Image Denoising Using Complex-Valued Deep CNN

Yuhui Quan[a], Yixin Chen[a,1], Yizhen Shao[a,1], Huan Teng[a], Yong Xu[a,*], Hui Ji[b]

[a]*School of Computer Science and Engineering, South China University of Technology, China.*
[b]*Department of Mathematics, National University of Singapore, Singapore.*

Abstract

While complex-valued transforms have been widely used in image processing and have their deep connections to biological vision systems, complex-valued convolutional neural networks (CNNs) have not seen their applications in image recovery. This paper aims at investigating the potentials of complex-valued CNNs for image denoising. A CNN is developed for image denoising with its key mathematical operations defined in the complex number field to exploit the merits of complex-valued operations, including the compactness of convolution given by the tensor product of 1D complex-valued filters, the nonlinear activation on phase, and the noise robustness of residual blocks. The experimental results show that, the proposed complex-valued denoising CNN performs competitively against existing state-of-the-art real-valued denoising CNNs, with better robustness to possible inconsistencies of noise models between training samples and test images. The results also suggest that complex-valued CNNs provide another promising deep-learning-based approach to image denoising and other image recovery tasks.

Keywords: Complex-Valued Operations; Convolutional Neural Network; Image Denoising; Deep Learning

1. Introduction

Image denoising refers to the task of removing the measurement noise from an input image. It is not only of practical importance with the prevalence of photography

*Corresponding author
[1]Equal contribution

using mobile devices, but also serves as a key component in most image recovery tasks; see *e.g.* [1, 2]. Inspired by the great success of deep leaning in many computer vision applications, in recent years, there have been extensive studies on deep-learning-based image denoising methods. Most of these methods are built upon *convolutional neural networks* (CNNs). Such CNN-based approaches showed promising performance, provided that the training samples fit well the characteristics of test data, in terms of both image content and noise characteristics.

1.1. Motivations

All the existing CNN-based methods for image denoising are built upon real-valued CNNs. In recent years, complex-valued neural networks (NNs) have started to receive increasing attention. Many works suggest that using complex numbers in NNs could enhance the representational capacity [3] and lead to other advantages, *e.g.* easier optimization [4], better generalization performance [5], and noise-robust memory mechanisms [6]. In many recognition tasks, the performance of complex-valued NNs has been very competitive against that of their real-valued counterparts. However, to the best of our knowledge, complex-valued NNs have not been investigated for their potential applications in image processing, which contradicts the wide adoption of many well-known complex-valued transforms in image processing. To list some, discrete Fourier transform, Gabor transform, and dual-tree complex wavelet transform.

Indeed, complex-valued transforms have deep connections to biological vision and visual perception. It is known that the primate's area V1 (visual cortex) is dominated by complex cells (see *e.g.*[7]), *i.e.*, the cells whose responses are characterized by the selectivity to orientation and frequency. Most cells of area V4 were also found to be more similar to V1's complex cells rather than simple cells (see *e.g.*[8]). The receptive fields and responses of complex cells are usually modeled by Gabor wavelets [9]. Furthermore, the so-called phase introduced by the complex-valued representation dominates the perception of visual scenes [10]. Recall that a signal $\boldsymbol{f}$ under a complex-valued transform, denoted by $\mathcal{F}(\boldsymbol{f})$, can be interpreted in terms to two quantities: magnitude $|\mathcal{F}(\boldsymbol{f})|$ and phase $\phi(\boldsymbol{f})$:

$$\mathcal{F}(\boldsymbol{f}) = |\mathcal{F}(\boldsymbol{f})|e^{i\phi(\boldsymbol{f})}. \tag{1}$$

It is also known in computer vision, the phase of an image provides sufficient information of objects on shapes, edges and orientations. It is sufficient to recover most information of an image only using the phase of the Fourier transform of this image [11]. The benefits of complex-valued transforms motivated us to investigate the potentials of complex-valued NNs for image recovery, and this paper focuses on one core problem: image denoising.

1.2. Merits of complex-valued representation

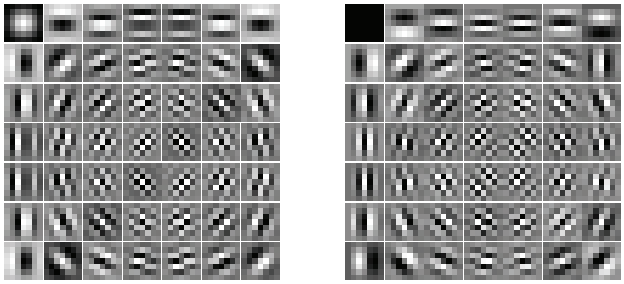
The main mathematical operations to build a CNN for image recovery include convolution, activation and residual learning. In the next, we discuss some merits of the complex-valued versions of these operations for image denoising.

Convolution. A 2D complex-valued filter has its special structure which is different from its real-valued counterpart. The filters with orientation selectivity are usually preferred in image processing, as local image edges are oriented in different directions. These filters are usually non-separable such that they cannot be expressed as the tensor product of two 1D real-valued filters, except the ones with the horizontal/vertical orientation. In contrast, the tensor product of two 1D complex-valued filters, denoted by $\boldsymbol{a}_1 + i\boldsymbol{b}_1$ and $\boldsymbol{a}_2 + i\boldsymbol{b}_2$, is not separable regarding its real part and imaginary part:

$$(\boldsymbol{a}_1 + i\boldsymbol{b}_1)(\boldsymbol{a}_2 + \boldsymbol{b}_2 i)^\top = (\boldsymbol{a}_1\boldsymbol{a}_2^\top - \boldsymbol{b}_1\boldsymbol{b}_2^\top) + i \cdot (\boldsymbol{a}_1\boldsymbol{b}_2^\top + \boldsymbol{b}_1\boldsymbol{a}_2^\top). \tag{2}$$

In other words, complex numbers allow using the tensor product of two 1D complex-valued filters to simulate 2D non-separable filters with different orientations. See Figure 1 for an illustration using 1D Gabor filters. Such a property leads to a more compact form of 2D non-separable filters with fewer freedoms. More specifically, the tensor product of two 1D complex-valued filters defined in $\mathbb{C}^L$, will lead to two 2D real-valued filters defined in $\mathbb{R}^{L\times L}$: one is from the real part, and the other is from the imaginary part. Thus, complex numbers enable using $4L$ freedoms to generate two 2D non-separable filters defined in $\mathbb{R}^{L\times L}$, which needs $2L^2$ freedoms when using real numbers. As a result, complex numbers allow a more compact representation for the operation of 2D convolution, which helps avoiding overfitting. This could be important when designing CNNs for image denoising. In image denoising, the training samples

include both truth images and their noisy versions. Although the set of truth images can be sufficient for training, their noisy counterparts could be insufficient, especially when the distribution of noise is complex or is spatially-varying. The amount of noisy data could be overwhelming in order to train a CNN with good generalizability.



(a) Real part (b) Imaginary part

Figure 1: Complex-valued 2D filters generated by the tensor products of all pairs of seven 1D Gabor filters in $\mathbb{C}^7$.

Activation function. There are many candidates of activation functions for complex-valued NNs. One is the generalization of the widely-used Rectified Linear Unit (ReLU) function from the real domain to the complex domain, which is defined as $\mathbb{C}$ReLU [12]:

$$\mathbb{C}\mathrm{ReLU}(z) = \mathrm{ReLU}(\Re(z)) + i \cdot \mathrm{ReLU}(\Im(z)), \tag{3}$$

where $\Re(\cdot)$ and $\Im(\cdot)$ denote the real part and imaginary part respectively. Recall that one key quantity introduced by the complex-valued representation is the phase, which the real-valued representation lacks. Consider a complex number in Figure 2. The $\mathbb{C}$ReLU not only has the same activation mechanism as the real-valued ReLU in terms of magnitude, but also has a quite complicated non-linear activation on the phase of the input. The concatenation of such non-linear operations on phase enables the definition of very sophisticated mappings in the phase domain. In other words, complex-valued NN allows defining the mapping between noisy images and noise-free images in both the amplitude domain and the phase domain.

