denoising result. HC Burger [20] attempted to learn the mapping from noisy images to noise-free images by using ordinary multi-layer perceptron (MLPs). Lore [21] used the self-encoder method of deep learning to train features of different low-light image signals for adaptive denoising. Chen [22] described a flexible learning framework (named as TNRD) based on the concept of nonlinear reaction diffusion model to solve various image denoising or recovery problems and obtained good performance that is comparable to BM3D [6].

Although the existing methods can solve the image denoising problem and improve the image quality, these still have shortcomings and limitations. The first one is that some of their hyper-parameters require manual selection based on experience. Thus it leads to slow convergence, lengthy training, and a large consumption of computational resources. Secondly, most of these methods assume that the image noise type is a certain one like Gaussian noise in a specific level. Therefore, the utilized denoising networks can only remove specific noise type in a certain level, which is not robust in the actual situation. Although Kai Zhang tried to solve this problem [23] by extending the removal of specific Gaussian noise to blind Gaussian noise, but it still has high limitations as it should be manually constructed to fit various noise types.

To solve the problems of the existing methods, our framework apply two types neural networks that are one for noise classification and the other for image denoising. The classification is a challenge as this decides the selection of denoising methods. Thus we utilized some classification networks on the noisy images that are from three public datasets: MNIST, Cifar-10 and BSD500. For the denoising, considering the visualization, our framework selected the images that are from high-quality dataset BSD500 and used these to produce noisy ones. Then we selected some

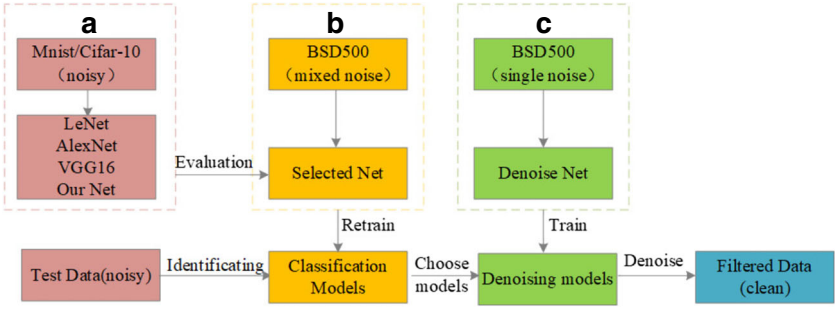

Fig. 2 The process of our framework

denoising models that are derived from the same denoising network [23] and trained each of them on the related type of noisy images. Based on the classification result, the denoising models are utilized one-by-one to denoise images.

3 Our framework

Our framework mainly consists of four processes: data preprocessing, noise classification, denoising, and testing. In the data preprocessing, our framework generates 4 groups of noisy images that are added single, two, multiple types of noises and mixture of these 3 groups. In the mixed types of noises case, each image may contain single, two or multiple types of noises. Then these grouped datasets are utilized to train corresponding classification models. The denoising models derive from a same denoising network and each of them is trained on corresponding single type noisy images. Based on the classification result, our framework selects related denoising models and use them one-by-one to denoise the images.

The overall framework is shown in Fig. 2 including the processes of classification and denoising. To select the best network for the classification, our framework evaluates the existing networks and our modified one on the noisy datasets that derived from MNIST, Cifar-10 and BSD500. As it is shown in Fig. 2a, it selects the classification network that achieves the highest accuracy on the datasets. Based on the evaluation, it trains the selected network to classify the noise type as Fig. 2b shows. Our framework selects the noisy images that are derived from BSD500 large-scale dataset for the visualization of the denoising results. Finally, as Fig. 2c shows, we use pre-trained models to denoise the image based on the classification result.

The process of generating noisy sample set can be formulated as below:

$$\hat{S} = Adn(S, \hat{L}), \; for \; \hat{L} \subset \{L_i\} and \; i \in (1, ..., N_{type}), \quad (1)$$

where S is the set of samples. L_i is the type of noise and N_{type} is the number of noise type. $\hat{L}$ is a sequence that records the order of adding different types of noise. $Adn(\cdot)$ is the function of adding noise to samples.

The process of training noise type classification network can be formulated as below:

$$MC = Train(\hat{S}^{train}, \hat{L}^{train}), \; for \; \hat{S}^{train} \subset \hat{S} \\ and \; \hat{L}^{train} \subset \hat{L}, \quad (2)$$

where MC is a classification network that is based on CNN. $\hat{S}^{train}$ is the training dataset and $\hat{L}^{train}$ is the corresponding label set.

The process of training denoising network on noisy samples can be formulated as below:

$$MD_i = Train(\hat{S}^{(i)train}, S^{train}, \hat{L}^{(i)train}), \\ for \; \hat{S}^{(i)train} \subset \hat{S}^{(i)} = Adn(S, L^{(i)train}) \\ and \; \hat{L}^{(i)train} \subset \hat{L}, \quad (3)$$

where MD is a denoising network based on CNN. MD_i is a model that is trained on related single type noisy dataset by the process $Train(\cdot)$. $\hat{S}^{(i)train}$ is the training dataset of single type noise. Then for a sample $\hat{S}_j \in \hat{S}^{test} = \hat{S} - \hat{S}^{train}$, where $j = (1, ...N_{sample})$, we can predict the noise types that are contained by the sample as below:

$$L_j^{(p)test} = MC(\hat{S}_j), \quad (4)$$

where $L_j^{(p)test}$ is the predicted label by pre-trained classification model MC. Actually $L_j^{(p)test}$ is a sequence of noise types that may be contained by the sample. Thus we can use related MD_k to denoise the sample. For example, if we set $\hat{S}_j^{input} = \hat{S}_j$ and k is the first value of sequence $L_j^{(p)test}$, then we can summarize the continuous process of denoising as below:

$$\hat{S}_j^{denoised} = MD_k(\hat{S}_j^{input}), then \; set \; \hat{S}_j^{input} \\ = S_j^{denoised} \; and \; k = L_j^{(p)test} \rightarrow next(k), \quad (5)$$

where $L_j^{(p)test} \rightarrow next(k)$ is the operation of returning next value of the k in the sequence $L_j^{(p)test}$. In more details, Algorithm 1 shows the process of our framework step by step.

Algorithm 1 The process of our framework.

