

## Article

# Disentangling Noise from Images: A Flow-Based Image Denoising Neural Network

Yang Liu <sup>1,2</sup>, Saeed Anwar <sup>1,2,3,\*</sup>, Zhenyue Qin <sup>1</sup>, Pan Ji <sup>4</sup>, Sabrina Caldwell <sup>1</sup> and Tom Gedeon <sup>1</sup> <sup>1</sup> The Research School of Computer Science, The Australian National University, Canberra, ACT 2600, Australia<sup>2</sup> Imaging and Computer Vision, Data61, CSIRO, Canberra, ACT 2600, Australia<sup>3</sup> The School of Computer Science, The University of Technology Sydney, 15 Broadway Ultimo, Sydney, NSW 2007, Australia<sup>4</sup> The OPPO US Research, San Francisco, CA 94303, USA

\* Correspondence: saeed.anwar@csiro.au

**Abstract:** The prevalent convolutional neural network (CNN)-based image denoising methods extract features of images to restore the clean ground truth, achieving high denoising accuracy. However, these methods may ignore the underlying distribution of clean images, inducing distortions or artifacts in denoising results. This paper proposes a new perspective to treat image denoising as a distribution learning and disentangling task. Since the noisy image distribution can be viewed as a joint distribution of clean images and noise, the denoised images can be obtained via manipulating the latent representations to the clean counterpart. This paper also provides a distribution-learning-based denoising framework. Following this framework, we present an invertible denoising network, FDN, without any assumptions on either clean or noise distributions, as well as a distribution disentanglement method. FDN learns the distribution of noisy images, which is different from the previous CNN-based discriminative mapping. Experimental results demonstrate FDN's capacity to remove synthetic additive white Gaussian noise (AWGN) on both category-specific and remote sensing images. Furthermore, the performance of FDN surpasses that of previously published methods in real image denoising with fewer parameters and faster speed.



**Citation:** Liu, Y.; Anwar, S.; Qin, Z.; Ji, P.; Caldwell, S.; Gedeon, T. Disentangling Noise from Images: A Flow-Based Image Denoising Neural Network. *Sensors* **2022**, *22*, 9844. <https://doi.org/10.3390/s22249844>

Academic Editor: Zhengguo Li

Received: 29 September 2022

Accepted: 8 December 2022

Published: 14 December 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** image denoising; invertible network; normalizing flow

## 1. Introduction

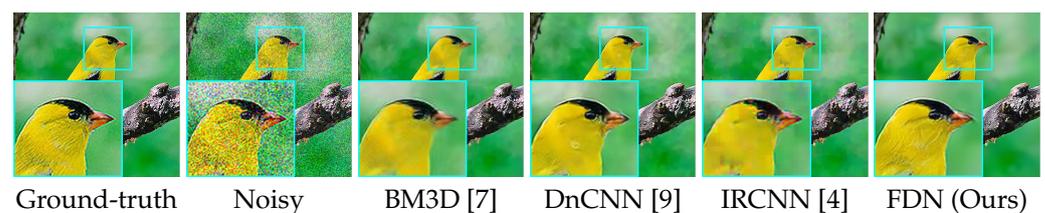
Despite decades of research, image denoising [1,2] is still an ongoing low-level image processing task in computer vision. The long-standing interest in image denoising has provided roots for a vast array of downstream applications, such as segmentation [3] and deblurring [4]. Nearly all images need to be denoised before further processing, especially those obtained in dark environments.

The purpose of image denoising is to reconstruct clean images from corrupted noisy observations. Traditional denoising methods rely on certain assumptions on noise distributions [5,6] or priors on ground truth clean images [7,8] to build optimization models. However, these assumptions and priors may differ from the real case, which can compromise the denoising accuracy. Deep learning denoising approaches proposed in recent years use convolutional neural networks (CNNs) to learn the models from a large number of noise-free and noisy image pairs and have achieved superior denoising performance [9,10]. These methods employ CNNs to learn the mapping functions between noisy images and clean ones. However, they usually overemphasize the pixel similarity between the denoised image and the clean ground truth while omitting the underlying distribution of clean images. Thus, although some deep methods can obtain high quantitative results, over-smoothed regions and artifacts are often brought into the restored images, resulting in degraded visual results.

This paper reconsiders image denoising from the perspective of distribution disentanglement. The distribution of noisy images can be treated as a joint distribution of clean images and noise. Thus, it is intuitive to consider conducting image denoising via disentangling these two distributions. Following this line of thought, the process of distribution-learning-based image denoising can be divided into three stages: the first is to learn the distribution of noisy images by transforming the noisy images into latent representations; the second is to disentangle the representation of clean images from the noisy ones; and the last is to restore clean images from the disentangled clean representation.

There are two challenges for us to overcome: determining which kind of network is suitable for learning the distributions and restoring images, and how to disentangle the two distributions. For the first problem, we resort to generative models to learn the distributions. We require the generative model to generate a denoised image given a disentangled latent code. The denoised image should follow a clean image distribution and be visually similar to the corresponding noisy image. Therefore, the candidate generative model should have a one-to-one mapping between the noisy image space and the latent space. Further, a subspace of the latent space, i.e., the space of the disentangled latent code for clean images, can also be one-to-one mapped to the clean image space. A variational autoencoder (VAE) [11] cannot guarantee the one-to-one mapping between the latent representations and images. On the other hand, although generative adversarial network (GAN) can ensure the one-to-one mapping [12], learning two different distributions, i.e., the distributions of noisy and clean images, requires two discriminative networks, making the design sub-optimal. In this paper, we adopt normalizing flows [13,14], an invertible generative model, to learn the distributions and design the denoising algorithm.

The advantages of normalizing flows are reflected in three aspects. First, their invertibility ensures one-to-one mapping between images and their latent representations [15], ensuring that the manipulation on the latent representation corresponds to modifying the original input image. Second, they are capable of transforming complex distributions to isotropic distributions without losing information [15]. Thus, we can obtain the accurate noisy distribution and also restore the clean images more precisely (see Figure 1). Last, it lets the encoder and the decoder share weights, making the model size much smaller and the training more efficient.



**Figure 1.** Visual comparison on CUB-200 [16] with  $\sigma = 50$  AWGN. Our method restores finer feathers, clearer eyes, and a sharper beak. Zooming in on a high-resolution display will allow better observation of the differences.

For the second challenge, we take advantage of the characteristics of the latent variables, which follow a distribution of  $N(\mathbf{0}, \mathbf{I})$ ; thus, the dimensions are independent to each other. We assume that these dimensions can be disentangled into two groups, i.e., some of the dimensions encode clean images while the others correspond to noise. If we set the noise dimensions to constants, such as  $\mathbf{0}$ , the joint distribution of the clean representations and new noise codes will be the same as the marginal distribution of the clean images. The denoised images can be obtained by passing the new latent representations to the reverse pass of the network.

The contributions of our work are listed below.

- We rethink the image denoising task and present a distribution-learning-based denoising framework.

- We propose a Flow-Based Image Denoising Neural Network (FDN). Unlike the widely used feature-learning-based CNNs in this area, FDN learns the distribution of noisy images instead of low-level features.
- We present a disentanglement method to obtain the distribution of clean ground truth from the noisy distribution, without making any assumptions on noise or employing the priors of images.
- We achieve competitive performance in removing synthetic noise from category-specific images and remote sensing images. For real noise, we also verify our denoising capacity by achieving a new state-of-the-art result on the real-world SIDD dataset.