Residual learning. The complex-valued representation is also related to the residual learning of NNs. It is shown in [6] that using complex numbers in the memory units could facilitate noise-robust retrieval mechanisms on the associative memory. In fact,

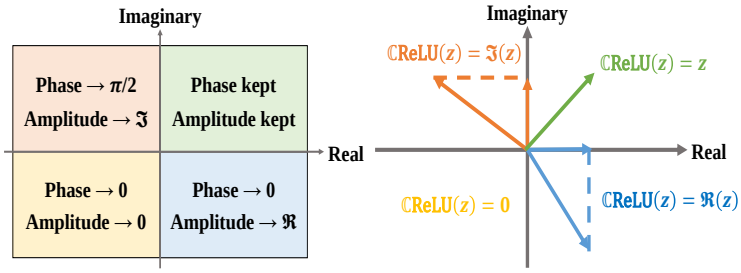


Figure 2: Activation on the amplitude and the phase of the input using ℂReLU.

the widely-used residual block [13] shares a similar architecture with the memory unit, in the sense that in each block the residual is computed and inserted into the "memory" provided by the identity connection [6]. Such a similarity implies the possible better robustness of the complex-valued residual block over its real-valued counterpart. It also implies the potential of complex-valued representation in residual learning for better robustness to noise model inconsistencies, *i.e.*, the noise characteristics of training samples are different from that of test images.

1.3. Contributions and significance

The contributions of our paper are three-fold:

- **First work that studies complex-valued CNN for image denoising**. In the past years, real-valued CNNs are the prominent choices of designing deep-learning-based methods for image recovery. In contrast, complex-valued representations and transforms are also widely used in image processing, including Gabor transform and dual-tree complex wavelet transform. The research in biological vision also showed the connections between complex-valued transforms and low-level processing in visual perception. This paper is the first one that investigates the potential of complex-valued CNN for low-level vision tasks, such as image denoising. Our study showed that the complex-valued CNN has its merits for image denoising.

- **New design of complex-valued essential mathematical operations involved in a denoising CNN**. It is known that the success of NNs to solve a problem lies

in both the careful design of NN architecture and the appropriate choice of essential operations. In this paper, we developed a complex-valued CNN for image denoising, as well as defined several basic operations in the complex number field to exploit possible advantages over their counterparts in the real number field. Namely, (i) a compact representation of 2D filters via the tensor product of 1D complex-valued filters, which is helpful to avoid overfitting; (ii) non-linear activation on phase, which helps improving denoising performance; and (iii) residual blocks with better robustness to the noise model inconsistencies that often occur in practice.

- **A practical denoising CNN with state-of-the-art performance and good robustness to noise model inconsistencies**. The experimental results on standard datasets show that our complex-valued CNN offers an alternative approach to designing effective denoising CNNs. The proposed complex-valued CNN showed competitive performance to the state-of-the-art real-valued denoising CNNs in the setting where the noise model of training samples exactly fits that of test images. Moreover, it has its advantages over other methods on the robustness to noise model inconsistencies, including the case where the noise levels of test images are unknown, where the standard deviation of noise varies among different pixels, and where the noise is a mixture of different types of noise.

1.4. Organization of The Work

The remaining part of the paper is organized as follows. In section 2, we give a literature review on related image denoising methods and existing complex-valued NNs designed for different applications. The proposed complex-valued denoising CNN is presented in Section 3 with all details. In Section 4, the experiments are conducted for the evaluation of the proposed method and the comparison to other closely-related methods. Section 5 concludes the paper.

2. Related Work

2.1. Image denoising

There is abundant literature on image denoising. This section focuses more on the discussion on the deep-learning-based approaches. The very early approaches modeled image denoising as either a filtering problem or a diffusion process; see [14] for a comprehensive survey. In the last two decades, sparsity-based regularizations have become one preferred choice for image denoising, which regularize a noise-free image by exploiting the sparsity prior of image under certain transforms, such as complex-valued ridgelet transform [15], wavelet [16] and adaptive dictionaries [17, 18, 19]. Another prominent approach is non-local methods (*e.g.* [20, 21, 22, 23]), which are based on the patch recurrence prior of natural images. The BM3D method [20] is arguably the most popular one which applies collaborative filtering to similar patches. The WNNM [24] and TWSC [25] are another two popular non-local methods that exploit the low rank structures of similar patches for denoising.

Instead of using the sparsity prior or the patch recurrence prior, some approaches learn image priors from visual data. Portilla *et al.* [26] proposed to learn a Gaussian scale mixture model on the wavelet coefficients of natural images. Roth *et al.* [27] proposed to learn a high-order Markov random field for modeling natural images. The classic EPLL approach [28] learns a Gaussian mixture model of image patches. Xu *et al.* [29] proposed to learn the distribution prior on similar patch groups. Instead of learning image priors, an alternative approach is to directly learn the denoising process. Schmidt *et al.* [30] unfolded the variational model of image denoising into a process with learnable parameters of filters and of shrinkage. Chen *et al.* [31] turned the diffusion process to a trainable one. Such approaches indeed can be viewed as training an NN for denoising.

Recently, many NN-based image denoisers have been proposed [1]. Jain *et al.* [32] trained a shallow CNN for denoising. Burger *et al.* [33] trained a multi-layer perceptron to denoise image patches. Agostinelli [34] trained a denoising auto-encoder for removing different types of noises. Vemulapalli *et al.* [35] unfolded the Gaussian conditional random field to a deep NN with automatic estimation of noise variance.

Zhang *et al.* [36] proposed a deep CNN called DnCNN, with residual learning for blind denoising. The DnCNN is trained to map noisy images to the noise, which helps the robustness of the NN to different noise levels. Nowadays, the DnCNN has become the benchmark method for evaluating CNN-based denoisers. Zhang *et al.* [37] further extended their work to deal with spatially-varying noise, which is done by combining a tunable noise level map into the input of the CNN. To obtain more training images for blind denoising, Chen *et al.* [38] trained a generative adversarial network (GAN) that estimates the noise distribution over the input noisy images and generates noisy image samples as additional training data. Lefkimmiatis *et al.* [39] inserted non-local filtering layers into the CNN to exploit the inherent patch recurrence of natural images. All the above CNNs are real-valued CNNs.

2.2. Complex-valued convolutional networks

The early works on complex-valued NNs mainly focus on addressing the basics of learning; see [40, 5, 3] for more details. In recent years, there have been extensive studies on complex-valued CNNs. Oyallon and Mallat [41] constructed a learning-free CNN with well-designed complex-valued wavelet filters. The resulting complex-valued CNN has its mathematical treatment, but with limited adaptivity as it is not learnable. Then, some mathematical understandings were presented in [42] for a trainable complex-valued CNN. The practical techniques for building trainable complex-valued CNNs were comprehensively studied and discussed in [3, 12]. For gaining certain invariance, several complex-valued CNNs were developed. Chintala *et al.* [43] proposed a complex-valued CNN with scale invariance. A similar architecture was proposed in [44]. Worrall *et al.* [45] replaced regular CNN filters with circular harmonics for the invariance to complicated rotations. We note that the applications of above complex-valued CNNs all focus on recognition tasks.

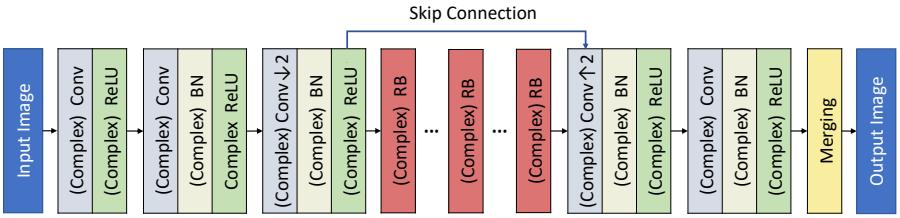


Figure 3: Diagram of framework of ℂDNet. Abbreviations: Conv for Convolution, ↓ 2 for downsampling by stride 2, BN for Batch Normalization, RB for Residual Block, and ReLU for Rectified Linear Unit.

3. Proposed Method

3.1. Framework

The complex-valued CNN proposed in this paper is called ℂDNet (*Complex-valued Denoising Network*). The ℂDNet maps a noisy image $\boldsymbol{Y}$ to a noise-free image $\boldsymbol{X}$:

$$\mathbb{C}\text{DNet} : \boldsymbol{Y} \in \mathbb{R}^{M_1 \times M_2} \rightarrow \boldsymbol{X} \in \mathbb{R}^{M_1 \times M_2}. \tag{4}$$

See Figure 3 for the outline of ℂDNet. Briefly, the input image is passed to 24 sequentially-connected convolutional units. Each convolutional unit except the first one contains a complex-valued convolutional layer which is sequentially followed by the complex-valued batch normalization and the complex-valued ReLU. All the convolutional layers use 64 convolutional kernels. The middle 18 convolutional units are implemented as 9 residual blocks [13] equipped with complex-valued representations for better performance and faster convergence [13]. To enlarge the receptive field for further improvement and better computational efficiency, we adopt a convolutional/deconvolutional layer with stride 2 in the convolutional units before the first residual block and after the last residual block for the function of down-scaling/upscaling the feature maps. Finally, a merging layer is employed to transform the complex-valued features from the previous convolutional unit to a real-valued image. We also use the skip connection to connect the input of the first residual block with the output of the last residual block for preserving image details. It can be seen that there are mainly five basic blocks in ℂDNet: (i) complex-valued convolutional layer; (ii) complex-valued

ReLU; (iii) complex-valued batch normalization; (iv) complex-valued residual block; and (v) merging layer.

