Random Sub-Samples Generation for Self-Supervised Real Image Denoising

Yizhong Pan[1], Xiao Liu[1], Xiangyu Liao[1], Yuanzhouhan Cao[2], Chao Ren[1(✉)]
[1]College of Electronics and Information Engineering, Sichuan University, China
[2]School of Computer and Information Technology, Beijing Jiaotong University, China
{panyizhong, liux, liaoxiangyu1}@stu.scu.edu.cn, yzhcao@bjtu.edu.cn, chaoren@scu.edu.cn

Abstract

With sufficient paired training samples, the supervised deep learning methods have attracted much attention in image denoising because of their superior performance. However, it is still very challenging to widely utilize the supervised methods in real cases due to the lack of paired noisy-clean images. Meanwhile, most self-supervised denoising methods are ineffective as well when applied to the real-world denoising tasks because of their strict assumptions in applications. For example, as a typical method for self-supervised denoising, the original blind spot network (BSN) assumes that the noise is pixel-wise independent, which is much different from the real cases. To solve this problem, we propose a novel self-supervised real image denoising framework named Sampling Difference As Perturbation (SDAP) based on Random Sub-samples Generation (RSG) with a cyclic sample difference loss. Specifically, we dig deeper into the properties of BSN to make it more suitable for real noise. Surprisingly, we find that adding an appropriate perturbation to the training images can effectively improve the performance of BSN. Further, we propose that the sampling difference can be considered as perturbation to achieve better results. Finally we propose a new BSN framework in combination with our RSG strategy. The results show that it significantly outperforms other state-of-the-art self-supervised denoising methods on real-world datasets. The code is available at https://github.com/p1y2z3/SDAP.

1. Introduction

Image denoising is a hot topic in low-level vision tasks, aiming to obtain high-quality images from noisy versions. In recent years, learning-based methods have been widely used in image denoising with their superior performance. Thanks to the construction of paired image datasets, the supervised approaches [44, 2, 28, 11, 33, 34] can be efficiently implemented. In these approaches, it is a common practice

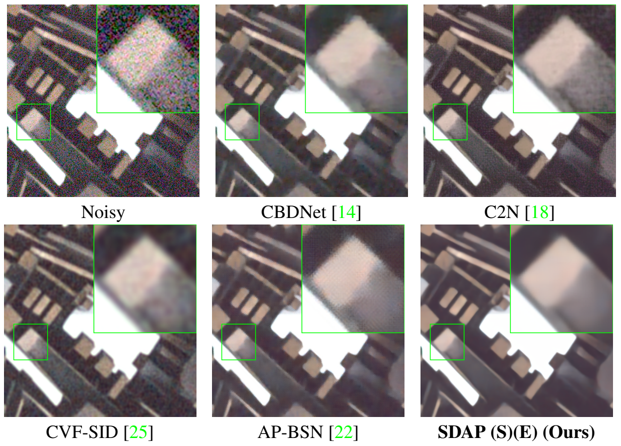


Figure 1: Real-world image denoising results on DND [27].

to synthesize paired datasets directly using a specific noise model, such as the signal-independent additive white Gaussian noise (AWGN) model.

However, the real images are captured by cameras with image signal processing (ISP) pipelines [4]. The noise is usually signal-dependent and spatially correlated in the real world. Moreover, ISP includes many non-linear operations that can complicate the noise distribution. Due to the influence of ISP pipelines within the camera, the noise distribution of real images is difficult to predict. Therefore, it is difficult and challenging for us to model the noise for real images. To overcome the problem of directly and explicitly modelling noise distribution, paired real-world datasets have been constructed by researchers. However, collecting paired noisy-clean images is difficult and time-expensive.

Recently, a series of unsupervised and self-supervised methods have been introduced. These methods do not require paired datasets for training, and thus further avoiding the difficulty of collecting real-world paired datasets for denoising. Among them, Noise2Void [20] proposes Blind-Spot Network (BSN) that can be trained with only single noisy images. Unfortunately, BSN has strict assumptions, which can only be applied to pixel-wise independent noise. AP-BSN [22] applies pixel-shuffle downsampling

✉Corresponding Author

(PD) strategy [47] to BSN, remove the spatial correlation in the real-world noise, and thus the real image can also meet the conditions for BSN to some extent. However, real noisy datasets usually have relatively limited numbers of samples. AP-BSN directly adopts the strategy of PD, which can only obtain s^2 sub-images for a noisy image (the stride factor of PD is s). This will inevitably lead to the insufficient samples, and make the training of BSN less effective. Therefore, the optimal performance of BSN cannot be achieved.