## 2. Related Work

### 2.1. Recent Trends of Image Denoising

**Traditional Methods.** Traditional denoising methods usually construct an optimization scheme, modeling the distributions of noise or the priors of natural images as penalties or constraints. The widely used natural image priors include sparsity [17], total variation [18,19], non-local similarity [20,21], and external statistical prior [22,23]. NLM [20] computes a weighted average of non-local similar patches to denoise images. The weights are calculated by the Euclidean distance between pixels. BM3D [7] employs the structure similarity of patches in a transform domain, achieving excellent accuracy on denoising additive white Gaussian noise (AWGN).

However, most traditional methods are designed to tackle generic natural images. Very few works study category-specific image denoising and consider the class-specific priors while designing algorithms. CSID [24] was the first to adopt external similar clean patches to facilitate denoising category-specific object images. The authors formulated an optimization problem using the priors in the transform domain. The objective consists of a Gaussian fidelity term that incorporates the category-specific information and a low-rank term that fortifies the similarity between noisy and external similar clean patches. They achieve superior denoising accuracy in removing noise from category-specific images. Nonetheless, a common problem that lies in most of these traditional model-driven methods is that they require noise levels as input. These methods usually implement various hard thresholds to deal with different noise levels. However, the noise level is usually unavailable, and we can only perform blind denoising in practice, limiting the application of these methods.

**Deep Learning Methods.** Deep learning denoising methods learn models from a large number of clean and noisy image pairs with CNNs, without providing image priors manually. The rapid progress of these methods has been seen in recent years, promoting the denoising effect significantly. The notable DnCNN [9] achieves good results on AWGN removal. After that, RIDNet [1] brings attention to denoising models, boosting the denoising performance further. VDN [25] makes assumptions on the distribution of clean images and noise, deriving a new form of evidence lower bound observation (ELBO) under the variational inference framework as the training objective. These CNN-based denoising methods learn low-level features in the network to restore the details of clean images.

There have also been a few attempts in designing category-specific denoising networks recently, for example, [26] proposes a class-aware CNN-based denoising method. The authors use a classifier to classify the noisy image into the supported classes first and then exploit the pre-trained class-specific denoising models for denoising. For each of the supported classes, the denoising model is pre-trained on the images from the same classes of ImageNet [27]. The denoising architecture they proposed is a feature-learning-based CNN. However, for category-specific images, the feature-learning-based denoising methods usually enforce the pixels of denoised images to be close to the clean ones but ignore the underlying distribution of the specific category. Thus, over-smoothed regions and artifacts are seen in restored images, degrading the visual effects of denoising. As far as we know, we are the first to conduct image denoising with distribution learning and disentanglement.

## 2.2. Flow-Based Invertible Networks

We employ normalizing flows based invertible neural networks to learn the distributions. Normalizing flows [28] are models for computing complex distributions accurately. By applying a sequence of invertible transformations to transform a simple prior distribution into a complex distribution, the complex distribution's exact log-likelihood can be computed.

The key design concept of normalizing flows is invertibility, ensuring that the mapping between an input and its output is one-to-one. Therefore, to estimate the probability density of image  $\mathbf{y}$ , we can alternatively achieve the same purpose by measuring the probability density of the counterpart latent variable  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . Estimating the probability density of  $\mathbf{y}$  through using the probabilities of  $\mathbf{z}$  requires taking the variations of metric spaces into consideration. Consequently, we have

$$p(\mathbf{y}) = p(\mathbf{z}) \left| \det \left( \frac{\partial f^{-1}(\mathbf{z})}{\partial \mathbf{z}} \right) \right| = p(\mathbf{z}) \left| \det \left( \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right) \right|^{-1}, \quad (1)$$

where  $\mathbf{z} = f(\mathbf{y})$  and  $\mathbf{y} = f^{-1}(\mathbf{z})$ ;  $f$  is the invertible function learned by normalizing flows.

To reduce the complexity of computing the determinants of Jacobian matrices, special designs are proposed in NICE [28] and Real NVP [13] to make each flow module have a triangular Jacobian matrix. Glow [14] extends the channel permutation methods in these two models and proposes invertible  $1 \times 1$  convolutional layers. These models are usually used in image generation, demonstrating superior generation quality of natural images.

So far, few studies have applied invertible networks to image denoising. Noise Flow [29] employs Glow [14] to learn the distribution of real-world noise and generate real noisy images for data augmentation. Extra information, such as raw images, ISO, and camera-specific parameters, is required during noisy image generation. Different from these studies, we are the first to exploit normalizing flows to learn the distribution of noisy images and disentangle the clean representations to restore images.

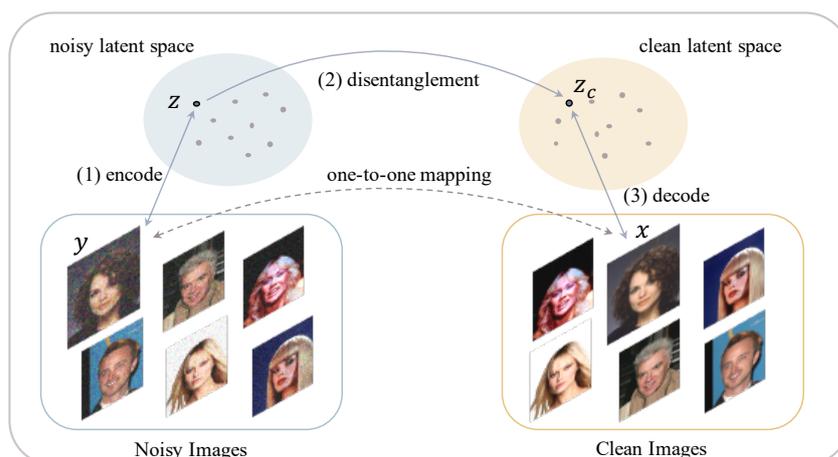
## 3. Our Method

In this section, we explain the design concept of FDN. Then, we introduce the detailed components of the network architecture. The objective function, as well as some training details, are also presented.

### 3.1. Concept of Design

We rethink the image denoising task from the perspective of distribution learning and disentanglement. Suppose the noisy image is  $\mathbf{y}$  and the corresponding clean ground truth is  $\mathbf{x}$ . The noise  $\mathbf{n} = \mathbf{y} - \mathbf{x}$ . We have  $p(\mathbf{y}) = p(\mathbf{x}, \mathbf{n}) = p(\mathbf{x})p(\mathbf{n}|\mathbf{x})$ —that is, the distribution of the noisy images  $p(\mathbf{y})$  is a joint distribution of clean images and noise. The clean representation can be achieved if we can disentangle the clean and noise representations from  $p(\mathbf{y})$ . Then, the clean images can be restored with the disentangled clean representation.

A framework of this scheme is presented in Figure 2, which contains three steps: (i) learn the distribution of noisy images by encoding  $\mathbf{y}$  to a noisy latent representation  $\mathbf{z}$ , (ii) disentangle the clean representation  $\mathbf{z}_C$  from  $\mathbf{z}$ , and (iii) restore the clean image by decoding  $\mathbf{z}_C$  to the clean image space. To ensure the denoising effect, the mappings between  $\mathbf{y}$  and  $\mathbf{z}$ ,  $\mathbf{z}_C$  and  $\mathbf{x}$  should be one-to-one.



**Figure 2.** Framework of distribution learning and disentanglement-based image denoising.

An invertible normalizing flow-based network is employed to learn the distribution of noisy images  $p(\mathbf{y})$ , transforming  $\mathbf{y}$  to latent variables  $\mathbf{z}$  following a simple prior distribution  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ . Thus,

$$p(\mathbf{y}) = p(\mathbf{x})p(\mathbf{n}|\mathbf{x}) = p(\mathbf{z}) \left| \det\left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}\right) \right|^{-1}, \quad (2)$$

where  $f(\cdot)$  is the model learned by the network. The dimensions of  $\mathbf{z}$  are independent of each other.