3.2. Complex-valued convolutional layer

The complex-valued convolutional layer is constructed by simply replacing the real-valued kernel by the complex-valued kernel in the convolution process. The layer takes a complex-valued feature cube $\boldsymbol{A}$ as input and outputs another complex-valued feature cube $\tilde{\boldsymbol{A}}$:

$$\text{Conv} : \boldsymbol{A} \in \mathbb{C}^{N_1 \times N_2 \times D_1} \rightarrow \tilde{\boldsymbol{A}} \in \mathbb{C}^{N_1' \times N_2' \times D_2}. \tag{5}$$

More specifically, the layer is composed of D_2 convolution operations with the complex-valued filters $\{\boldsymbol{K}_i \in \mathbb{C}^{L \times L \times D_1}\}_{i=1}^{D_2}$, which extends through the full depth of $\boldsymbol{A}$ (*i.e.* D_1). During the forward pass, each kernel $\boldsymbol{K}_i$ is convolved across the width and height of $\boldsymbol{A}$ as follows:

$$(\boldsymbol{A} * \boldsymbol{K}_i)(x, y) = \sum_{x_0, y_0, z_0} \boldsymbol{A}(x - x_0, y - y_0, z_0) \boldsymbol{K}_i(x_0, y_0, z_0), \tag{6}$$

which produces a 2-dimensional feature map regarding $\boldsymbol{K}_i$. Stacking the feature maps for all filters along the depth dimension forms the full output. In practice, we set $L = 3$ for all the convolutional layers.

In all residual blocks, the 2D convolution is implemented by two consecutive 1D convolutions. Concretely, we use the following scheme:

$$\boldsymbol{A} * \boldsymbol{K}_i \rightarrow \boldsymbol{A} * \boldsymbol{k}_i^1 * \boldsymbol{k}_i^2, \tag{7}$$

where $\boldsymbol{k}_i^1 \in \mathbb{C}^{L \times 1 \times D}, \boldsymbol{k}_i^2 \in \mathbb{C}^{1 \times L \times D}$. Such a factorization represents the 2D convolution in a more compact way, *i.e.*, the number of parameters of each convolutional layer is reduced from DL^2 to $2DL$, where D is number of channels.

Many existing CNN toolboxes do not support complex-valued convolutions. We implement the complex-valued convolution using the real-valued convolutions available in existing toolboxes. The complex-valued convolution can be expressed as

$$\begin{aligned} \boldsymbol{A} * \boldsymbol{K} = &(\Re(\boldsymbol{A}) * \Re(\boldsymbol{K}) - \Im(\boldsymbol{A}) * \Im(\boldsymbol{K})) \\ &+ (\Re(\boldsymbol{A}) * \Im(\boldsymbol{K}) + \Im(\boldsymbol{A}) * \Re(\boldsymbol{K}))i. \end{aligned} \tag{8}$$

It can be seen from (8), the complex-valued convolution can be implemented by four real-valued convolutions. Thus, each convolutional layer is the mapping of

$$\begin{pmatrix} \Re(\cdot) \in \mathbb{R}^{N_1 \times N_2 \times D} \\ \Im(\cdot) \in \mathbb{R}^{N_1 \times N_2 \times D} \end{pmatrix} \to \begin{pmatrix} \Re(\cdot) \in \mathbb{R}^{N_1 \times N_2 \times D} \\ \Im(\cdot) \in \mathbb{R}^{N_1 \times N_2 \times D} \end{pmatrix}, \tag{9}$$

which allows using existing real-number-based toolboxes. It is noted that such an implementation is similar to a double-width CNN. The main difference is that the convolution in complex number field introduces additional interactions between the real part and the imaginary part of a complex-valued feature. Such interactions can be implemented in a real-valued NN with additional connections, but no real-valued NNs will do such a connection without the motivations from complex-valued convolutions.

The back-propagation about the complex-valued convolution kernels is similar to that of their real-valued counterparts, except that the related operations are defined on complex numbers. More specifically, let $\boldsymbol{K}, \boldsymbol{A}$ denote a complex-valued kernel and an input complex-valued feature map respectively. Let $\boldsymbol{B} = \boldsymbol{A} * \boldsymbol{K}$ and $f(\boldsymbol{B})$ is a scalar function on $\boldsymbol{B}$. This sufficiently covers the calculation of the gradients encountered in the training of complex-valued CNNs. By the chain rule in complex analysis, we have

$$\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{K}} = \frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{B}} \frac{\partial \boldsymbol{B}}{\partial \boldsymbol{K}} = \frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{B}} * \boldsymbol{A}. \tag{10}$$

Note that $\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{B}}$ and $\boldsymbol{A}$ are both complex-valued, and thus $\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{K}}$ is also complex-valued with the form: $\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{K}} = \Re(\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{K}}) + i \cdot \Im(\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{K}})$. Based on (8) we have

$$\Re(\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{K}}) = \Re(\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{B}}) * \Re(\boldsymbol{A}) - \Im(\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{B}}) * \Im(\boldsymbol{A}), \tag{11}$$

$$\Im(\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{K}}) = \Re(\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{B}}) * \Im(\boldsymbol{A}) + \Im(\frac{\partial f(\boldsymbol{B})}{\partial \boldsymbol{B}}) * \Re(\boldsymbol{A}). \tag{12}$$

3.3. $\mathbb{C}$ReLU for complex numbers

The ReLU is arguably the prominent choice for the activation functions in CNNs for image recovery, which is also used in $\mathbb{C}$DNet. Same as the real-valued one, the complex-valued ReLU (CReLU) activation is an element-wise mapping denoted by

$$\mathrm{CReLU} : \mathbb{C}^{N_1 \times N_2 \times D} \to \mathbb{C}^{N_1 \times N_2 \times D}, \tag{13}$$

$$\mathrm{CReLU}(\boldsymbol{A})(k) = \mathrm{CReLU}(\boldsymbol{A}(k)). \tag{14}$$

There are many choices for these CReLUs. In this paper, we propose to use the $\mathbb{C}$ReLU [12] as the CReLU that enables sophisticated non-linear operations on the phase, which is defined by

$$\mathbb{C}\mathrm{ReLU}(\boldsymbol{z}) = \mathrm{ReLU}(\Re(\boldsymbol{z})) + i \cdot \mathrm{ReLU}(\Im(\boldsymbol{z})), \tag{15}$$

for a complex-valued vector $\boldsymbol{z}$. The $\mathbb{C}$ReLU applies the ReLU activation on the real part and imaginary part of the input respectively. It can be seen from Figure 2 that the $\mathbb{C}$ReLU allows four different patterns in the ring of phase. In addition to $\mathbb{C}$ReLU, there are also other options for CReLUs. One is the Modular ReLU (MoeReLU) [3] defined by

$$\mathrm{ModReLU}(z) = \begin{cases} (|z| + b)\frac{z}{|z|}, & \text{if } |z| + b \geq 0; \\ 0, & \text{if } |z| + b < 0, \end{cases} \tag{16}$$

where $b \in \mathbb{R}$ is a trainable bias parameter. The ModReLU can maintain the phase after the activation. It is implemented by feeding the magnitude to the real-valued ReLU, followed by the modulation with the original phase. Another choice is the zReLU [3] defined by

$$z\mathrm{ReLU}(z) = \begin{cases} z, & \text{if } \phi(z) \in [0, \frac{\pi}{2}]; \\ 0, & \text{otherwise}, \end{cases} \tag{17}$$

where $\phi(\cdot)$ denotes the phase. The zReLU is activated when both the real part and the imaginary part are positive. It is implemented by

$$z\mathrm{ReLU}(z) = \frac{\mathrm{ReLU}(\Re(z) \cdot \Im(z))}{\Im(z)} + i\frac{\mathrm{ReLU}(\Re(z) \cdot \Im(z))}{\Re(z)}. \tag{18}$$

It will be shown in the experiments that, with a simple yet effective nonlinear operation on phase, $\mathbb{C}$ReLU yields better results than ModReLU and zReLU.