STEP1: Preprocessing

Set S as the set of original samples. L_i is the noise type, $i = (1, ..., N_{type})$, N_{type} is a number of noise type. $Adn(\cdot)$ is the function of adding noise to samples.

for i=1 to N_{type} **do**

$S^{(i)} = Adn(S, L_i)$

$L^{(i)} = L_i$

end for

Then we can get the set of single type noisy samples $S^{single} = \left\{S^{(i)}\right\}$, and the corresponding label set of noise type $L^{single} = \left\{L^{(i)}\right\}$. Following the similar process, we can get the set of samples S^{two} and its label set L^{two} by adding two kinds of noises to the original data. By adding three or more kinds of noises, we can get S^{multi} and L^{multi}. When a label belongs to L^{two} or L^{multi}, the label also indicates the sequence of added noise types.

STEP2: Classification

Set $\hat{L} \subseteq \left\{L^{single}, L^{two}, L^{multi}\right\}$ is the label set of sample set $\hat{S} \subseteq \left\{S^{single}, S^{two}, S^{multi}\right\}$. N_{sample} is the number of samples in $\hat{S}$.

Set $N_{train} < N_{sample}$.

Set MC_k is a classification neural network,$k = (1, ..., N_{network})$, $N_{network}$ is the number of neural networks.

for k=1 to $N_{network}$ **do**

Set $\hat{S}^{train} = \left\{\hat{S}_j\right\}$, $\hat{L}^{train} = \left\{\hat{L}_j\right\}, j = (1, ..., N_{train})$

Set $\hat{S}^{test} = \left\{\hat{S}_j\right\}$, $\hat{L}^{test} = \left\{\hat{L}_j\right\}, j = (N_{train} + 1, ..., N_{sample})$

Then train each neural network MC_k on $\hat{S}^{train}$ and $\hat{L}^{train}$.

end for

Set $P_k = Rate(MC_k(\hat{S}^{test}), \hat{L}^{test})$, which is to check the correct rate of MC_k on $\hat{S}^{test}$, $k = (1, ..., N_{network})$.

Then we can select the best network by $MC_x = argmax_{MC_k}(P_k), k = (1, ..., N_{network})$.

STEP3: Denosing

Set MD is a denoising neural network. Through **STEP1**, we obtained $S^{(i)}$ and $L^{(i)}$, $i = (1, ..., N_{type})$, N_{type} is the number of noise type, N_{sample} is the number of the samples in $S^{(i)}$.

for i=1 to N_{type} **do**

Set $S^{(i)train} = \left\{S_j^{(i)}\right\}$, $L^{(i)train} = \left\{L_j^{(i)}\right\}, j = (1, ..., N_{train})$

Set $S^{(i)test} = \left\{S_j^{(i)}\right\}$, $L^{((i)test} = \left\{L_j^{(i)}\right\}, j = (N_{train} + 1, ..., N_{sample})$

Then we trained a model of MD, which is MD_i on $S^{(i)train}$ and $L^{(i)train}$. Each MD_i is used to denoise the noise type L_i.

end for

STEP4: Testing

Set MC_x is the best network that is selected by the **STEP2**. For $\hat{S}^{test}$ and $\hat{L}^{test}$ of **STEP2**, we use MC_x to classify the noise type, which result is $\hat{L}^{(p)test}$. Then $Rate(\hat{L}^{(p)test}, \hat{L}^{test})$ denotes the correct rate of MC_x on $\hat{S}^{test}$.

Set N_{sample} is the number of samples in $\hat{S}^{test}$.

for j=1 to N_{sample} **do**

- If $\hat{L}_j^{(p)test} \in L^{single}$, it uses MD_i to denoise $\hat{S}_j^{test}$ to get denoised sample $S_j^{denoised}$, $i = \hat{L}_j^{(p)test}$.
- Else, it means $\hat{L}_j^{(p)test} \in L^{two}$ or L^{multi}, thus our framework uses MD_i of **STEP3** one by one following the sequence of added noises indicated by $\hat{L}_j^{(p)test}$, $i \in (1, ..., N_{type})$. Then it can get denoised sample $\hat{S}_j^{denoised}$.

end for

Then we get the denoised set $\hat{S}^{denoised} = \left\{\hat{S}_j^{denoised}\right\}$ from the $\hat{S}^{test}$. Each sample of $\hat{S}^{test}$ may not only contain single type noise, but also two or multiple types of noises.

3.1 Data preprocessing

The data preprocessing is shown in Fig. 3. It firstly added noise to the image that are single, two, multiple (3 or 4) types and mixed of them. The images derive from three datasets: MNIST, Cifar-10 and BSD500. We give the sample method of adding Gaussian (Ga) noise is as below:

$$Ga : P_{out} = P_{in} + X_{means} + sigma \times G(d), \tag{6}$$

where P_{out} means the noisy image and P_{in} means the original image. $G(\cdot)$ represents a Gaussian distribution generated based on X_{means} and $sigma$, where the former represents the mean value(zero) and the latter represents the variance value(unit). d is a pseudo-random number generated based on the system time and we put this into $G(\cdot)$ to obtain the corresponding Gaussian distribution random value.

The method of adding salt (Sa) noise is as below:

$$Sa : P_{out} = \begin{cases} P_{in} Original^{n-k}(x, y) \\ P_{in} Random^{k}(x, y) = 0 \ or \ 255 \ , \end{cases} \tag{7}$$

where P_{out} means the noisy image which contains two parts. The original image contains n pixels, from which we randomly take k pixels and assign their values to 0 or 255. The remaining $n-k$ pixels remain unchanged. In our paper, k occupies 40 percent of n, and the selected pixel is set to 0 or 255 by the same probability.

The method of adding speckle (Sp) noise as below:

$$Sp : P_{out} = P_{in} + P_{in} \times F(avg, sig), \tag{8}$$

where P_{out} means the noisy image and P_{in} means the original image.$F(ave, sig)$ is uniformly distributed random noise with the parameters of mean avg and variance sig. $F(\cdot)$ is introduced in paper [24, 25].

The method of adding Poisson (Po) noise as below:

$$Po : P_{out} = P_{in} + Po(x), \tag{9}$$

where P_{out} means the noise image and P_{in} means the original image. $Po(x)$ represents Poisson-distributed noise generated based on pixel values of the original image referred to paper [26, 27].