We assume that  $\mathbf{z}$  can be disentangled; some of the dimensions of  $\mathbf{z}$  encode the distribution of clean images (denoted as  $\mathbf{z}_C$ ) and the remaining embed noise (denoted as  $\mathbf{z}_N$ ). The clean image  $\mathbf{x}$  can be restored through the following transformation:

$$p(\mathbf{x}) = p(\mathbf{z}_C) \left| \det\left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}_C}\right) \right|^{-1}. \quad (3)$$

However, how to obtain  $\mathbf{z}_C$  with  $\mathbf{z}$  is not so obvious. We propose a way of disentanglement by setting  $\mathbf{z}_N = \mathbf{0}$ —that is,

$$\hat{\mathbf{z}} = \mathbf{m} \odot \mathbf{z}, \quad (4)$$

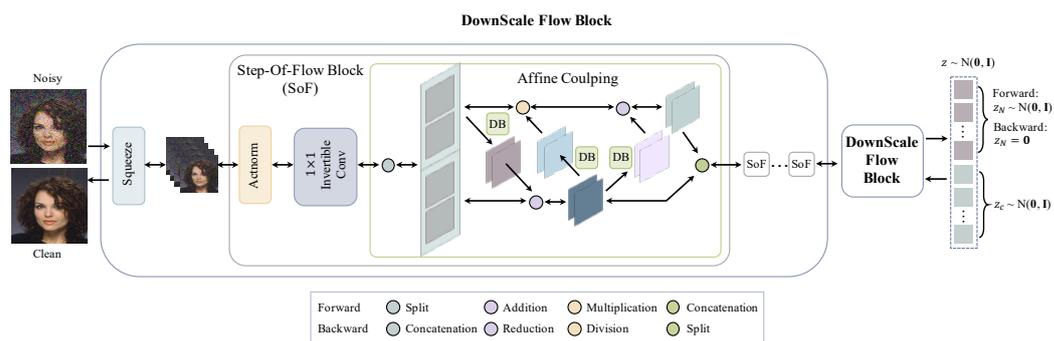
where  $\mathbf{m}$  is a mask that is 1 in the dimensions for clean variables and 0 in those for noise.  $\hat{\mathbf{z}}$  is a new latent code that only contains the clean representations.  $\odot$  denotes the element-wise product. Thus, we have  $p(\mathbf{z}_N = \mathbf{0}) = 1$ , and the distribution of  $\hat{\mathbf{z}}$  becomes

$$p(\hat{\mathbf{z}}) = p(\mathbf{z}_C)p(\mathbf{z}_N) = p(\mathbf{z}_C). \quad (5)$$

Then, the clean image can be obtained via Equation (3).

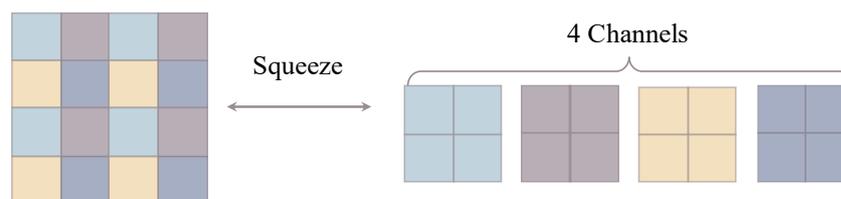
### 3.2. Network Architecture

The details of our FDN architecture are presented in this section. FDN is composed of several invertible DownScale Flow Blocks, as shown in Figure 3. Each block consists of a Squeeze layer to downscale the latent representations followed by several Step-Of-Flow Blocks to perform distribution transformation. The details of each layer are described below.



**Figure 3.** Our FDN Network Architecture. FDN consists of several invertible DownScale Flow Blocks. Module DB is the Dense Block proposed in [30]. The forward pass encodes the corrupted image to latent variables  $z$ , which follow a Gaussian distribution, i.e.,  $z = (z_N, z_C) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . The latent representations of noise  $z_N$  are set to  $\mathbf{0}$ , combined with the clean latent variables  $z_C$  as a new latent representation  $\hat{z}$ . The backward pass decodes  $\hat{z}$  to the denoised image.

**Squeeze.** The Squeeze layers take every other element of the intermediate latent variables, resulting in new downscaled latent representations with quadruple channels, as illustrated in Figure 4.



**Figure 4.** The Squeeze operation downscales latent representations according to a checkerboard pattern.

**Actnorm.** The Actnorm layers apply the affine transformation on latent variables, as illustrated in Equation (7).

$$\mathbf{h}_{i+1} = \mathbf{s}_1 \odot \mathbf{h}_i + \mathbf{b}_1, \tag{6}$$

where  $\mathbf{h}_i$  and  $\mathbf{h}_{i+1}$  are the intermediate latent representations during transformation. The output of the forward pass  $z$  can be treated as the final layer of the latent representations.  $\mathbf{s}_1$  and  $\mathbf{b}_1$  are the scale and translation parameters, respectively.  $\odot$  is the Hadamard product of tensors. The reverse operation of the Actnorm layer is

$$\mathbf{h}_i = (\mathbf{h}_{i+1} - \mathbf{b}_1) / \mathbf{s}_1, \tag{7}$$

$\mathbf{s}_1$  and  $\mathbf{b}_1$  are initialized to make each channel of the representations have zero mean and unit variance, such as the normalization operation. However, during training, this operation is different from the widely used normalization methods. Specifically,  $\mathbf{s}_1$  and  $\mathbf{b}_1$  are updated through back-propagation, without any further constraints on the mean and variance of the latent variables. Employing the Actnorm layers is able to improve the training stability and performance.

**Invertible  $1 \times 1$  Convolutional Layers.** Different from ordinary convolutional layers, we use the invertible  $1 \times 1$  convolutional layers, which are designed for normalizing flows to support invertibility. The operation in these layers can be represented as

$$\mathbf{h}_{i+1} = \mathbf{W}\mathbf{h}_i, \tag{8}$$

where  $\mathbf{W}$  is a square matrix that is initialized randomly. Its reverse function is

$$\mathbf{h}_i = \mathbf{W}^{-1}\mathbf{h}_{i+1}. \tag{9}$$

These layers are used to permute different channels of latent representations.

**Affine Coupling Layers.** The Affine Coupling layers capture the correlations among spatial dimensions [13,28]. The forward operations include

$$\begin{aligned} \mathbf{h}_i^a, \mathbf{h}_i^b &= \text{Split}(\mathbf{h}_i), \\ \mathbf{h}_{i+1}^a &= \mathbf{h}_i^a + \mathfrak{g}_1(\mathbf{h}_i^b), \\ \mathbf{h}_{i+1}^b &= \mathfrak{g}_2(\mathbf{h}_{i+1}^a) \odot \mathbf{h}_i^b + \mathfrak{g}_3(\mathbf{h}_{i+1}^a), \\ \mathbf{h}_{i+1} &= \text{Concat}(\mathbf{h}_{i+1}^a, \mathbf{h}_{i+1}^b), \end{aligned}$$

where  $\text{Split}(\cdot)$  and  $\text{Concat}(\cdot)$  operate along channel dimensions.  $\text{Split}(\cdot)$  splits  $\mathbf{h}_i$  into two tensors  $\mathbf{h}_i^a$  and  $\mathbf{h}_i^b$ .  $\text{Concat}(\cdot)$  concatenates two tensors  $\mathbf{h}_{i+1}^a, \mathbf{h}_{i+1}^b$  channel-wise to obtain  $\mathbf{h}_{i+1}$ .  $\mathfrak{g}_i(\cdot)$  ( $i = 1, 2, 3$ ) is the neural network. The reverse operations are

$$\begin{aligned} \mathbf{h}_{i+1}^a, \mathbf{h}_{i+1}^b &= \text{Split}(\mathbf{h}_{i+1}), \\ \mathbf{h}_i^b &= (\mathbf{h}_{i+1}^b - \mathfrak{g}_3(\mathbf{h}_{i+1}^a)) / \mathfrak{g}_2(\mathbf{h}_{i+1}^a), \\ \mathbf{h}_i^a &= \mathbf{h}_{i+1}^a - \mathfrak{g}_1(\mathbf{h}_i^b), \\ \mathbf{h}_i &= \text{Concat}(\mathbf{h}_i^a, \mathbf{h}_i^b). \end{aligned}$$