3.4. Complex-valued batch normalization

Batch normalization is a commonly-used module for better generalization performance as well as better convergence of training, which is denoted by

$$\mathrm{BN} : \mathbb{C}^{N_1 \times N_2 \times D} \to \mathbb{C}^{N_1 \times N_2 \times D}. \tag{19}$$

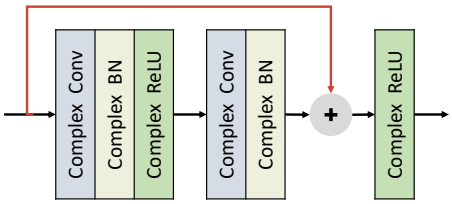


Figure 4: Diagram of complex-valued residual block. Abbreviations: Conv for Convolution, RB for Residual Block, and ReLU for Rectified Linear Unit.

We adapt batch normalization to the complex number field by separately running batch normalization on the real and imaginary parts of complex numbers respectively:

$$\mathrm{BN}(\boldsymbol{z}) = \mathrm{ReBN}(\Re(\boldsymbol{z})) + i \cdot \mathrm{ReBN}(\Im(\boldsymbol{z})) \tag{20}$$

where $\mathrm{ReBN}(\cdot)$ is the standard batch normalization with real-valued input.

3.5. Complex-valued residual block

Residual blocks [13] not only deal with the vanishing gradients in the back propagation during training, but also benefit the preservation of image details in denoising by passing the previous features to the subsequent layers. The generalization of residual blocks to the complex number field is straightforward. See Figure 4 for an illustration of the structure of the complex-valued residual block. It is noted that the residual block shares similarity with the memory unit, by regarding that the residual is computed and inserted into the "memory" provided by the identity connection [6]. It is shown in [6] that introducing complex numbers into the memory units could facilitate noise-robust retrieval mechanisms on the associative memory. Therefore, the extension of the residual block to complex number field is beneficial to the robustness of the CNN to noise model inconsistencies.

3.6. Merging layer

The merging layer transforms the complex-valued feature maps into the output real-valued image:

$$\mathrm{Merging} : \mathbb{C}^{M_1 \times M_2 \times D} \to \mathbb{R}^{M_1 \times M_2}. \tag{21}$$

Given a feature cube $\boldsymbol{A} \in \mathbb{C}^{M_1 \times M_2 \times D}$ as input, we first apply a convolution with a $L \times L \times D$ kernel to map $\boldsymbol{A}$ to a feature map $\boldsymbol{B} \in \mathbb{C}^{M_1 \times M_2}$. The conversion of $\boldsymbol{B}$ into a real-valued image $\boldsymbol{X} \in \mathbb{R}^{M_1 \times M_2}$ is done as follows:

$$\boldsymbol{X} = \sqrt{\Re(\boldsymbol{B})^2 + \Im(\boldsymbol{B})^2}. \tag{22}$$

In other words, the conversion is done by only taking the amplitude of a complex-valued signal.

3.7. Training

Given the training data $\{(\boldsymbol{Y}_k, \bar{\boldsymbol{X}}_k)\}_{k=1}^K$ where $\boldsymbol{Y}_k, \bar{\boldsymbol{X}}_k$ represent the kth noisy image (patch) and its truth respectively. Let $\boldsymbol{\theta}$ denote the parameter vector encoding all parameters of the $\mathbb{C}$DNet. The loss function for training is simply defined by the mean square error as follows:

$$\ell(\boldsymbol{\theta}) := \frac{1}{K} \sum_{k=1}^{K} \|\mathbb{C}\mathrm{DNet}(\boldsymbol{Y}_k; \boldsymbol{\theta}) - \bar{\boldsymbol{X}}_k\|_2^2. \tag{23}$$

The weights are initialized using Xavier, and the training loss is optimized by Adam.

4. Experiments

We applied the $\mathbb{C}$DNet to image denoising under different settings, including non-blind removal of additive white Gaussian noise (AWGN), blind AWGN removal, and removal of spatially-varying noises. The quantitative results reported in the following are the average values over multiple runs. The $\mathbb{C}$DNet is implemented using TensorFlow with CUDA acceleration. The experiments were carried out on a workstation with a 3.2GHz Intel Core i7-8700 CPU, 32G RAM and an NVIDIA Gefore RTX 2080Ti GPU. Throughout the experiments, all convolutional layers are with kernel sizes of 3×3 and zero padding of length 2. The number of channels is set to 64 for all convolutional layers except that it is the same as the number of channels of the input image for the last convolutional layer. As described in Section 3, there are totally 24 convolutional layers in $\mathbb{C}$DNet. It is worth mentioning that the depth and width parameters were simply adjusted by trying some common values used in the existing methods

such that the resulting model has similar size with the standard benchmark methods, *e.g.* DnCNN. We did not fine-tune the depth and width, as this is a very time-consuming task.

4.1. Non-blind removal of AWGN

4.1.1. Test methodology

In this setting, we aim at denoising the images degraded by the AWGN of known noise levels. Since the noise levels are known, the denoisers are separately trained and tested on different noise levels. Regarding the training of $\mathbb{C}$DNet, we follow [31, 36, 31] for fair comparison, which uses 400 BSD images [36] of size 180×180. Same as [36], the images are cut into patches of size 40×40 for data augmentation, and then 226800 of them are sampled and degraded by the AWGN for training. Both low and high noise levels are used for training, including $\sigma = 15, 25, 35, 50, 60, 70, 75, 80$. For each noise level, the $\mathbb{C}$DNet model is trained with 70 epochs and with the learning rate decaying from 1×10^{-2} to 1×10^{-3}. The training time takes around 9 hours on each noise level. For test, we select two widely-used benchmark datasets including Set12 [36] and BSD68 [27]. There is no overlap between the set of test images and the set of training images, and the content of test images has sufficient variations and is different from that of training images. All the images used in the experiments are gray-scale. With each of the above noise levels, we use the corresponding AWGN to corrupt the test images, and then the $\mathbb{C}$DNet trained on that level is used to denoise the noisy images. The peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) on the denoised images are used for quantifying the performance.

4.1.2. Results and comparison

We compare our $\mathbb{C}$DNet with both the classic methods and the state-of-the-art ones, including BM3D [20], WNNM [24], EPLL [28], TNRD [31], DnCNN [36], IR-CNN [2], SF-20L [46], UNLNet [39], TWSC [25] and FFDNet [37]. These methods cover different types of image denoisers. Among them, BM3D, WNNM, EPLL and TWSC are four representative traditional methods, while others are the recent deep-learning-based methods. The reported results of the compared methods, whenever

possible, are quoted from the published works. Or otherwise they are produced by the codes published by the original authors. The test and training (if possible) of these compared methods are done in the same manner as ours, which makes the comparison fair.

Table 1 and Table 2 show the PSNR values of all the compared methods with different noise levels on the Set12 and BSD68 datasets respectively. The corresponding SSIM values are given in Table 3 and Table 4. It can be seen that our $\mathbb{C}$DNet performs better than other compared methods on both light AWGN and heavy AWGN. We show some denoising results in Figure 5 and Figure 6 for visual comparison.

Table 1: Average PSNR(dB) of denoised images by different methods on Set12 in non-blind AWGN removal.

σ	15	25	35	50	60	70	75	80
BM3D	32.37	29.97	28.40	26.72	25.95	25.25	24.93	24.63
WNNM	32.69	30.25	28.69	27.05	26.20	25.50	25.19	24.91
EPLL	32.14	29.69	28.11	26.47	25.60	24.89	24.59	24.30
TNRD	32.50	30.05	n/a	26.81	n/a	n/a	n/a	n/a
IRCNN	32.77	30.38	28.80	27.14	n/a	n/a	n/a	n/a
DnCNN	32.85	30.43	28.82	27.17	26.27	25.67	25.33	25.01
TWSC	32.60	30.18	28.63	26.96	26.08	25.34	25.00	24.67
FFDNet	32.75	30.43	28.92	27.32	26.54	25.83	25.49	25.22
$\mathbb{C}$DNet	**32.87**	**30.53**	**28.99**	**27.38**	**26.58**	**25.89**	**25.58**	**25.30**

4.2. Blind AWGN removal

4.2.1. Test methodology

In practice, the blind AWGN removal (*i.e.* denoising without knowing the noise level) is more valuable. We evaluate our $\mathbb{C}$DNet on the blind AWGN removal on the previously-used Set12 and BSD68 datasets. The noisy images are generated as follows. Given a clean image, the noise level σ is randomly picked up from $[5, 80]$. Then the AWGN with σ is added to the image. For the training of $\mathbb{C}$DNet for blind denoising, we follow the scheme used in [39], which divides the noise level $\sigma \in [5, 80]$ into three intervals: $[5, 30]$, $[30, 55]$ and $[55, 80]$. Then the $\mathbb{C}$DNet is trained on these intervals respectively. The aforementioned 400 BSD images are used for training. Similar to the

Table 2: Average PSNR(dB) of denoised images by different methods on BSD68 in non-blind AWGN removal.