Then it uses the noise-added datasets to train the models of the classification networks. The models can be summarized to 4 types based on the datasets: single-type noise, two-type noise, multiple-type noise and mixed-type noise. Single-type noise models can only classify single type noise as these are trained on the single type noise images. Two-type or Multiple-type noise models can classify two or multiple types of noises. Mixed-type noise model assumes that there are single or two or multiple types of noises in the image, so that, it is more robust in the real applications than the others. In order to pre-train the noise type classification models, we selected three public and large datasets: MNIST (Grayscale), Cifar-10 (Color)and BSD500(color and Grayscale). The dataset BSD500 contains color images and we converted it into grayscale images. Finally, we added single-type, two-type, multi-type and mixed-type noises on these datasets as it is introduced in (10).

$$\begin{aligned} NDC = Adn(&MNIST, Cifar-10, BSD500, S(Ga), \\ &S(Sa), S(Sp), S(Po)), \end{aligned} \tag{10}$$

where MNIST, Cifar-10 and BSD500 mean the datasets of MNIST, Cifar-10 and BSD500. S(Ga),S(Sa), S(Sp),S(Po) mean the selection of the related noise type. NDC means the final dataset for classification that contains noisy images. Finally, we got 4 sets of single-type noisy images, 6 sets of two-type noisy images and 5 sets of multi-type noisy images. By integrating these noise datasets, we got a dataset that obtain 15 types of noisy images. In order to train the denoising models,we selected the large dataset BSD500, which is convenient for visualization before and after denoising. We trained the denoising models on the dataset(derived from BSD500) as it is introduced in (10) and evaluated the performance.

3.2 Classification network

As paper [28] mentioned, the noise variance is highly content-correlated. This inspires us to use CNNs to extract the features and perform the classification of noise type. CNNs are good alternatives in many kind of pattern recognition. The application of neural networks involves feature extraction, activation function and training of the network. Thus the CNNs can be trained to classify the content-correlated noise by extracting the features of different type noisy images.

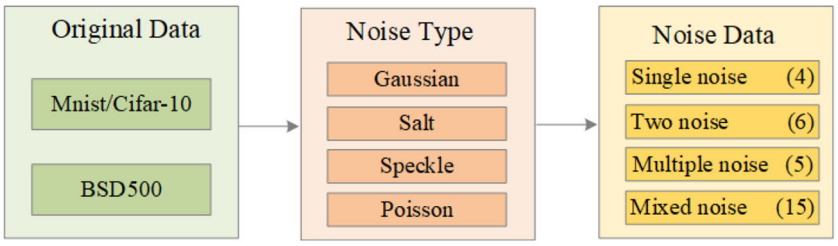

Fig. 3 Construct the noisy datasets

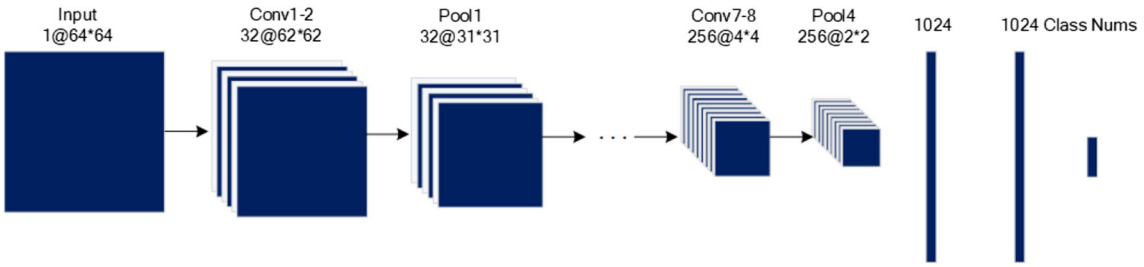

Fig. 4 The structure of classification network

To evaluate the possibility of improving classification networks, our framework designed a network for classification that is based on VGG16 [29]. Our network includes 8 convolutional layers, 4 downsampling layers and 3 fully connected layers. The network structure is shown in Fig. 4. Detailed network layer parameters can be found in Table 1. Conv1-2, Conv3-4, Conv5-6 and Conv7-8 mean the convolutional layers. Pool1, Pool2, Pool3 and Pool4 mean the pooling layers. FC1, FC2, FC3 mean the fully connected layers.

Our framework adjusted the size of noise images to 64×64 and then put these to the classification network for training. The first two convolutional layers have 32 convolution kernels, each has a size of 3×3 and a sliding step size of 1. Similarly, the third and fourth convolutional layers have 64 convolution kernels, the fifth and sixth convolutional layers have 128 convolution kernels and the seventh and eighth convolutional layers have 256. The size and sliding step size of the convolution kernel are the same as the first two convolutional layers. Between every two convolutional layers, it performed the LReLU [30] activation function. After each of the two convolutional layers, we designed a maximum down-sampling layer with a convolution kernel size of 2×2, a sliding step size of 2 and batch normalization [31] operations were performed. At the end of the network, it designed three fully connected layers: the first two layers of 1024 neurons; the last layer is for categories and uses the $Softmax$ [32] function to generate a vector that presents the probability of the labels of classification. In the training process, we take $Cross-Entropy$ as the loss function which formula is as follows:

$$H_{y'}(y) = -\sum_{i} y'_i log(y_i), \tag{11}$$

where y'_i is the ith value in the actual label, and y_i is the corresponding ith component of the vector after being normalized by $Softmax$. The higher accuracy the classification reaches, the more y_i is closer to 1 and $H_{y'}(y)$ gets smaller value.

3.3 Denoising network

We treat the problem of image denoising as a plain discriminative learning problem, which is to separate the noise from the image by feed-forward CNN. The reason of using CNN are because of two factors. First, CNN with deep architecture is effective to increase the capacity and flexibility for exploiting image features. Second, many advances have been achieved on regularization and learning skills for training CNN including batch normalization [31], variant LReLU [30] and residual learning [4]. These methods help CNN to speed up the training process and improve the denoising performance.

Table 1 Parameter configuration of our network

Layers	*Kernel_num/Channels*	*Kernel_size*	*Stride*	*Padding*	*Type*
Conv1-2	32	3 × 3	1	VALID	—
Pool1	32	2 × 2	2	VALID	MAX
Conv3-4	64	3 × 3	1	VALID	—
Pool2	64	2 × 2	2	VALID	MAX
Conv5-6	128	3 × 3	1	VALID	—
Pool3	128	2 × 2	2	VALID	MAX
Conv7-8	256	3 × 3	1	VALID	—
Pool4	256	2 × 2	2	VALID	MAX
FC1	1024	—	—	—	—
FC2	1024	—	—	—	—
FC3	Class_num	—	—	—	—

The input of our denoising network can be seen as a noise observation $y = x + v$. Discriminative denoising models such as MLP [20] and CSF [33] aim to learn a mapping function $F(y) = x$ to predict the latent clean image. It adopts the residual learning formulation to train a residual mapping $R(y) \approx v$ and then has $x = y - R(y)$.

$$L(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \| \mathbf{R}(\mathbf{y}_i; \Theta) - (\mathbf{y}_i - \mathbf{x}_i) \|_F^2 . \quad (12)$$

Formally, the averaged mean squared error between the desired residual images and estimated ones from noisy input can be adopted as the loss function to learn the trainable parameters Θ by neural networks. Here $\{(y_i, x_i)\}_{i=1}^{N}$ represents N noisy-clean training image (patch) pairs. Figure 5 illustrates the architecture of the denoising network for learning $R(\mathbf{y}_i; \Theta)$.