The operations in the second and third row turn  $+$  into  $-$  and  $\odot$  into  $/$ .  $\mathfrak{g}_i(\cdot)$  ( $i = 1, 2, 3$ ) can be any neural network. Following [31,32], we employ Dense Block (DB) in our network as  $\mathfrak{g}_i(\cdot)$ .

### 3.3. Objective Function

Our objective function consists of two components: the distribution learning loss to encode the input noisy image  $\mathbf{y}$  into latent code  $\mathbf{z}$ , and the reconstruction loss to restore the corresponding clean image  $\mathbf{x}$  with clean code  $\mathbf{z}_C$ . The details of these two losses are as below.

#### Distribution Learning Loss.

$$L_{\text{dis}} = -\log p(\mathbf{y}) = -(\log p_z(\mathbf{z}) + \sum_{i=1}^L \log(\det |\frac{\partial f_i}{\partial \mathbf{h}_i}|)), \quad (10)$$

where  $L$  is the number of invertible layers in FDN and  $f_i$  is the function learned by each layer.  $p_z(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and  $\mathbf{z} = (\mathbf{z}_C, \mathbf{z}_N)$ . To reconstruct the clean image, we set  $\mathbf{z}_N = \mathbf{0}$  and achieve a new latent representation  $\hat{\mathbf{z}} = (\mathbf{z}_C, \mathbf{0})$ , which lies in a subspace of  $\mathbf{z}$ .

**Reconstruction Loss.**  $\hat{\mathbf{z}}$  is passed through the reverse network to restore the clean ground truth.

$$L_{\text{rec}} = \|\mathbf{f}^{-1}(\hat{\mathbf{z}}) - \mathbf{x}\|_1. \quad (11)$$

**Total Loss.** The total objective function we use during training is

$$L = \lambda_1 L_{\text{dis}} + \lambda_2 L_{\text{rec}}, \quad (12)$$

where  $\lambda_1$  and  $\lambda_2$  are the weights for the two loss components.

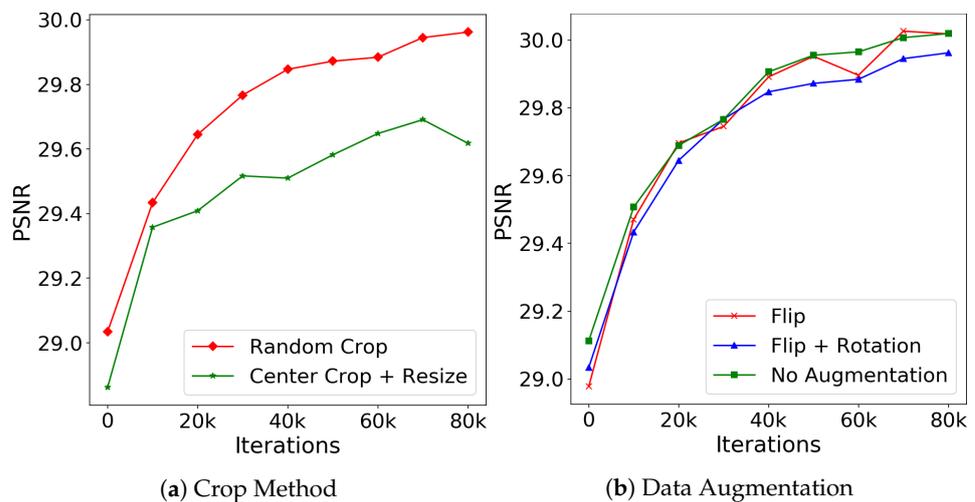
### 3.4. Data Preprocessing

Since FDN is the first distribution-learning-based denoising network, we explore different data preprocessing techniques to demonstrate how to make the best use of it.

**Random vs. Center Crop and resize.** The widely used training strategy in feature-learning-based CNN denoising networks is to randomly crop patches from the training dataset to learn features. However, it is not obvious whether the random crop strategy is also superior in distribution-learning-based networks. Take face image denoising as an example; if we center crop the face region and resize it into an appropriate size, we will

obtain a downscaled face image, following a similar distribution as the face image test set. Intuitively, the center crop with the resizing method will facilitate the network to learn the distribution better and achieve superior denoising results.

Thus, we compare training with random crop and center crop with resizing, illustrating the curves of the validation results in Figure 5a. Contrary to our intuition, cropping training patches randomly outperforms the center crop with resizing consistently and significantly. Therefore, we apply random crops while training the FDN.



**Figure 5.** Training with different processing strategies. (a) Illustrates the difference between random and center crop accompanied with resizing. (b) The validation curves using different data augmentation methods. All the models are trained on CelebA [33] with  $\sigma = 50$ .

**Data Augmentation vs. No Data Augmentation.** Feature-learning-based networks usually employ horizontal and vertical flip and rotation with 90, 180, and 270 degrees for data augmentation. However, these methods will bring in unrealistic patches, compromising the learning of distributions. For example, if we rotate a patch of a face image, we may obtain patches with the eyes under the mouth or on the mouth's left side, which is impossible in real face images. Although data augmentation can lead to better generalizability for discriminative models, it may bring noise when learning the distributions.

We train three models for comparison: one with flip and rotation as data augmentation, one with only flip as data augmentation, and the other is trained without any augmentation. The validation results are shown in Figure 5b. The results verify our concern that inappropriate data augmentation such as rotation introduces noise to the distribution model, resulting in the lower denoising accuracy shown by the blue curve. Training with only flip as data augmentation achieves almost the same results as without data augmentation; however, the latter is more stable during training. The potential reason might be that the horizontal flip also generates realistic images for face images, while the vertical flip creates impossible face images, making the training unstable. Thus, to avoid unrealistic samples, we train our distribution-learning-based networks without data augmentation when the training set is large enough to learn the distribution. If the training set is small, we only conduct data augmentation that will not generate unrealistic patches.

#### 4. Experiment

We perform thorough experiments to demonstrate the effectiveness of our method. We first apply FDN to denoise category-specific images. Since category-specific images usually have similar patterns in all the images, such as similar facial contours and features in human faces, their distribution is easier to learn than random nature images. The experiment is then extended to denoising more difficult remote sensing images, which contain diverse terrain patterns, such as mountains or forests, following intricate distributions. Finally, we investigate our capacity to remove noise, which follows complicated distribution in the

real noise dataset. Further details are provided about the datasets, training strategies, and qualitative and quantitative results.

#### 4.1. Experimental Settings

##### 4.1.1. Training Details

FDN with two DownScale Flow blocks and eight SoF blocks in each Flow block is exploited in our experiment, where ADAM [34] is applied as an optimizer. The learning rate is initialized as  $2 \times 10^{-4}$  and halved after every 50K iterations. To evaluate the methods, we employ Peak Signal-to-Noise Ratio (PSNR) as the evaluation metric.