σ	15	25	35	50	60	70	75	80
BM3D	31.07	28.57	27.08	25.62	25.07	24.52	24.28	24.05
WNNM	31.37	28.83	27.30	25.87	25.16	24.63	24.38	24.15
EPLL	31.21	28.68	27.21	25.67	25.01	24.43	24.18	23.95
TNRD	31.42	28.92	n/a	25.97	n/a	n/a	n/a	n/a
IRCNN	31.63	29.15	27.66	26.19	n/a	n/a	n/a	n/a
DnCNN	31.73	29.23	27.69	26.23	25.41	24.87	24.70	24.44
TWSC	31.28	28.76	27.25	25.77	25.04	24.44	24.17	23.93
SF-20L	31.29	28.82	n/a	26.02	n/a	n/a	24.43	n/a
FFDNet	31.63	29.19	27.73	26.29	25.62	25.05	24.79	24.55
$\mathbb{C}$DNet	**31.74**	**29.28**	**27.77**	**26.36**	**25.67**	**25.10**	**24.85**	**24.63**

Table 3: Average SSIM of denoised images by different methods on Set12 in non-blind AWGN removal.

σ	15	25	35	50	60	70	75	80
BM3D	.8963	.8509	.8111	.7661	.7377	.7088	.6980	.6860
WNNM	.8938	.8457	.8071	.7562	.7289	.7039	.6932	.6838
EPLL	.8938	.8457	.8071	.7562	.7289	.7039	.6932	.6838
IRCNN	.9008	.8601	.8256	.7804	n/a	n/a	n/a	n/a
DnCNN	.9027	.8618	.8259	.7827	.7369	.7026	.6842	.6710
TWSC	.8989	.8549	.8192	.7731	.7454	.7200	.7086	.6974
FFDNet	.9029	.8641	.8316	.7906	.7673	.7451	.7332	.7214
$\mathbb{C}$DNet	**.9034**	**.8646**	**.8328**	**.7924**	**.7696**	**.7494**	**.7393**	**.7308**

Table 4: Average SSIM of denoised images by different methods on BSD68 in non-blind AWGN removal.

σ	15	25	35	50	60	70	75	80
BM3D	.8744	.8044	.7511	.6931	.6627	.6363	.6248	.6137
WNNM	.8780	.8100	.7553	.6984	.6660	.6458	.6333	.6213
EPLL	.8824	.8120	.7558	.6915	.6581	.6307	.6181	.6073
IRCNN	.8881	.8249	.7746	.7171	n/a	n/a	n/a	n/a
DnCNN	.8906	.8278	.7765	.7189	.6422	.5998	.5785	.5657
TWSC	.8782	.8077	.7530	.6903	.6573	.6293	.6168	.6050
FFDNet	.8902	.8295	.7815	.7261	.6959	.6697	.6579	.6443
$\mathbb{C}$DNet	**.8916**	**.8314**	**.7833**	**.7272**	**.6974**	**.6737**	**.6619**	**.6518**

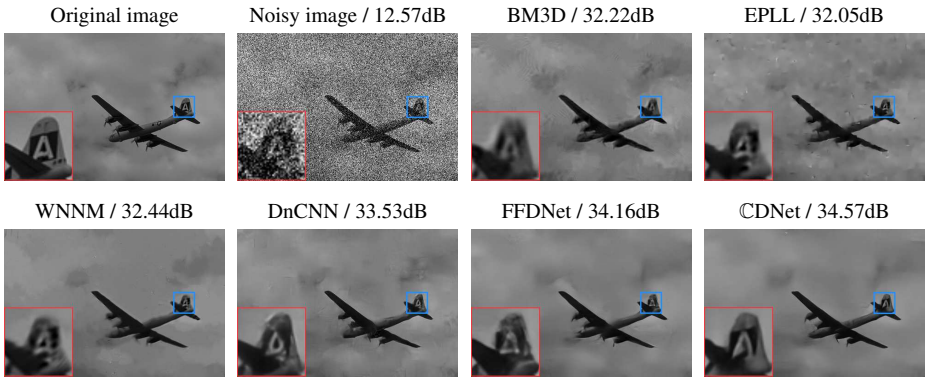


Figure 5: Denoising results of an image from BSD68 dataset in non-blind AWGN removal with noise level $\sigma = 60$.

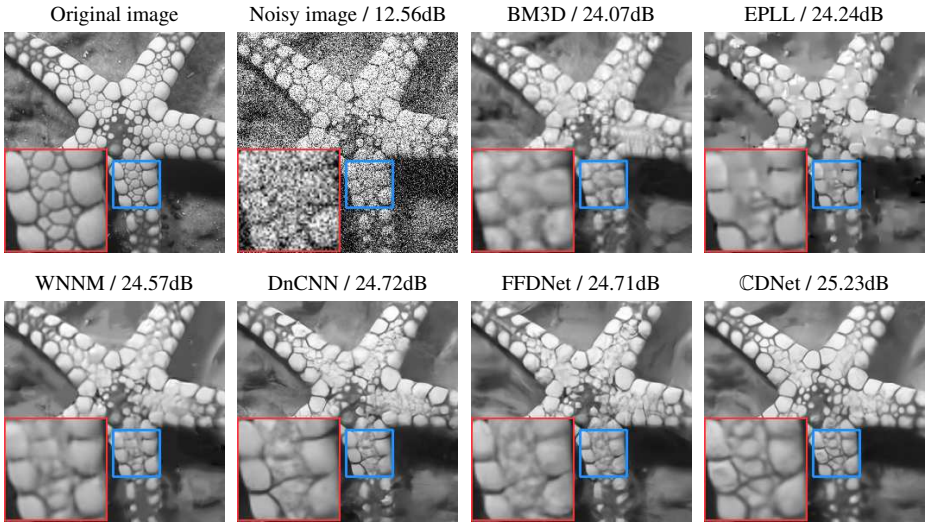


Figure 6: Denoising results of image "starfish" in non-blind AWGN removal with noise level $\sigma = 60$.

previous experiment, the images are cut into 40×40 patches for data augmentation, and then 226800 of them are sampled and degraded by the AWGN with the randomly-chosen levels in the interval for training. The $\mathbb{C}$DNet model is trained with 140 epochs and with the learning rate decaying from 1×10^{-2} to 1×10^{-4}. The training takes around 15 hours. In test, we generate the images corrupted by the AWGN with $\sigma = 15, 25, 35, 50, 60, 70, 75, 80$ respectively and report the results for each noise level.

In addition to gray-scale images, we also train our model for denoising color images with known noise levels. We use a color version of the Berkeley segmentation dataset, of which 432 color images are used for training and the remaining 68 images are used to form the test set. Similar to the gray image denoising, the color images are cut into patches of size 40×40 for training. The model are trained at three different noise levels, including $\sigma = 25, 35, 50$, and the number of epoch per training is fixed at 51. We compare our $\mathbb{C}$DNet with CBM3D (BM3D for color images) and DnCNN [36].

4.2.2. Results and comparison

Results on gray-scale image denoising. We compare our $\mathbb{C}$DNet with two CNN-based denoisers that support blind denoising, including DnCNN [36] and UNLNet [39]. The BM3D [20] is also used for comparison. The test and training (if possible) of these compared methods are also done in the same blind manner as ours. The PSNR results are summarized in Table 5 and Table 6, and the SSIM results are given in Table 7 and Table 8. The results of the compared methods are quoted from [39] whenever available, or otherwise produced by the published codes from the authors of the original works. Note that UNLNet has no available results on Set12 and no published code either, we only compare it on BSD68. It can be seen that our method is powerful in the task of blind AWGN removal. On both datasets, our $\mathbb{C}$DNet achieved the best results among all the compared methods. We visualize some denoising results in Figure 7 and Figure 8. Compared to other methods, $\mathbb{C}$DNet can better handle both the textured regions and smooth regions. The improvement of $\mathbb{C}$DNet over other compared methods has demonstrated benefits of using complex numbers in denoising CNN for better generalization to the processing of unknown noise level which is an often-seen type of noise model inconsistencies. It can also be observed that the PSNR improvement of $\mathbb{C}$DNet

tends to be larger as the noise level increases. The reason is probably that higher unknown noise levels may cause worse inconsistencies of noise models and our ℂDNet has better robustness to those inconsistencies.