In this paper, our framework modified a denoising CNN network [23] based on VGG16 [29] to train various denoising models. The network has a total $(2d + 1)$ layers, where d is the depth of the network and is primarily set according to the effective patch size. In order to make full use of the context information in the image area, it increases the size of the experience field as much as possible. A reasonable depth d is important to help us balance the performance and efficiency of the network model.

Our denoising network mainly consists of three types of layers, which are shown in Fig. 5 and represented by three different colors. For the first layer, we used 96 sizes of 3×3×c filters for convolution and generated 96 feature maps. We used the variant LReLU [30] of the rectified linear unit for nonlinearity when inputting the output of each layer to the next one. c represents the number of channels of the training picture. We set c as 1 for gray-scale and 3 for color map. From the 2nd to (d-1)th layers, we used 96 filters of size 3×3×96. We used LReLU and performed batch normalization [31] operation between the convolutional layer and LReLU layer, thereby it can increase the speed of training and the convergence process. For the last layer, we reconstructed the output using c filters of size 3×3×96. We also removed all of the down sampling layers because this greatly reduces the denoising effect of our model.

4 Experiment

4.1 Dataset

As Section 3.1 introduced, our experiment selected BSD500, Mnist and Cifar-10 datasets. We added single type, two types, multiple types and mixed types of noise to the samples. As a result, we obtained sampling datasets of various noise types including 4 sets of single type noise, 6 of two types of noises, 5 of multi-type noises and 15 types of mixed noises. For the noise classification network, we input various noise data that derived from MNIST and Cifar-10 into the network for pre-training to obtain multiple classification models. In order to match the denoising network, we also input the 15 types of mixed noisy data that derived from BSD500 into the classification network and trained the corresponding noise classification models. When training the denoising models, we only input the datasets(derived from the BSD500) that each of them contain single type noise into the denoising network to obtain some denoising models.

4.2 Classification results

For single-type noisy dataset, we used 32,000 images for training and the remaining 8000 for testing. Similarly, for two-type noisy dataset, the number of training set is 48,000 and the number of testing set is 12,000. The multi-type noisy training set contains 40,000 samples, which are separated into 30,000 for training set and 10,000 for the testing set. In addition, our experiment integrated the these datasets to form a mixed noisy dataset. The training set contains 120,000 samples and the testing set contains 30,000.

We selected the LeNet [34], AlexNet [35],VGG16 [29] to train classification models for comparison. LeNet [34] consists of two convolutional layers, two averaged pooling layers and two fully connected layers. The convolution kernel size is 5*5 and the number of fully connected layer units are 120 and 84, respectively. AlexNet [35] consists of five convolutional layers, three maximum pooling layers and three fully connected layers. The convolution kernel sizes of the first, second, and last three convolutional layers

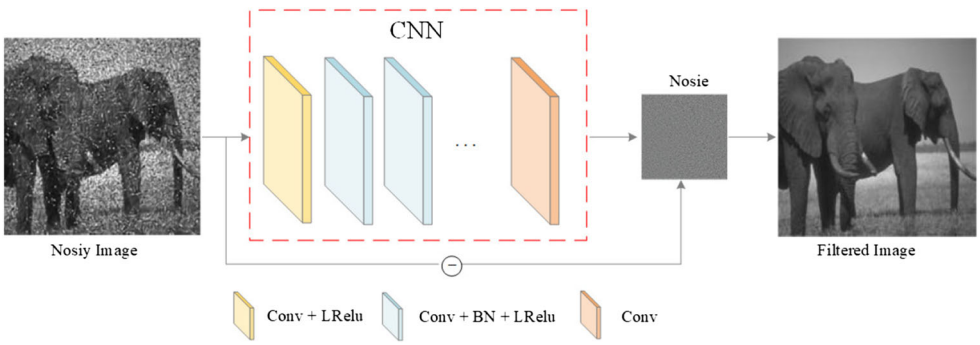

Fig. 5 Denoising network model structure

Table 2 Noise recognition rates on Cifar-10 / Mnist datasets

Network	*Single_noise*	*Two_noise*	*Multip_noise*	*Mixed_noise*
LeNet [34]	98.1% / 74.9%	87.1% / 85.2%	63.1% / 69.1%	61.6% / 52.8%
AlexNet [35]	99.2% / 76.1%	90.7% / 85.2%	61.6% / 71.1%	54.4% / 49.8%
VGG16 [29]	99.5% / 74.8%	92.8% / 98.7%	67.3% / 77.6%	74.6% / 77.1%
Our Net	99.8% / 88.5%	93.0% / 96.4%	66.6% / 79.3%	74.3% / 87.9%

are 11*11, 5*5 and 3*3. The number of units in the fully connected layer are 4096, 4096 and 1000, respectively. VGG16 [29] contains 13 convolution layers with a convolution kernel size of 3*3, 3 maximum pooling layers and 3 fully connected layers with 4096, 4096 and 1000 units.

We adjusted the image size to 64×64 for network training, batch-size was set to 128 and the learning rate was 0.0001. For mixed-type noise, we trained 40 epochs and the other three trained 20 epochs. After the training, we evaluated the performance of the classification models and found that our designed network has the highest recognition rate for the classification of noise type. The comparison results can be seen in Table 2. The VGG16 network has a slightly higher recognition rate for only certain types of noises than our designed network. Therefore, our network is still competitive in general cases. To match the denoising network, we also input the noisy dataset that is derived from BSD500 into the classification network, of which 6000 images are used for training and the remaining 1500 samples are used for testing. The recognition rate can be seen in Table 3. It also shows that our network has higher accuracy than other ones.