##### 4.1.2. Datasets

Next, we provide information about the category-specific, remote sensing, and real-world datasets.

**Category-Specific Datasets:** We investigate the capacity of FDN in removing AWGN on three category-specific datasets: faces, flowers, and birds.

- CelebA [33] is a large human face dataset containing 202,599 face images. We use the 162,770 training images for training and 19,867 validation images for testing. The training images are cropped into  $64 \times 64$  patches randomly as the network's input at the training stage. Since the training set is large enough, we do not apply any data augmentation during training.
- The Flower Dataset [35] contains 102 categories of flowers, including 1020 training images, 1020 validation images, and 6149 test images. To better learn the distribution of flowers, we change the dataset's partition and use the 6149 images as the training set and the remaining 2040 images as the test set. The training images are randomly cropped into patches with a size of  $128 \times 128$ . Flipping and rotation are employed as data augmentation.
- The CUB-200 Dataset [16] includes 11,788 bird images, covering 200 categories of birds. We use 5989 images as the training set and 5790 images as the test set. The training images are cropped into  $128 \times 128$  patches with random flipping as data augmentation during training.

**Remote Sensing Datasets:** We attempt to denoise two remote sensing datasets (RICE1 and RICE2 [36]) with AWGN added to explore our capability when the distribution of ground truth images becomes complex. The datasets contain 500 and 450 pairs of images, respectively, each with a size of  $512 \times 512$ . We randomly crop patches of size  $64 \times 64$  from the images for training and add AWGN with  $\sigma = 30, 50, \text{ and } 70$  to obtain noise-free and noisy pairs, respectively. Random flipping, as well as rotation, are utilized for data augmentation.

**Real Noisy Datasets:** Finally, we verify FDN's effectiveness in removing real noise, which follows a complex distribution. Real image noise can result from photon shot noise, fixed pattern noise, dark current, readout noise, quantization noise, etc. during the imaging process [37]. We conduct real noise removal on the dataset SIDD [38], which is taken by five smartphone cameras with small apertures and sensor sizes. The medium SIDD dataset contains 320 clean and noisy pairs. Patches with a size of  $144 \times 144$  are randomly cropped for training. Flipping and rotation are adopted for augmentation.

#### 4.2. Category-Specific Image Denoising

**Quantitative Results.** In Table 1, we report numeric values for the three category-specific datasets with added AWGN with the levels of  $\sigma = 15, 25, \text{ and } 50$ . Compared with other competitive methods in synthetic noise removal, FDN achieves the highest PSNR on all the datasets and noise levels.

**Table 1.** Quantitative comparison of removing synthetic noise on three category-specific datasets. Our FDN outperforms the other competitive methods on all of the three datasets for various noise levels.

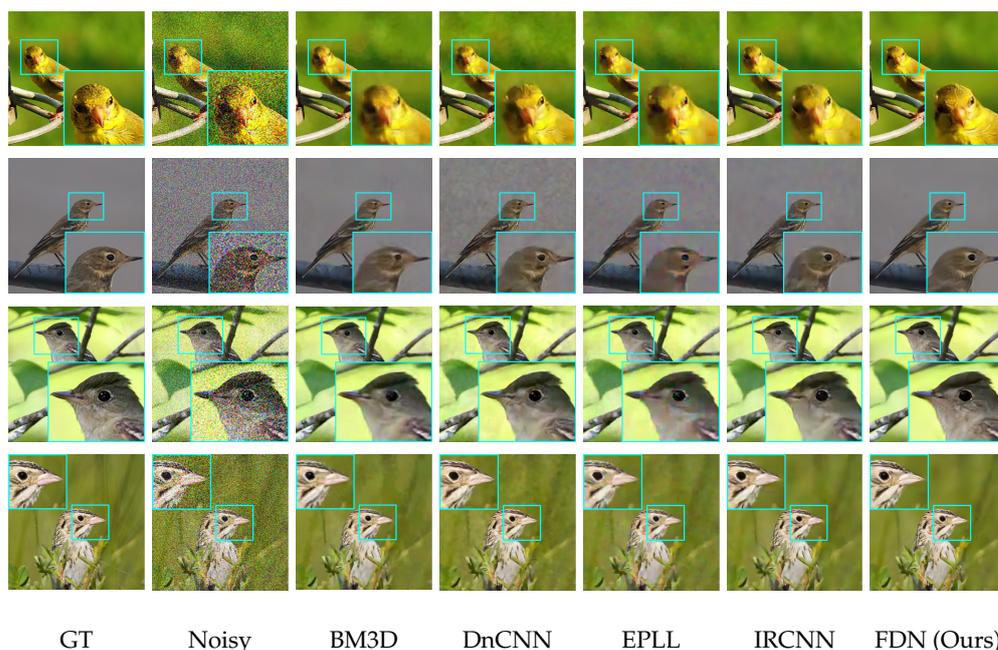
| Dataset      | $\sigma$ | BM3D [7] | EPLL [39] | IRCNN [4] | REDNet [40] | DnCNN [9] | FFDNet [10] | FDN (Ours)   |
|--------------|----------|----------|-----------|-----------|-------------|-----------|-------------|--------------|
| CelebA [33]  | 15       | 35.46    | 33.29     | 35.20     | 35.23       | 35.04     | 35.14       | <b>35.74</b> |
|              | 25       | 32.80    | 30.81     | 32.62     | 32.68       | 32.63     | 32.40       | <b>32.95</b> |
|              | 50       | 29.46    | 27.65     | 29.24     | 29.56       | 29.57     | 29.44       | <b>30.29</b> |
|              | Blind    | –        | –         | 31.92     | 33.16       | 32.17     | 32.20       | <b>33.52</b> |
| Flower [35]  | 15       | 37.20    | 35.41     | 36.83     | 36.93       | 36.73     | 36.49       | <b>37.38</b> |
|              | 25       | 34.73    | 32.92     | 34.47     | 34.75       | 34.17     | 33.89       | <b>34.82</b> |
|              | 50       | 31.38    | 29.58     | 30.8      | 31.34       | 30.38     | 30.74       | <b>31.71</b> |
|              | Blind    | –        | –         | 34.91     | 34.57       | 34.55     | 34.61       | <b>35.18</b> |
| CUB-200 [16] | 15       | 35.08    | 33.31     | 35.14     | 35.16       | 35.21     | 34.86       | <b>35.30</b> |
|              | 25       | 32.59    | 30.83     | 32.71     | 32.80       | 32.45     | 32.33       | <b>32.94</b> |
|              | 50       | 29.32    | 27.61     | 29.28     | 29.72       | 28.87     | 28.61       | <b>29.79</b> |
|              | Blind    | –        | –         | 32.49     | 33.18       | 32.89     | 33.07       | <b>33.20</b> |

We also employ the same FDN for blind denoising with noise levels between [0, 55], as shown in Table 1. The distribution of blind noise is a Gaussian Mixture Model, which is much more complicated than the Gaussian distribution with a certain noise level. Although traditional methods such as BM3D [7] and EPLL [39] are good at removing Gaussian noise, their capacity in blind denoising is unavailable due to the requirement of the noise level as input. Comparing with other feature-learning-based CNN methods, FDN outperforms others to a large extent, exhibiting its superiority in category-specific image denoising.