Results on color image denoising. Table 9 shows the PSNR and SSIM values of all the compared methods with different noise levels on the color version of BSD68 datasets respectively. It can be seen that our ℂDNet is the top performer among all compared methods. We show some denoising results in Figure 9 for visual comparison.

Table 5: Average PSNR(dB) of denoised images by different methods on Set12 in blind AWGN removal.

σ	15	25	35	50	60	70	75	80
BM3D	32.37	29.97	28.40	26.72	25.95	25.25	24.93	24.63
DnCNN	32.71	30.29	28.69	27.11	26.25	25.53	25.22	24.85
ℂDNet	**32.79**	**30.47**	**28.71**	**27.34**	**26.46**	**25.87**	**25.55**	**25.29**

Table 6: Average PSNR(dB) of denoised images by different methods on BSD68 in blind AWGN removal.

σ	15	25	35	50	60	70	75	80
BM3D	31.07	28.57	27.08	25.62	25.07	24.52	24.28	24.05
UNLNet	31.47	28.96	27.50	26.04	n/a	n/a	n/a	n/a
DnCNN	31.61	29.11	27.54	26.17	25.44	24.85	24.61	24.32
ℂDNet	**31.64**	**29.18**	**27.61**	**26.27**	**25.56**	**25.05**	**24.78**	**24.59**

Table 7: Average SSIM of denoised images by different methods on Set12 in blind AWGN removal.

σ	15	25	35	50	60	70	75	80
BM3D	.8963	.8509	.8111	.7661	.7377	.7088	.6980	.6860
DnCNN	.8970	.8499	.8191	.7616	.7417	.7039	.6848	.6671
ℂDNet	**.9022**	**.8616**	**.8211**	**.7894**	**.7634**	**.7467**	**.0751**	**.7272**

4.3. Removal of spatially-varying noises

4.3.1. Test methodology

We further evaluate the performance of our ℂDNet on blindly removing spatially-varying noises. We use a similar setting to [38], with two types of spatially-varying

Table 8: Average SSIM of denoised images by different methods on BSD68 in blind AWGN removal.

σ	15	25	35	50	60	70	75	80
BM3D	.8744	.8044	.7511	.6931	.6627	.6363	.6248	.6137
DnCNN	.8803	.7977	.7611	.6808	.6616	.6276	.6085	.5894
ℂDNet	**.8911**	**.8228**	**.7764**	**.7140**	**.6956**	**.6627**	**.6476**	**.6411**

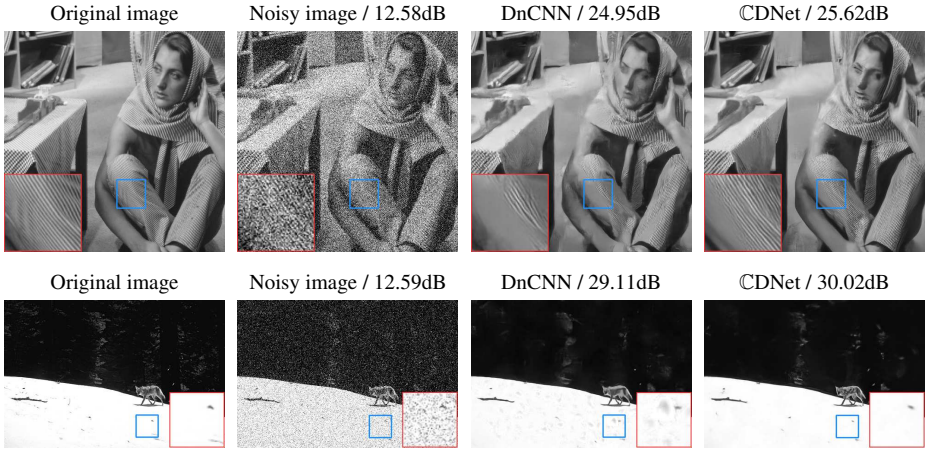


Figure 7: Denoising results on two noisy images in blind AWGN removal with noise level $\sigma = 60$.

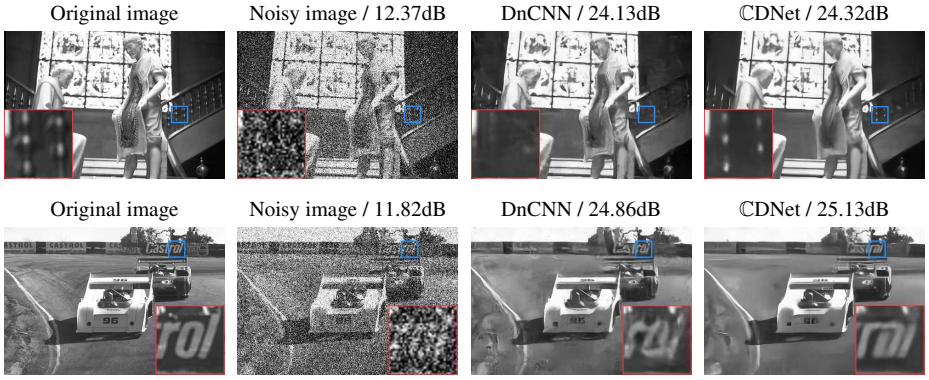


Figure 8: Denoising results on two noisy images in blind AWGN removal with noise level $\sigma = 75$.

Table 9: Average PSNR(dB) and SSIM results of color image denoising by different methods on BSD68 in non-blind AWGN removal.

Method	PSNR			SSIM		
	25	35	50	25	35	50
CBM3D	30.64	28.83	27.31	0.931	0.901	0.870
DnCNN	31.31	29.65	28.01	0.884	0.844	0.792
ℂDNet	**31.34**	**29.84**	**28.14**	**0.937**	**0.915**	**0.882**

Figure 9: Denoising results on a color image with noise level $\sigma = 50$.

noises considered. The first is the AWGN with spatially-varying high noise levels. We use two settings: (i) 70% pixels corrupted by $\mathcal{N}(0, 60)$ and 30% pixels corrupted by $\mathcal{N}(0, 75)$; and (ii) 50% pixels corrupted by $\mathcal{N}(0, 60)$, 35% pixels corrupted by $\mathcal{N}(0, 70)$ and 15% pixels corrupted by $\mathcal{N}(0, 80)$. The second type is the spatially-varying light AWGN/uniform noise. Each pixel is either degraded with the AWGN of a small variance, or the uniform noise in the range of $[-s, s]$. We fix the AWGN to be $\mathcal{N}(0, 1)$ on 20% pixels and $\mathcal{N}(0, 0.02)$ on 70% pixels and set $s = 5, 10, 15$.

In the blind setting, it is complicated to generate the training images that contain the above spatially-mixed noises of all possible combinations. Therefore, we do not re-train our model but instead directly use the one trained for blind AWGN removal during the test. This indeed tests the generalizability and transferability of a trained CNN to the processing of other noise models.

4.3.2. Results and comparison

For comparison, we select BM3D [20], WNNM [24], EPLL [28] and DnCNN [36]. The former three are the training-free methods and we run their published codes with their parameters finely tuned. Regarding DnCNN, for fair comparison with ours, we also use its pre-trained model in the blind AWGN removal for the test. Table 10 summarizes the PSNR results on the removal of spatially-varying noises. Some denoising results are shown in Figure 10 and Figure 11. Both the PSNR results and visual results have demonstrated the superior performance of $\mathbb{C}$DNet over other approaches when generalized for handling the noise with different characteristic from that of training samples. Such improved generalizability comes from the better robustness of $\mathbb{C}$DNet to noise model inconsistencies.

Table 10: Average PSNR(dB) of denoised images on Set12 and BSD68 datasets by different methods in removing spatially-varying noises.