4.3 Denoising results

4.3.1 Single type noise

In order to clearly visualize the comparison before and after denoising, our experiment selected the large-scale dataset that is Berkeley segmentation dataset (BSD500) and processes this to obtain the corresponding gray-scale images. Then single-type noise is added to this and we got four kinds of noisy datasets in color and gray. Taking the gray-scale image as an example, we adjusted the image size to 200×200, and then selected 400 of them for training and the remaining 100 for testing. We set the noise level to $\sigma \in [0,55]$, the patch sizes to 35×35 and 128×1600 to train the model. Similarly, we also trained the denoising model of the color image. The noise level is set to $\sigma \in [0,55]$, the patch sizes are set to 43×43 and 128×3200 to train the model.

To capture enough information to denoise the images, we set the network depth to 17 and epochs to 30 to train the gray-scale image denoising model. When training the color image denoising model, we set the network depth to 21 and trained 50 epochs. During the training, we used the SGD [36] with weight 0.0001, the momentum to 0.8 and the batch-size to 128.

Since there are 4 kinds of noisy datasets, we finally got 8 kindes of denoising models, including 4 gray-scale and 4 color image denoising models, corresponding to Gaussian, salt and pepper, speckle and Poisson denoising. We used these denoising models to denoise the test set of the noisy images(BSD100). When denoising grayscale images, our grayscale denoising models have achieved good results. However, when denoising color images, only the Gaussian denoising model works well and other denoising models have poor results. Therefore, for color images with non-additive Gaussian noises, we carry out a novel method called $RGBSM$(SM means split-merge) to conducted additional experiments. $RGBSM$ separates color images into three channels of RGB, then denoises each channel image with grayscale denoising model corresponding to the noise type and integrates the denoised channel results into color images. The average values of PSNR and SSIM are shown in Table 4. The comparison of sample pictures before and after denoising is shown in Figs. 6 and 7. When denoising color images, $RGBSM$ achieves higher evaluation values and better visualization results than color denoising models, which can be seen in Table 4 and Fig. 7, respectively. However, images denoised by the $RGBSM$ method lose some of the texture information compared with the ground-truth images and sometimes contains some highlight plaques. We also utlized $RGBSM$ for color images with Gaussian noise but there is almost no improvement compared with the color denoising models.

We also compared some existing denoising methods including BM3D [6], MLP [20], TNRD [22], MemNet [37].

Table 3 Noise recognition rates on BSD

Dataset	*Image_type*	*LeNet* [34]	*AlexNet* [35]	*VGG16* [29]	*OurNet*
BSD500	Gray	78.6%	83.4%	90.1%	90.1%
	RGB	81.3%	87.5%	93.5%	93.7%

Table 4 Single-type denoising models: PSNR(DB) and SSIM values

Dataset	*Noise_type*	*Ga*	*Sa*	*Sp*	*Po*
BSD100	Gray	29.07 / 0.8822	27.54 / 0.9401	28.35 / 0.8391	30.03 / 0.9337
	RGB	30.58 / 0.9276	11.82 / 0.2926	12.33 / 0.3219	12.44 / 0.3361
	RGBSM	30.02 / 0.9134	21.42 / 0.6376	27.26 / 0.8122	28.14 / 0.8321

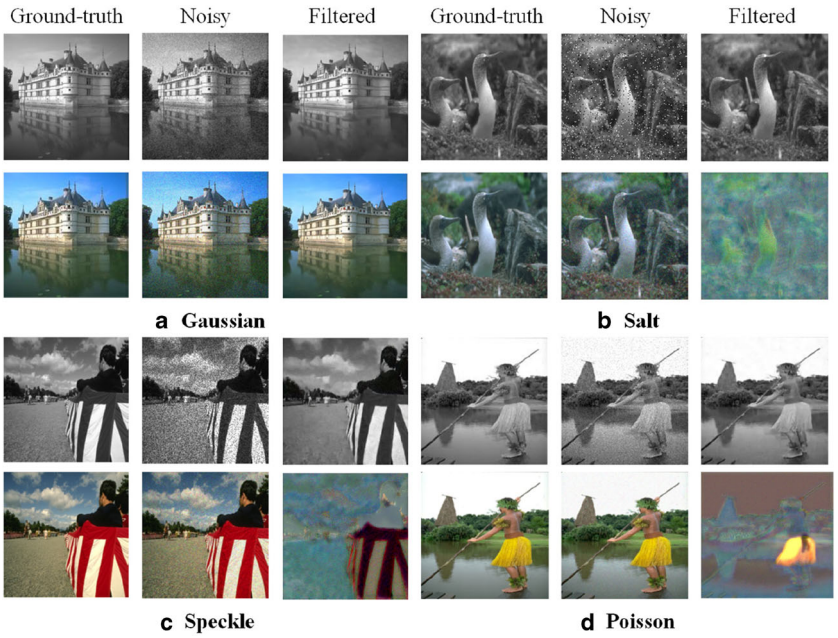

Fig. 6 Denoising results for some single-type noise images

Fig. 7 Denoised results for some non-additive Gaussian noise images based on RGBSM

Table 5 Comparison of PSNR(DB) and SSIM values of different methods

Noise_level	*Image_type*	*BM3D* [6]	*MLP* [20]	*TNRD* [22]	*MemNet* [37]	*Our*
σ=30	Gray	28.17 / 0.8722	28.38 / 0.8725	28.92 / 0.8736	29.04 / 0.8750	29.07 / 0.8822
	RGB	28.34 / 0.8621	28.52 / 0.8716	29.63 / 0.8824	29.92 / 0.8996	30.58 / 0.9276

BM3D [6] employes a strategy based on transform domain enhanced sparse representation to achieve image denoising by grouping similar 2-D image segments into 3-D data arrays. MLP [20] utilizes a common multi-layer perceptron applied to image patches to learn a mapping from noisy images to noise-free images by eliminating image noise. TNRD [22] trains nonlinear reactive diffusion models by highly parametrized linear filters as well as highly parametrized influence functions , which can be learned from training data based on loss-based methods to achieve image denoising. MemNet [37] introduced multiple stacked memory blocks containing recursive units and gate units to learn multi-layer representations in different receptive fields. It combines multiple convolutional operations before and after the memory blocks to achieve image restoration.

As the existing denoising methods mostly deal with Gaussian denoising, we compared the average PSNR and SSIM values of the testing set after Gaussian denoising. The noise level σ is set to 30 when compared with these methods. Experimental results show that our Gaussian denoising model obtains both higher PSNR and SSIM values than the existing methods. Although BM3D [6] and MLP [20] have achieved good PSNR values in some cases, they all have their own limitations. One of them is that these require a high degree of complexity and are not suitable for parallel computing. The other one is that these are tailored to a single level of noise and do not work well on other noise levels. TNRD [22] is capable of efficient parallel training on the GPU, but relies on a lot of prior knowledge and has limitations in capturing all features of the image. MemNet [37] gets good experimental results, but its PSNR and SSIM values are lower than our network model. The comparison can be seen in Table 5. The comparison of some samples before and after denoising is shown in Fig. 8, where (a) to (f) are gray-scale contrast maps and (g) to (l) are color contrast maps.