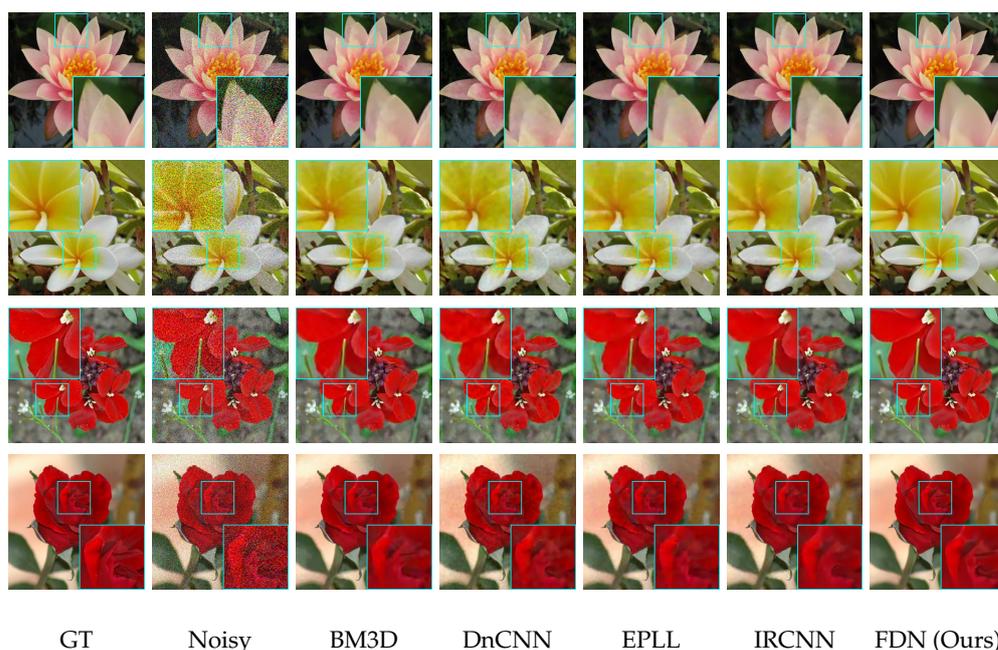
**Qualitative Results.** The visual results are shown in Figures 6–8. For CelebA, we observe that, although the other competitive methods can restore the facial contour, they lose many detailed facial features. Thus, the denoised images of these methods are blurred with artifacts. In contrast, our denoising results are much clearer and closer to the ground truth images. For the Flower and CUB-200 datasets, the foreground and background are more diverse and complicated than CelebA. Our results are clean with sharp edges (in close-up versions), while other methods have artifacts near and at the edges. This illustrates that FDN can handle category-specific image denoising very well.



**Figure 6.** Image denoising results of FDN on CelebA dataset with  $\sigma = 50$  against competitive methods. Our network produces results close to the ground-truth without any kind of deformation or artifact. The effects are best viewed zoomed-in.



**Figure 7.** Comparison of denoising results on CUB-200 dataset having  $\sigma = 50$ . Our method removes artifacts and noise, providing clean edges and textures.



**Figure 8.** Visual comparison on the Flower dataset for  $\sigma = 50$  against state-of-the-art methods.

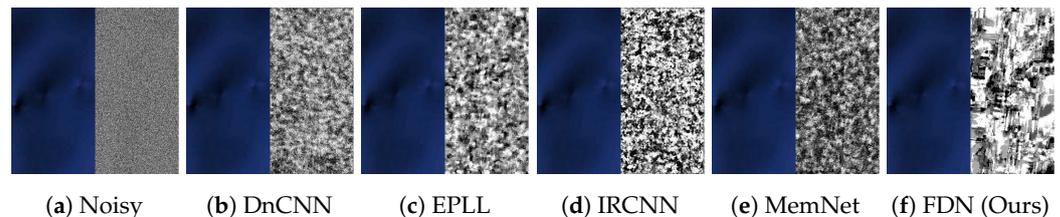
#### 4.3. Remote Sensing Image Denoising

**Quantitative Results.** The results of denoising the RICE1 and RICE2 [36] datasets with  $\sigma = 30, 50,$  and  $70$  are reported in Table 2. FDN improves by 1.13 dB–1.74 dB on RICE1 with different noise levels compared with the highest results from other competitive methods. On RICE2, FDN outperforms other methods when the noise levels are large, i.e., achieving increases of 0.97 dB and 1.55 dB for  $\sigma = 50$  and  $70$ , respectively. These results demonstrate that FDN is also capable of restoring images following complex distributions.

**Table 2.** Performance comparison on the two remote sensing datasets.

| Dataset    | $\sigma$ | EPLL [39] | MemNet [41]  | IRCNN [4] | REDNet [40] | DnCNN [9] | FFDNet [10] | FDN (Ours)   |
|------------|----------|-----------|--------------|-----------|-------------|-----------|-------------|--------------|
| RICE1 [36] | 30       | 31.95     | 31.82        | 31.12     | 29.98       | 30.69     | 22.68       | <b>33.08</b> |
|            | 50       | 29.65     | 27.71        | 27.50     | 28.82       | 26.99     | 24.17       | <b>31.14</b> |
|            | 70       | 28.29     | 27.12        | 26.53     | 26.56       | 25.04     | 23.51       | <b>30.03</b> |
| RICE2 [36] | 30       | 36.05     | <b>36.49</b> | 35.83     | 33.12       | 34.68     | 34.02       | 35.93        |
|            | 50       | 33.22     | 33.62        | 33.74     | 30.40       | 29.57     | 30.26       | <b>34.71</b> |
|            | 70       | 31.63     | 31.73        | 32.43     | 27.55       | 30.81     | 28.51       | <b>33.98</b> |

**Qualitative Results.** The visual results are illustrated in Figure 9 (Although in RGB images, blacker pixels represent smaller values, we change every pixel in the right regions by using 255 minus the value; thus, the whiter regions are smaller.). The remote sensing datasets are mainly composed of images with two types of regions: the texture regions such as mountains, and the smooth regions, such as deserts. An example of the smooth region from RICE1 with  $\sigma = 70$  is taken. Our FDN outperforms other methods significantly from the right regions of Figure 9b–f. Thus, our distribution learning and disentanglement based denoising method, i.e., FDN, has proven to be effective not only for category-specific data but also for images following more complex distributions.



**Figure 9.** Visual results on RICE1 with  $\sigma = 70$ . For (a), the left part is the clean image and the right part is the noise. For (b–f), the left part is the denoised image and the right region reflects the difference between the denoised and GT images. Whiter pixels represent better denoising performance. The denoised image restored by FDN is closer to the ground truth.

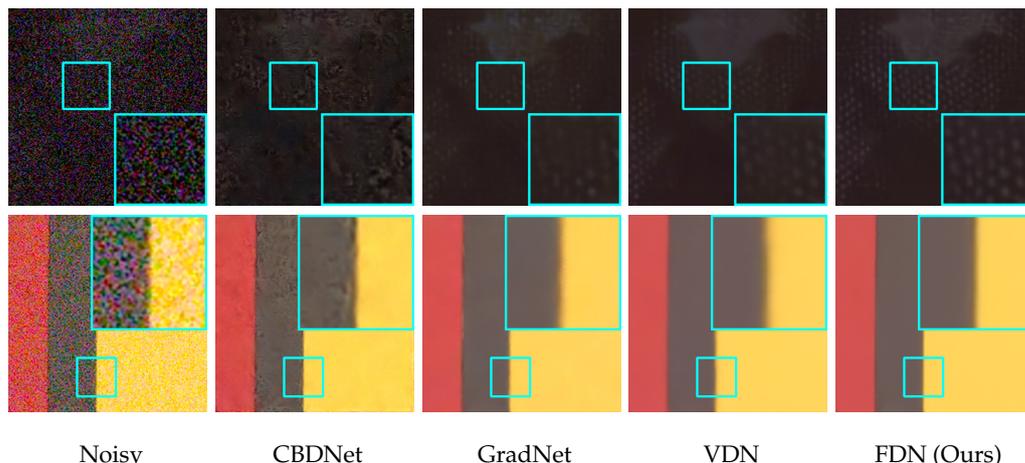
#### 4.4. Real Image Denoising

**Quantitative Results.** The performance comparison on the test set of the real noise dataset SIDD [38] is listed in Table 3. We achieve a new state-of-the-art denoising accuracy comparing with other methods. In addition, our model size (4.38 M) is much smaller than the competitive AINDNet [42] (13.76 M) and VDN [25] (7.81 M), illustrating that FDN is suitable to be deployed on small edge devices. We also report the inference time (in GigaFlops) of one  $256 \times 256$  image for each method. FDN is much faster than VDN [25].