		Mixed AWGN		AWGN + Uniform		
	Method	Setting#1	Setting#2	$s = 5$	$s = 10$	$s = 15$
SET12	BM3D	25.55	25.46	40.48	40.2	37.32
	EPLL	25.16	25.05	41.24	40.82	37.89
	WNNM	24.85	24.74	40.21	39.22	36.19
	DnCNN	25.91	25.77	41.14	40.25	39.51
	$\mathbb{C}$DNet	**26.15**	**26.04**	**44.13**	**42.52**	**40.66**
BSD68	BM3D	24.79	24.68	41.21	40.68	36.72
	EPLL	24.68	24.58	42.55	41.74	37.73
	WNNM	24.14	24.06	41.02	40.05	35.73
	DnCNN	25.09	24.99	41.89	40.94	39.72
	$\mathbb{C}$DNet	**25.31**	**25.22**	**45.78**	**43.21**	**40.74**

4.4. More Discussions

4.4.1. Effectiveness of 1D convolution

Recall that in $\mathbb{C}$DNet the convolutional layers of residual blocks are built upon 1D complex-valued convolutions. We replace such convolutional layers with the ones that directly use 2D complex-valued convolutions and then test the performance of the

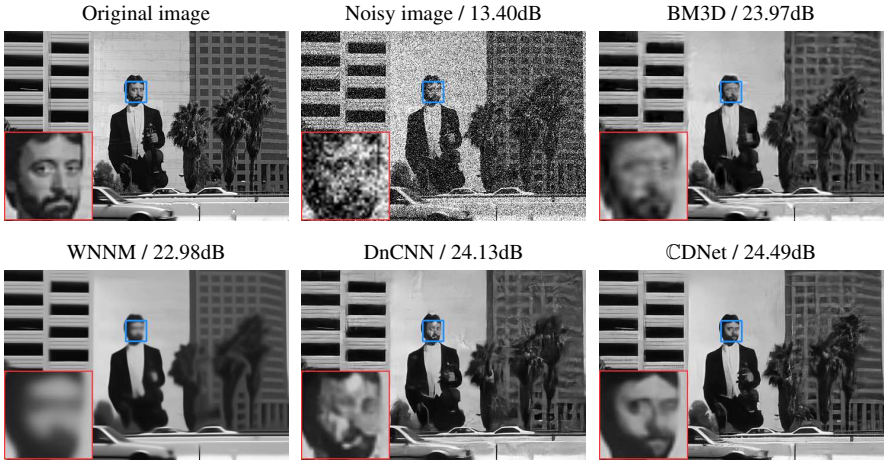


Figure 10: Denoising results of an image from BSD68 dataset in spatially-varying AWGN removal with Setting 1.

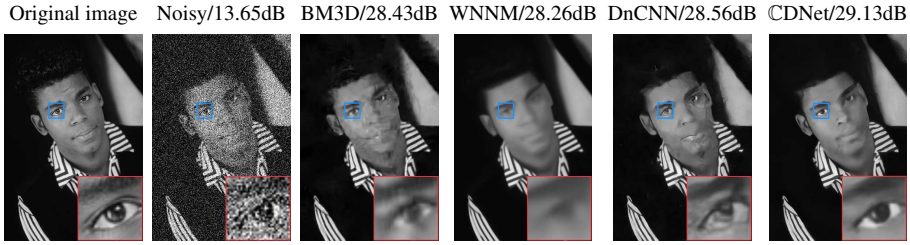


Figure 11: Denoising results of an image from BSD68 dataset in spatially-varying AWGN removal with Setting 2.

modified $\mathbb{C}$DNet in nonblind AWGN removal. On all the results the PSNR changes are bounded in $[-0.06\text{dB}, 0.06\text{dB}]$. In addition, the original $\mathbb{C}$DNet outperforms the modified one on more than half the results. In other words, the complex-valued CNN allows using compact 1D convolutions which have comparable expressibility and even better generalization performance than the 2D ones. See Table 11 for some results.

4.4.2. ReLU selection

The definition of the CReLU is not unique. Recall that there are another two choices: ModReLU defined by (16) and zReLU defined by (17). We are interested in how these CReLUs perform in denoising. Thus, we replace all $\mathbb{C}$ReLUs in $\mathbb{C}$DNet with the ModReLUs and zReLUs respectively, and re-conduct the denoising experiments in blind AWGN removal.

See Table 11 for some results. The ModReLU performed worse that $\mathbb{C}$ReLU with 0.15dB-0.3dB PSNR gap. The reason is probably that the ModReLU keeps the phase unchanged, which limits the expressibility of the complex-valued CNN for denoising. Note that the phase indeed encodes the main image structures and may be corrupted by noises, which should be deliberately treated in denoising. The zReLU performed even much worse than $\mathbb{C}$ReLU. We note the expressibility of zReLU is not as good as $\mathbb{C}$ReLU, considering that zReLU generates only two different patterns in the ring of phase, with limited operations on the phase. Recall from Figure 2 that $\mathbb{C}$ReLU generates four different patterns with richer phase operations. Another disadvantage of zReLU is that its implementation is more complicated than the other two ReLUs, which may increase the difficulty in optimizing the resulting loss.

Table 11: Average PSNR(dB) by $\mathbb{C}$DNet with different modifications in blind AWGN denoising with noise level $\sigma = 60$. 'Original': $\mathbb{C}$DNet without modifications; '2D Conv': replace 1D convolutions with 2D ones in residual blocks; 'ModReLU': replace $\mathbb{C}$ReLU with ModReLU; 'zRELU': replace $\mathbb{C}$ReLU with zReLU; Real: replace all complex-valued units with real-valued ones and with double number of channels.

Dataset	Original	2D Conv	zReLU	ModReLU	Real
Set12	**26.46**	26.43	22.53	26.23	26.11
BSD68	**25.56**	25.55	22.69	25.40	25.28

4.4.3. *Benefits of complex-valued architecture*

We evaluate the benefits of using complex numbers in denoising CNN by comparing it to a real-valued version of $\mathbb{C}$DNet. The real-valued version is constructed by replacing all complex-valued units in $\mathbb{C}$DNet with the real-valued ones. The number of channels in each convolution is doubled for fairness. The comparison is done on the blind AWGN removal with $\sigma = 60$, in which the PSNR result of the real-valued version is 0.35dB/0.28dB less than the original $\mathbb{C}$DNet on the Set12/BSD68 dataset. See Table 11 for the results and Figure 12 for some visual comparison. While a larger real-valued model can gain better expressibility, the side-effect is possible overfitting. In contrast, complex-valued NNs implicitly impose additional regularizations on the convolution processes, which is helpful for alleviating the overfitting. In other words, $\mathbb{C}$DNet is not a simple double-dimension real-valued CNN; it has its own specific characteristics. Such characteristics can lead to better denoising results over its real-valued counterpart. We also evaluated the running time. The results show that $\mathbb{C}$DNet is 1.4 times slower than its real-valued counterpart.

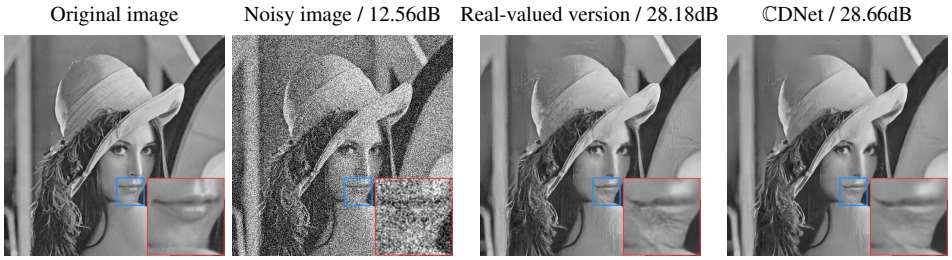


Figure 12: Denoising results of image "Lena" in blind AWGN removal with noise level $\sigma = 60$.

5. Conclusion

In this paper, we proposed the $\mathbb{C}$DNet, a complex-valued CNN for image denoising. Introducing complex-valued essential operations to the CNN-based denoiser has several merits: compact form of 2D non-separable convolution, non-linear activation on phase, and better noise robustness of residual blocks. By exploiting these merits in the proposed $\mathbb{C}$DNet, the $\mathbb{C}$DNet showed its good performance on non-blind AWGN

removal, as well as its advantages on blind AWGN removal and blind removal of noise with spatially-varying standard deviations.

In the past, many studies have shown that complex-valued CNNs can benefit high-level vision tasks such as image recognition, but none has been conducted to investigate their potentials in low-level vision tasks. Our work is the first one that showed the potential of the complex-valued CNN in a fundamental low-level task, *i.e.* image denoising. The results in this paper provide strong inspirations to the development of complex-valued CNNs for other low-level vision tasks. Though our method was only tested on real-valued images, with small modifications it can be directly applied to processing complex-valued signals.

In future, we would like to extend the proposed $\mathbb{C}$DNet to solving other image recovery problems, especially the ones involving complex-valued images. In addition, we would like to further refine the architecture and operations of the $\mathbb{C}$DNet to have more performance gain in image recovery. One possible direction is designing other non-linear activation functions on phase and introducing convolution-based operations on phase. The design of such functions and operations is a challenging task. Recall that the direct calculation of phase is not numerically stable when its corresponding value is small and is wrapped to $[0, 2\pi]$ (or $[-\pi, \pi]$) for resolving the periodicity ambiguity of phase. As a result, the NN will suffer from possible instability of back propagation and related computational issues for gradient-descend-based training. Thus, it is a better option to design the activation functions and operations on phase without explicitly calling phase as input. As phase has clear physical meaning, it is highly non-trivial to design such functions and operations with strong physical motivations.