4.3.2 Two and multiple types of noises

To test the removal of two-type and multiple-type noises, our method also selected 100 noisy images that are derived from the BSD500. If the testing images are identified by the classification network and found that it contains two or more types of noises, it goes through the corresponding single-type noise removal models in related order to denoise the image. If the image contains Gaussian noise, we remove this first. If not, we calculate a new evaluation index named PSSI and determine the order of single denoising models based on the PSNR and SSIM values of Table 4 except Gaussian noise, which is calculated as (13). Then, we obtained the sequence (Po, Sp, Sa) by sorting the calculated PSSI values from high to low. Finally, we follow the (Ga, Po, Sp, Sa) sequence through the corresponding single denoising model when removing two and multiple

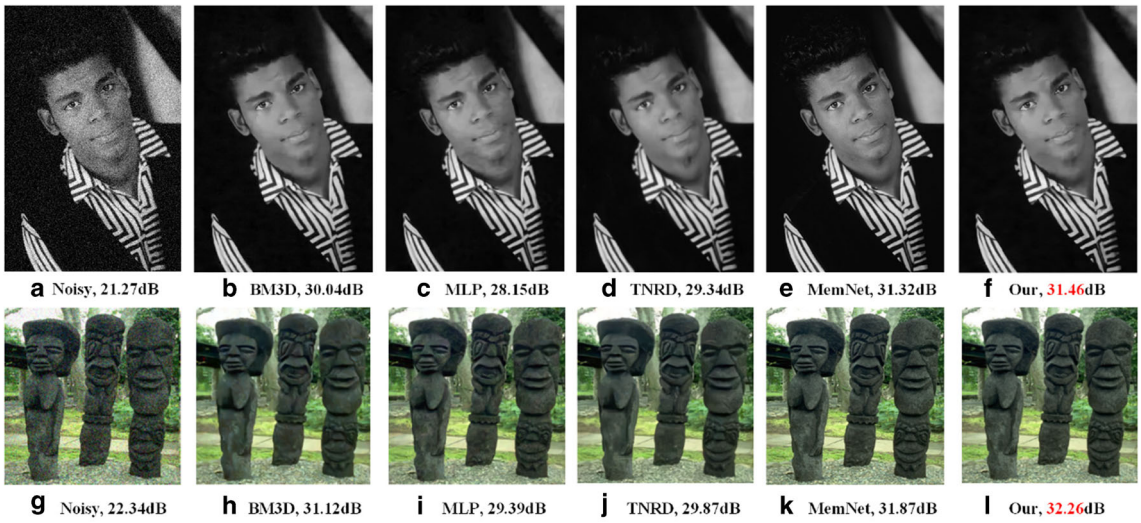

Fig. 8 Denoised results for some images between different methods

Table 6 Denoising two-type noises: PSNR(DB) and SSIM values

Dataset	*Noise_type*	*Ga&Sa*	*Ga&Sp*	*Ga&Po*	*Sa&Sp*	*Sa&Po*	*Sp&Po*
BSD100	Gray	21.04 / 0.5205	25.96 / 0.7352	27.17 / 0.8195	27.11 / 0.7983	26.83 / 0.7731	27.52 / 0.8409
	RGB	11.14 / 0.3501	11.89 / 0.3137	11.90 / 0.3502	11.59 / 0.2447	11.81 / 0.2856	12.37 / 0.3032
	RGBSM	21.02 / 0.6014	23.56 / 0.6834	24.15 / 0.6892	18.94 / 0.5123	19.23 / 0.5347	20.13 / 0.5426

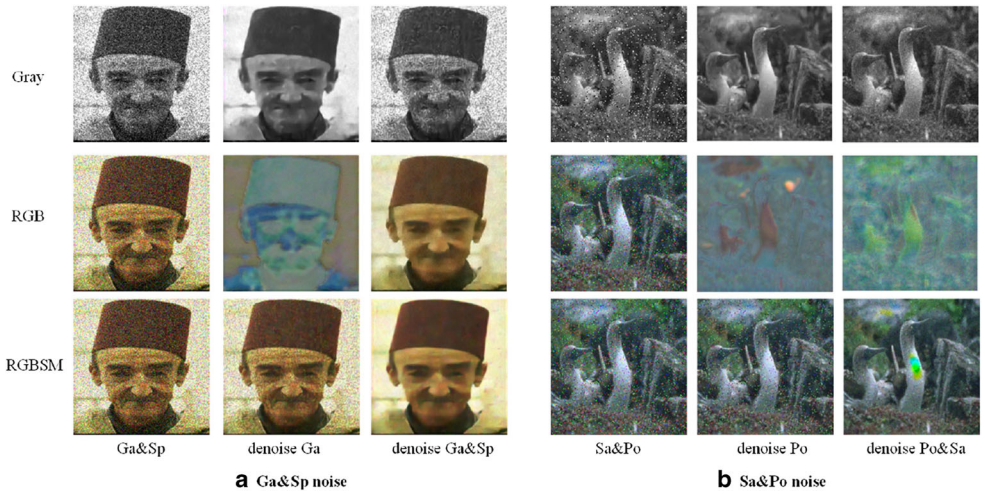

Fig. 9 Denoised results for some two-type noisy images

Table 7 Denoising multi-type noises: PSNR(DB) and SSIM values

Dataset	*Noise_type*	*Ga&Sa&Sp*	*Ga&Sa&Po*	*Ga&Sp&Po*	*Sa&Sp&Po*	*Ga&Sa&Sp&Po*
BSD100	Gray	19.79 / 0.5484	18.47 / 0.5395	16.17 / 0.5169	16.92 / 0.5067	15.38 / 0.4801
	RGB	12.45 / 0.3740	12.70 / 0.3881	12.74 / 0.3754	11.61 / 0.2458	12.63 / 0.3742
	RGBSM	14.47 / 0.4023	15.13 / 0.4076	17.78 / 0.5121	13.17 / 0.3812	13.08 / 0.3125