**Table 3.** Quantitative comparison on the real noisy SIDD dataset trained on SIDD medium dataset.

| Method                  | DnCNN [9] | TNRD [43] | BM3D [7] | CBDNet [44] | GradNet [2] | AINDNet [42] | VDN [25] | FDN (Ours)   |
|-------------------------|-----------|-----------|----------|-------------|-------------|--------------|----------|--------------|
| PSNR (dB)               | 23.66     | 24.73     | 25.65    | 33.28       | 38.34       | 39.08        | 39.26    | <b>39.31</b> |
| SSIM                    | 0.583     | 0.643     | 0.685    | 0.868       | 0.953       | 0.955        | 0.955    | <b>0.955</b> |
| Param (M)               | 0.56      | –         | –        | 4.34        | 1.60        | 13.76        | 7.81     | 4.38         |
| Inference time (GFlops) | 73.32     | –         | –        | 80.76       | 213.06      | –            | 99.00    | 76.80        |

**Qualitative Results.** To further present the effectiveness of FDN against other state-of-the-art methods, we show the visual results of denoised images in Figure 10. FDN restores accurate textures and well-shaped edges, while other methods blur details and introduce artifacts. This indicates that FDN is also superior in removing real-world noise.



**Figure 10.** The visual comparison on the SIDD dataset against state-of-the-art methods. In the first row, FDN reconstructs the white dot patterns clearly in a dark environment without smoothing and artifacts. In the second row, FDN preserves more crisp edges.

#### 4.5. Ablation Study and Discussion

**Number of Flow and SoF Blocks.** We study the denoising effects of employing different numbers of Flow and SoF blocks in FDN. We train models on CelebA [33] with  $\sigma = 50$  AWGN added. The results of the 50Kth iteration on the validation set are reported in Table 4. In general, given the same number of Flow blocks, the more SoF blocks in each of the Flow blocks, the higher the denoising accuracy. However, the improvement is not significant when we have three Flow blocks. On the other hand, with the same number of SoF blocks in each Flow block, increasing Flow block numbers from one to two improves the performance. Nevertheless, further increments result in similar accuracy when SoF = 4 and even slightly decrease when SoF = 8. Thus, we adopt two Flow blocks with eight SoF blocks contained in our experiment.

**Table 4.** Comparisons on the denoising accuracy of different numbers of DownScale Blocks and Invertible Blocks.

| PSNR        |         | SoF Blocks |              |
|-------------|---------|------------|--------------|
|             |         | Num = 4    | Num = 8      |
| Flow Blocks | num = 1 | 29.59      | 29.86        |
|             | num = 2 | 29.87      | <b>30.00</b> |
|             | num = 3 | 29.89      | 29.90        |

**The split of  $\mathbf{z}_C$  and  $\mathbf{z}_N$ .** We also study the effects of different dimension numbers of  $\mathbf{z}_C$  (i.e.,  $\dim(\mathbf{z}_C)$ ). Models are trained with  $\dim(\mathbf{z}_C) = 1/8, 1/4, 1/2, 3/4,$  and  $7/8 \dim(\mathbf{z})$  separately, and the validation results of the 50Kth iteration are reported in Table 5. In general, the denoising accuracy improves with the increase of the proportion of  $\dim(\mathbf{z}_C)$ . On the other hand, the results are almost the same when  $\dim(\mathbf{z}_C) = 3/4$  and  $7/8 \dim(\mathbf{z})$ , illustrating that extending the dimensions of  $\mathbf{z}_C$  will not boost the denoising performance further. Thus, we use  $\dim(\mathbf{z}_C) = 3/4 \dim(\mathbf{z})$  in our experiment.

**Table 5.** Comparisons on different proportions of the dimensions of  $\mathbf{z}_C$  in  $\mathbf{z}$ .

| $\dim(\mathbf{z}_C)$ | 1/8   | 1/4   | 1/2   | 3/4          | 7/8   |
|----------------------|-------|-------|-------|--------------|-------|
| PSNR                 | 29.81 | 30.00 | 30.00 | <b>30.19</b> | 30.18 |

## 5. Conclusions

The widely used image denoising CNNs are discriminative models, learning the mapping between noisy images and their clean counterparts via learning features of images. However, these methods may overlook the underlying distribution of the clean ground truth, resulting in downgraded visual results with blurry regions or artifacts. This paper provides a new perspective to understand image denoising as a distribution disentangling task. Since the distribution of noisy images can be treated as a joint distribution of clean images and noise, the denoised images can be obtained via the clean images' latent representations. A distribution-learning-based denoising framework is proposed in this paper. We also present a novel denoising network, FDN, based on normalizing flows without adding any assumptions on clean images and noise distributions. FDN learns the distribution instead of features from noisy images, which is different from previous feature-learning-based networks. A distribution disentanglement method for denoising is introduced as well. Experimental results verify the effectiveness of FDN on both category-specific and remote sensing images denoising with synthetic AWGN. Moreover, FDN shows its superiority in real image denoising with fewer parameters and a lower running time. In conclusion, this paper presents a new potential direction to optimize image denoising methods in the future.

**Author Contributions:** Conceptualization, Y.L. and S.A.; methodology, Y.L., S.A. and Z.Q.; validation, Y.L. and Z.Q.; formal analysis, S.A. and P.J.; investigation, Y.L., S.A. and Z.Q.; resources, S.A.; data curation, Y.L. and Z.Q.; writing—original draft preparation, Y.L. and Z.Q.; writing—review and editing, S.A., P.J., T.G. and S.C.; supervision, S.A., P.J., T.G. and S.C.; project administration, S.A., P.J., T.G. and S.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The data and models are available at <https://github.com/Yang-Liu1082/FDN.git> (accessed on 15 October 2022).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Anwar, S.; Barnes, N. Real Image Denoising with Feature Attention. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.
2. Liu, Y.; Anwar, S.; Zheng, L.; Tian, Q. GradNet Image Denoising. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 508–509.
3. Minaee, S.; Boykov, Y.; Porikli, F.; Plaza, A.; Kehtarnavaz, N.; Terzopoulos, D. Image Segmentation Using Deep Learning: A Survey. *arXiv* **2020**, arXiv:cs.CV/2001.05566.
4. Zhang, K.; Zuo, W.; Gu, S.; Zhang, L. Learning Deep CNN Denoiser Prior for Image Restoration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
5. Meng, D.; De la Torre, F. Robust Matrix Factorization with Unknown Noise. In Proceedings of the 2013 IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 1337–1344. [[CrossRef](#)]
6. Zhu, F.; Chen, G.; Hao, J.; Heng, P. Blind Image Denoising via Dependent Dirichlet Process Tree. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1518–1531. [[CrossRef](#)] [[PubMed](#)]
7. Dabov, K.; Foi, A.; Katkovnik, V.; Egiazarian, K. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Trans. Image Process.* **2007**, *16*, 2080–2095. [[CrossRef](#)] [[PubMed](#)]
8. Peng, Y.; Ganesh, A.; Wright, J.; Xu, W.; Ma, Y. RASL: Robust Alignment by Sparse and Low-Rank Decomposition for Linearly Correlated Images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2233–2246. [[CrossRef](#)] [[PubMed](#)]
9. Zhang, K.; Zuo, W.; Chen, Y.; Meng, D.; Zhang, L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Trans. Image Process.* **2017**, *26*, 3142–3155. [[CrossRef](#)] [[PubMed](#)]
10. Zhang, K.; Zuo, W.; Zhang, L. FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising. *IEEE Trans. Image Process.* **2018**, *27*, 4608–4622. [[CrossRef](#)]
11. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. *arXiv* **2013**, arXiv:stat.ML/1312.6114.
12. Ma, F.; Ayaz, U.; Karaman, S. Invertibility of convolutional generative networks from partial measurements. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 9628–9637.
13. Dinh, L.; Sohl-Dickstein, J.; Bengio, S. Density estimation using Real NVP. *arXiv* **2016**, arXiv:cs.LG/1605.08803.