Acknowledgments

This work is supported by National Natural Science Foundation of China (61872151, U1611461), Natural Science Foundation of Guangdong Province (2020A1515011128), Science and Technology Program of Guangzhou (201802010055). Ji Hui also would like to acknowledge the support of Singapore MOE AcRF (MOE2017-T2-2-156).

References

[1] I. Hong, Y. Hwang, D. Kim, Efficient deep learning of image denoising using patch complexity local divide and deep conquer, Pattern Recognition 96 (2019) 106945.

[2] K. Zhang, W. Zuo, S. Gu, L. Zhang, Learning deep cnn denoiser prior for image restoration, in: Proc. IEEE Conf. Comput. Vision Pattern Recognition, Vol. 2, 2017.

[3] N. Guberman, On complex valued convolutional neural networks, arXiv preprint arXiv:1602.09046 (2016).

[4] T. Nitta, On the critical points of the complex-valued neural network, in: Proc. Int. Conf. Neural Info. Process., Vol. 3, IEEE, 2002, pp. 1099–1103.

[5] A. Hirose, S. Yoshida, Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence, IEEE Trans. Neural Netw. Learning Syst. 23 (4) (2012) 541–551.

[6] I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, A. Graves, Associative long short-term memory, arXiv preprint arXiv:1602.03032 (2016).

[7] B. M. Dow, et al., Functional classes of cells and their laminar distribution in monkey visual cortex, J. Neurophysiol 37 (1974) 927–946.

[8] J. L. Gallant, J. Braun, D. C. Van Essen, Selectivity for polar, hyperbolic, and cartesian gratings in macaque visual cortex, Science 259 (5091) (1993) 100–103.

[9] J. G. Daugman, Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters, J. Optical Society America A 2 (7) (1985) 1160–1169.

[10] I. E. Gordon, Theories of visual perception, Psychology press, 2004.

[11] A. V. Oppenheim, J. S. Lim, The importance of phase in signals, Proc. of the IEEE 69 (5) (1981) 529–541.

[12] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, C. J. Pal, Deep complex networks, in: Proc. Int. Conf. Learning Representations, 2018.
URL https://openreview.net/forum?id=H1T2hmZAb

[13] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conf. Comput. Vision Pattern Recognition, 2016, pp. 770–778.

[14] Y. Zhang, H. Cheng, J. Huang, X. Tang, An effective and objective criterion for evaluating the performance of denoising filters, Pattern Recognition 45 (7) (2012) 2743 – 2757.

[15] G. Chen, B. Kégl, Image denoising with complex ridgelets, Pattern Recognition 40 (2) (2007) 578 – 585.

[16] G. Chen, T. Bui, A. Krzyżak, Image denoising with neighbour dependency and customized wavelet and threshold, Pattern Recognition 38 (1) (2005) 115 – 124.

[17] Z. Hou, Adaptive singular value decomposition in wavelet domain for image denoising, Pattern Recognition 36 (8) (2003) 1747 – 1763.

[18] M. Elad, M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, IEEE Trans. Image Process. 15 (12) (2006) 3736–3745.

[19] J. Wang, M. Wang, X. Hu, S. Yan, Visual data denoising with a unified schatten-p norm and lq norm regularized principal component pursuit, Pattern Recognition 48 (10) (2015) 3135 – 3144.

[20] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, Image denoising by sparse 3-d transform-domain collaborative filtering, IEEE Trans. Image Process. 16 (8) (2007) 2080–2095.

[21] Z. Sun, S. Chen, L. Qiao, A general non-local denoising model using multi-kernel-induced measures, Pattern Recognition 47 (4) (2014) 1751 – 1763.

[22] Y. Quan, H. Ji, Z. Shen, Data-driven multi-scale non-local wavelet frame construction and image recovery, J. Sci.Comput. 63 (2) (2015) 307–329.

[23] H. Li, C. Y. Suen, A novel non-local means image denoising method based on grey theory, Pattern Recognition 49 (2016) 237 – 248.

[24] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, L. Zhang, Weighted nuclear norm minimization and its applications to low level vision, Int. J. Comput. Vision 121 (2) (2017) 183–208.

[25] J. Xu, L. Zhang, D. Zhang, A trilateral weighted sparse coding scheme for real-world image denoising, in: Proc. European Conf. Comput. Vision, 2018.

[26] J. Portilla, V. Strela, M. J. Wainwright, E. P. Simoncelli, Image denoising using scale mixtures of gaussians in the wavelet domain, IEEE Trans. Image Process. 12 (11) (2003) 1338–1351.

[27] S. Roth, M. J. Black, Fields of experts, Proc. Int. J. Comput. Vision 82 (2) (2009) 205.

[28] D. Zoran, Y. Weiss, From learning models of natural image patches to whole image restoration, in: Proc. IEEE Int. Conf. Comput. Vision, IEEE, 2011, pp. 479–486.

[29] J. Xu, L. Zhang, W. Zuo, D. Zhang, X. Feng, Patch group based nonlocal self-similarity prior learning for image denoising, in: Proc. IEEE Int. Conf. Comput. Vision, 2015, pp. 244–252.

[30] U. Schmidt, S. Roth, Shrinkage fields for effective image restoration, in: Proc. IEEE Conf. Comput. Vision Pattern Recognition, 2014, pp. 2774–2781.

[31] Y. Chen, T. Pock, Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2017) 1256–1272.

[32] V. Jain, S. Seung, Natural image denoising with convolutional networks, in: Advances in Neural Inform. Process. Syst., 2009, pp. 769–776.

[33] H. C. Burger, C. J. Schuler, S. Harmeling, Image denoising: Can plain neural networks compete with bm3d?, in: Proc. IEEE Conf. Comput. Vision Pattern Recognition, IEEE, 2012, pp. 2392–2399.

[34] F. Agostinelli, M. R. Anderson, H. Lee, Adaptive multi-column deep neural networks with application to robust image denoising, in: Advances in Neural Inform. Process. Syst., 2013, pp. 1493–1501.

[35] R. Vemulapalli, O. Tuzel, M.-Y. Liu, Deep gaussian conditional random field network: A model-based deep network for discriminative denoising, in: Proc. IEEE Conf. Comput. Vision Pattern Recognition, 2016, pp. 4801–4809.

[36] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising, IEEE Trans. Image Proc. 26 (7) (2017) 3142–3155.

[37] K. Zhang, W. Zuo, L. Zhang, Ffdnet: Toward a fast and flexible solution for cnn based image denoising, IEEE Trans. Image Proc. (2018).

[38] J. Chen, J. Chen, H. Chao, M. Yang, Image blind denoising with generative adversarial network based noise modeling, in: Proc. IEEE Conf. Comput. Vision Pattern Recognition, 2018, pp. 3155–3164.

[39] S. Lefkimmiatis, Universal denoising networks: A novel cnn architecture for image denoising, in: Proc. IEEE Conf. Comput. Vision Pattern Recognition, 2018, pp. 3204–3213.

[40] A. Hirose, Complex-valued neural networks: An introduction, in: Complex-Valued Neural Netw.: Theories and App., World Scientific, 2003, pp. 1–6.

[41] E. Oyallon, S. Mallat, Deep roto-translation scattering for object classification, in: Proc. IEEE Conf. Comput. Vision Pattern Recognition, 2015, pp. 2865–2873.

[42] M. Tygert, J. Bruna, S. Chintala, Y. LeCun, S. Piantino, A. Szlam, A mathematical motivation for complex-valued convolutional networks, Neural Comput. 28 (5) (2016) 815–825.

[43] S. Chintala, A. Szlam, Y. Tian, M. Tygert, W. Zaremba, et al., Scale-invariant learning and convolutional networks, Appl. Comput. Harmonic Anal. 42 (1) (2017) 154–166.

[44] M. Wilmanski, C. Kreucher, A. Hero, Complex input convolutional neural networks for wide angle sar atr, in: IEEE Global Conf. Signal Inf. Process., IEEE, 2016, pp. 1037–1041.

[45] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, G. J. Brostow, Harmonic networks: Deep translation and rotation equivariance, in: Proc. IEEE Conf. Comput. Vision Pattern Recognition, IEEE, 2017, pp. 7168–7177.

[46] C. Godard, K. Matzen, M. Uyttendaele, Deep burst denoising, in: Proc. European Conf. Comput. Vision, 2018.