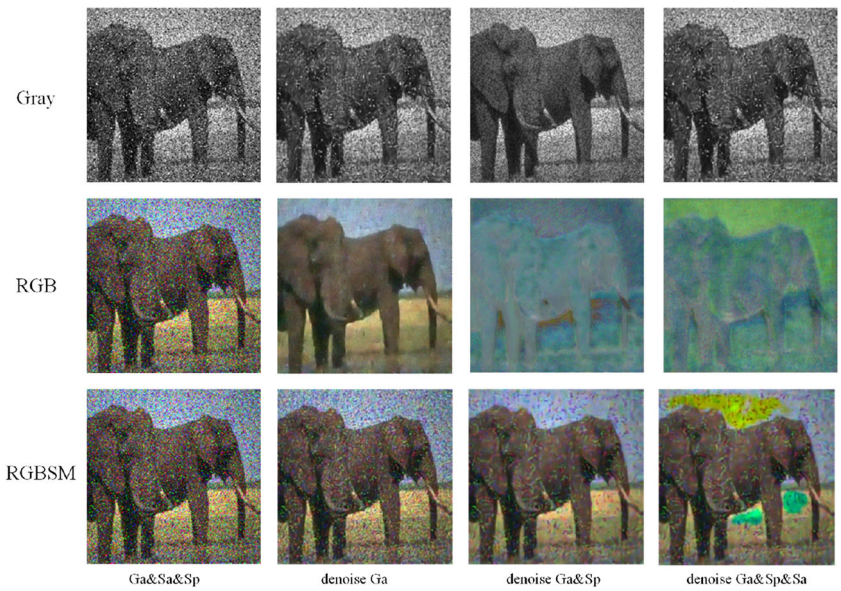

Fig. 10 Denoised results of Ga&Sa&Sp noisy images

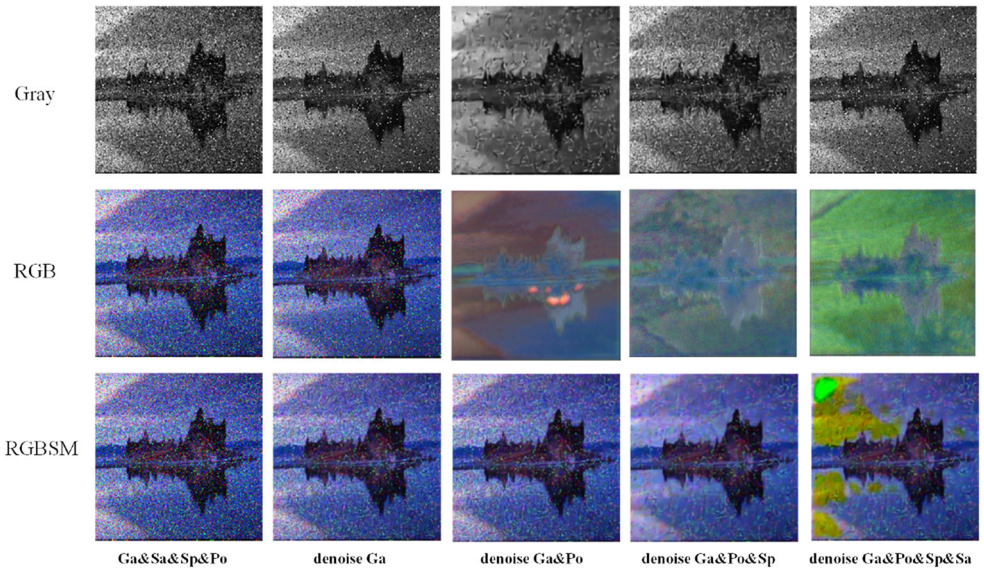

Fig. 11 Denoised results of Ga&Sa&Sp&Po noisy images

types of noises. For color images with two-type and multiple-type noises, we also experimented the proposed $RGBSM$ method, and the performance has an improvement compared with the color denoising models.

$$PSSI = 0.6 \times PSNR + 0.4 \times SSIM \tag{13}$$

The PSNR and SSIM values of the two types of noisy pictures are shown in Table 6. We selected Ga&Sp and Sa&Po as the examples and the comparison of some images before and after denoising is shown in Fig. 9. The PSNR and SSIM values after removal of multiple types of noisy samples are shown in Table 7. We selected the samples that contain Ga&Sa&Sp and Ga&Sa&Sp&Po. The comparison of sample images before and after denoising is shown in Figs. 10 and 11.

4.3.3 Discussion

Our models can achieve good result in both classification of noise type and denoising effect. On the other side, the denoising results are not good enough in some color or multi-type noisy image cases. In gray-scale case, there is only one color channel that makes deep learning methods easy to capture features to restore images. In color image case, multiple color channels(RGB) cause the restoration is a challenge as it is hard to get good result in all of these channels. In multi-type noises case, there is also the same problem as a good restoration in one type noise may cause the bad result in the other type of noise.

Increasing the size and variety of dataset may be a good solution for these problems, but we will try to solve this in another way. To solve these problems, we can try to independently perform the restoration on each channel and balance these. When restoring the multi-type noise, we can also solve these problems by balancing multiple channels. In other words, each type of noise can be seen as a individual channel and we should balance the restoration among the channels to get good result. We will perform this solution in our future work.

For the noisy images that contain multiple types of noise, designing an end-to-end model that can directly remove multiple types of noise is ideal and efficient but it is a big challenge. When different types of noises are mixed, the distributions make the contents and features complicated. This leads to great challenges in the loss function design and optimization process of the network model. Although the method we proposed requires more computational budget, it is easier to understand and implement because the single type of noise distribution is relatively simple.

5 Conclusions and future work

This paper utilized two kinds of neural networks: one is used to classify the types of noise contained in the image; the other one selects suitable denoising models to remove noise according to the classification result. If the image contains only a single type of noise, it can easily choose the corresponding denoising model. Otherwise, the noise image passes through a sequence of corresponding denoising models in a related order to remove the noise. Experimental results show that our noise classification model achieves good recognition rate for single or more types of noises. In addition, for gray-scale images, our denoising model obtains higher PSNR and SSIM values on various types of noises. The denoising effect on Gaussian noise is particularly prominent and superior to the

existing denoising methods. For color images, our Gaussian denoising model also works well but the other models have poor results. Therefore, we have adopted a novel method called $RGBSM$ to improve the denoising effect of color noise images.

In the future work, we will design a better denoising network and train various noise denoising models to improve the denoising effect of color images. Our plan is to design a network that can directly remove multiple types of noise by using multi-channel methods with balancing skills, rather than denoising through models one by one or optimizing the result in each channel. Additionally, we will try to use our framework to denoise the specific images such as medical CT images or remote sensing SAR images, etc.