14. Kingma, D.P.; Dhariwal, P. Glow: Generative Flow with Invertible  $1 \times 1$  Convolutions. In *Advances in Neural Information Processing Systems 31*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; pp. 10215–10224.
15. Liu, Y.; Qin, Z.; Anwar, S.; Caldwell, S.; Gedeon, T. Are Deep Neural Architectures Losing Information? Invertibility Is Indispensable. *arXiv* **2020**, arXiv:cs.CV/2009.03173.
16. Welinder, P.; Branson, S.; Mita, T.; Wah, C.; Schroff, F.; Belongie, S.; Perona, P. *Caltech-UCSD Birds 200*; Technical Report CNS-TR-2010-001; California Institute of Technology: Pasadena, CA, USA, 2010.
17. Mairal, J.; Bach, F.; Ponce, J.; Sapiro, G.; Zisserman, A. Non-local sparse models for image restoration. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 2272–2279. [[CrossRef](#)]
18. Rudin, L.I.; Osher, S.; Fatemi, E. Nonlinear total variation based noise removal algorithms. *Phys. Nonlinear Phenom.* **1992**, *60*, 259–268. [[CrossRef](#)]
19. Nawaz, M. Variational Regularization for Multi-Channel Image Denoising. *Pak. J. Eng. Technol.* **2019**, *2*, 51–58.
20. Buades, A.; Coll, B.; Morel, J. A non-local algorithm for image denoising. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–26 June 2005; Volume 2, pp. 60–65. [[CrossRef](#)]
21. Foi, A.; Katkovnik, V.; Egiazarian, K. Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images. *IEEE Trans. Image Process.* **2007**, *16*, 1395–1411. [[CrossRef](#)]
22. Xu, J.; Zhang, L.; Zuo, W.; Zhang, D.; Feng, X. Patch Group Based Nonlocal Self-Similarity Prior Learning for Image Denoising. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Las Condes, Chile, 11–18 December 2015; pp. 244–252. [[CrossRef](#)]
23. Zoran, D.; Weiss, Y. From learning models of natural image patches to whole image restoration. In Proceedings of the 2011 International Conference on Computer Vision, Tokyo, Japan, 25–27 May 2011; pp. 479–486. [[CrossRef](#)]
24. Anwar, S.; Porikli, F.; Huynh, C.P. Category-Specific Object Image Denoising. *IEEE Trans. Image Process.* **2017**, *26*, 5506–5518. [[CrossRef](#)] [[PubMed](#)]
25. Yue, Z.; Yong, H.; Zhao, Q.; Meng, D.; Zhang, L. Variational Denoising Network: Toward Blind Noise Modeling and Removal. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 1690–1701.
26. Remez, T.; Litany, O.; Giryas, R.; Bronstein, A.M. Class-Aware Fully Convolutional Gaussian and Poisson Denoising. *IEEE Trans. Image Process.* **2018**, *27*, 5707–5722. [[CrossRef](#)] [[PubMed](#)]
27. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [[CrossRef](#)]
28. Dinh, L.; Krueger, D.; Bengio, Y. NICE: Non-linear Independent Components Estimation. *arXiv* **2014**, arXiv:cs.LG/1410.8516.
29. Abdelhamed, A.; Brubaker, M.A.; Brown, M.S. Noise Flow: Noise Modeling With Conditional Normalizing Flows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.
30. Huang, G.; Liu, Z.; Weinberger, K.Q. Densely Connected Convolutional Networks. *arXiv* **2016**, arXiv:1608.06993.
31. Ardizzone, L.; Lüth, C.; Kruse, J.; Rother, C.; Köthe, U. Guided Image Generation with Conditional Invertible Neural Networks. *arXiv* **2019**, arXiv:cs.CV/1907.02392.
32. Xiao, M.; Zheng, S.; Liu, C.; Wang, Y.; He, D.; Ke, G.; Bian, J.; Lin, Z.; Liu, T.Y. Invertible Image Rescaling. In Proceedings of the 16th European Conference Computer Vision (ECCV 2020), Glasgow, UK, 23–28 August 2020.
33. Liu, Z.; Luo, P.; Wang, X.; Tang, X. Deep Learning Face Attributes in the Wild. In Proceedings of the International Conference on Computer Vision (ICCV), Las Condes, Chile, 11–18 December 2015.
34. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
35. Nilsback, M.E.; Zisserman, A. Automated Flower Classification over a Large Number of Classes. In Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing, Bhubaneswar, India, 16–19 December 2008.
36. Lin, D.; Xu, G.; Wang, X.; Wang, Y.; Sun, X.; Fu, K. A Remote Sensing Image Dataset for Cloud Removal. *arXiv* **2019**, arXiv:cs.CV/1901.00600.
37. Xu, J.; Li, H.; Liang, Z.; Zhang, D.; Zhang, L. Real-world Noisy Image Denoising: A New Benchmark. *arXiv* **2018**, arXiv:1804.02603.
38. Abdelhamed, A.; Lin, S.; Brown, M.S. A High-Quality Denoising Dataset for Smartphone Cameras. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
39. Hurault, S.; Ehret, T.; Arias, P. EPLL: An Image Denoising Method Using a Gaussian Mixture Model Learned on a Large Set of Patches. *Image Process. Line* **2018**, *8*, 465–489. [[CrossRef](#)]
40. Mao, X.; Shen, C.; Yang, Y. Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections. In Proceedings of the Advances in Neural Information Processing Systems 29 (NIPS 2016), Barcelona, Spain, 5–10 December 2016.
41. Tai, Y.; Yang, J.; Liu, X.; Xu, C. MemNet: A Persistent Memory Network for Image Restoration. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
42. Kim, Y.; Soh, J.; Park, G.; Cho, N. Transfer Learning From Synthetic to Real-Noise Denoising With Adaptive Instance Normalization. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Seattle, WA, USA, 13–19 June 2020; pp. 3479–3489. [[CrossRef](#)]

- 
43. Chen, Y.; Pock, T. Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1256–1272. [[CrossRef](#)] [[PubMed](#)]
  44. Guo, S.; Yan, Z.; Zhang, K.; Zuo, W.; Zhang, L. Toward convolutional blind denoising of real photographs. *CVPR 2019, 2019*, 1712–1722.