Acknowledgements This work has been supported by the National Natural Science Foundation of China (Grant No. 61802279, 6180021345, 61702366, 61602342 and 51607122) and Natural Science Foundation of Tianjin (Grant No.16JCYBJC42300, 16JCYBJC41500 and 17JCQNJC00100).

References

1. Buades A, Coll B, Morel JM (2005) A non-local algorithm for image denoising. Comput Vis Pattern Recogn, 60–65
2. Mohamed AR, Hinton G, Penn G (2012) Understanding how deep belief networks perform acoustic modelling. IEEE Int Conf Acoust, Speech Signal Process, 4273–4276
3. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst, 1–9
4. Ulas C, Tetteh G, Kaczmarz S et al (2018) DeepASL: kinetic model incorporated loss for denoising arterial spin labeled MRI via deep residual learning. In: International conference on medical image computing and computer-assisted intervention, pp 30–38
5. Chierchia G, Cozzolino D, Poggi G et al (2017) SAR image despeckling through convolutional neural networks. In: IEEE International geoscience and remote sensing symposium (IGARSS), pp 5438–5441
6. Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Trans Image Process 16:2080–2095
7. Xu J, Zhang L, Zuo W, Zhang D, Feng X (2015) Patch group based nonlocal self-similarity prior learning for image denoising. In: International conference on computer vision, pp 244–252
8. Osher S, Burger M, Goldfarb D, Xu J, Yin W (2005) An iterative regularization method for total variation-based image restoration. Multiscale Model Simul 4:460–489
9. Weiss Y, Freeman WT (2007) What makes a good model of natural images? In: IEEE Conference on computer vision and pattern recognition, pp 1–8
10. Elad M, Aharon M (2006) Image denoising via sparse and redundant representations over learned dictionaries. IEEE Trans Image Process 15:3736–3745
11. Dong W, Zhang L, Shi G, Li X (2013) Nonlocally centralized sparse representation for image restoration. IEEE Trans Image Process 22:1620–1630
12. Xie J, Xu L, Chen E (2012) Image denoising and inpainting with deep neural networks. In: International conference on neural information processing systems, pp 341–349
13. Agostinelli F, Anderson MR, Lee H (2013) Robust image denoising with multi-column deep neural networks. Advances in Neural Information Processing Systems
14. Xu L, Ren JSJ, Liu C et al (2014) Deep convolutional neural network for image deconvolution. In: International conference on neural information processing systems, pp 1790–1798
15. Patidar P, Gupta M, Srivastava S et al (2010) Image denoising by various filters for different noise. Int J Comput Appl 9:24–28
16. Gupta P, Bampis CG, Jin Y et al (2018) Natural scene statistics for noise estimation. In: IEEE Southwest symposium on image analysis and interpretation (SSIAI)
17. Zhai G, Wu X (2011) Noise estimation using statistics of natural images. IEEE International Conference on Image Processing
18. Jain V, Seung S (2009) Natural image denoising with convolutional networks. Adv Neural Inf Proces Syst, 769–776
19. Vincent P et al (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11:3371–3408
20. Burger HC, Schuler CJ, Harmeling S (2012) Image denoising: can plain neural networks compete with BM3D? In: IEEE Conference on computer vision and pattern recognition, pp 2392–2399
21. Lore KG, Akintayo A, Sarkar S (2017) LLNet: a deep autoencoder approach to natural low-light image enhancement. Pattern Recogn 61:650–662
22. Chen Y, Pock T (2015) Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration. IEEE Trans Pattern Anal Mach Intell 39(6):1256–1272
23. Zhang K, Chen Y, Chen Y et al (2017) Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. IEEE Trans Image Process 99:1–1
24. Lopez-Martlnez C, Fabregas X (2003) Polarimetric SAR speckle noise model. IEEE Trans Geosci Remote Sens 41(10):2232–2242
25. Lopez-Martlnez C, Fabregas X, Pottier E (2005) Multidimensional speckle noise model. Eurasip J Adv Signal Process 20:1–13
26. Katti SK, Rao AV (1967) Handbook of the poisson distribution. J Oper Res Soc 10(2):412–412
27. Chandra NK, Roy D, Ghosh T (2013) A generalized poisson distribution. Commun Stat - Theory Methods 42(15):2786–2797
28. Yue H, Zhou S, Yang J, Sun X, Hou C (2018) Deep joint noise estimation and removal for high ISO JPEG images. In: 2018 24th International conference on pattern recognition (ICPR)
29. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations
30. Uchida K, Tanaka M, Okutomi M (2018) Coupled convolution layer for convolutional neural network. In: International conference on pattern recognition. IEEE, p 197
31. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on international conference on machine learning
32. Yuan B (2017) Efficient hardware architecture of softmax layer in deep neural network. In: System-on-chip conference. IEEE, pp 323–326
33. Schmidt U, Roth S (2014) Shrinkage fields for effective image restoration. In: IEEE Conference on computer vision and pattern recognition, pp 2774–2781

34. Lauer F, Suen CY, Bloch G (2007) A trainable feature extractor for handwritten digit recognition. Pattern Recogn 40(6):1816–1824
35. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: International conference on neural information processing systems, pp 1097–1105
36. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT, pp 177–186
37. Tai Y, Yang J, Liu X, Xu C (2017) MemNet: a persistent memory network for image restoration. In: The IEEE International conference on computer vision (ICCV), pp 4539–4547

Fan Liu was born in Hubei, China, in 1994. He received the Bachelor degree in Yancheng Teachers University, Jiangsu, China, in 2017.Since 2017, has been a graduate student in Tianjin Polytechnic University. He has published 3 research papers in international SCI/EI journals and conferences. His research interests include computer vision, AI, deep learning.

Qingzeng Song was born in Hebei, China, in 1980. He received the B.S. degree in Hebei University of Technology, Tianjin, China, in 2003. He received the Ph.D. degree in Hebei University of Technology, Tianjin, China, in 2012. Since 2012, has been an Associate Professor in Tianjin Polytechnic University. His research interests include reconfiguration computing, heterogeneous computing, embedded systems, electromagnetic Numerical Computation.

Guanghao Jin was born in Jilin, China, in 1979. He received the B.S. degree in Peking University, Beijing, China, in 2002. He received the Ph.D. degree in Tokyo Institute of Technology, Tokyo, Japan, in 2014. Since 2016, has been a Lecturer in Tianjin Polytechnic University. He has published about 20 research papers in international SCI/EI journals and conferences. His research interests include computer vision, AI, deep learning, heterogeneous computing.