We observe that adding some perturbations to the training images can greatly expand the training data, which improves the performance of BSN. Further, we consider the samples difference as the perturbation. To obtain more random perturbations and more sub-samples, we propose random sub-samples generation (RSG) strategy to break the fixed sampling pattern of PD. Based on this, we propose a new cyclic sampling difference loss for BSN and a new BSN framework in combination with our RSG strategy. In summary, our contributions include:

- We provide a new idea about adding perturbations to the training data to improve the BSN performance. Then, we suggest that the sub-samples difference generated by sampling can be considered as perturbations for higher performance.
- We propose a new self-supervised framework for real image denoising with random sub-samples generation and cyclic sampling difference loss.
- Our method performs very favorably against state-of-the-art (SOTA) self-supervised denoising methods on real-world datasets.

2. Related Work

The image denoising task aims to restore a clean image from its noisy counterpart. There are two main types of non-learning-based image denoising: filtering-based methods and model-based methods. Filter-based methods involve using some artificially designed low-pass filter to remove image noise. Taking advantage of the condition of having many similar image blocks in the same image, noise can also be removed by the local similarity of image, such as NLM [5], BM3D [12], etc. Model-based methods involve modeling the distribution of natural images or noise and then using the model distribution before obtaining a clear image with an optimization algorithm, such as WNNM [13]. The learning-based methods are effective ways to reduce image noise. Usually, it can be divided into traditional and deep network-based methods. In recent years, due to the continuous development of the deep learning technology, deep network-based methods have achieved superior denoising results compared to previous methods, and thus they have become the mainstream denoising methods. Generally, deep network-based methods can be further classified according to their training manners.

2.1. Supervised Image Denoising

Most of the early works [24, 44, 31, 45, 46] assume that the noise is independent and uniformly distributed, and additive Gaussian white noise (AWGN) is usually used to model image noise and synthetic paired datasets. These methods achieve SOTA results on AWGN denoising tasks. However, the noise model in the real world is complex and unknown. Models trained with synthetic datasets are difficult to be applied to the real world. Recently, most of the methods [14, 2, 19, 41, 43, 6, 42, 29, 9] use produced real-world paired datasets for training. Due to the differences in camera parameters, these methods may not be widely used in the real world.

2.2. Pseudo-Supervised Image denoising

The supervised denoising methods require paired noisy-clean images, while the pseudo-supervised denoising methods relax the data requirements. Noise2Noise [23] proposes that pairs of noisy images can be used to train the network when the noise mean is zero. When some unpaired noisy-clean images are available, some methods [8, 7, 15, 18] learn the noise distribution of noisy images by generative adversarial networks (GAN). Then, clean images are used to generate pseudo-paired noisy-clean images to train the denoising network. However, these methods are still difficult to be applied because the scene distribution of noisy images often does not match the available clean images [18].

2.3. Self-Supervised Image Denoising

The self-supervised denoising methods require single noisy images for training. Noise2Void [20] finds that masking a portion of pixels in noisy images for pairing can train the denoising network, and thus proposes a BSN for self-supervised denoising. Laine19 [21] and D-BSN [36] further optimize the BSN and improve its performance. Noise2Same [37] and Blind2Unblind [35] propose new denoising losses for self-supervised training through mask-based blind-spot methods. Neighbor2Neighbor [17] samples the noisy image into two similar sub-images to form a noisy-noisy pair for training. CVF-SID [25] can disentangle the clean image, signal-dependent and signal-independent noises from the real-world noisy input via various self-supervised training objectives. AP-BSN [22] combines PD and BSN to process real-world sRGB noisy images and employs asymmetric PD stride factors for training and inference. However, current self-supervised methods for real noise removal usually still require a large number of noisy images and are challenging to train with less training data to achieve superior performance.

3. Proposed Method

3.1. Revisiting Noise2Void

Noise2Void (N2V) [20] requires only single noisy images for training, and it assumes that the noise n is pixel-wise independent (the noises at different pixel positions are independent) given the clean image x. y is the noisy image, i.e. $y = x + n$. Due to the local correlation of the pixels within an image, clean central pixels can be estimated from the surrounding noisy pixels. Therefore, using the pixel of the noisy image as a target for its surrounding pixels will not obtain the noisy pixel. Thus, N2V can train the denoising network in a self-supervised approach. Specifically, N2V loss L_{N2V} can be written as follows:

$$L_{N2V} = E_y\{\|f(y_r) - y_c\|_2^2\}, \tag{1}$$

where $f(\cdot)$ is the normal denoising network, y_r is the patch of y centered on y_c with the same dimensions as the network receptive field. Note that y_r does not contain y_c.

To facilitate the implementation of training with L_{N2V}, N2V proposes BSN $B(\cdot)$ capable of masking centroids. The noisy images used to train the BSN must satisfy the BSN assumption that the noise is pixel-wise independent and zero-mean. BSN can be trained by optimizing the following loss L_{BSN}:

$$L_{BSN} = E_y\{\|B(y) - y\|_2^2\}. \tag{2}$$

L_{N2V} and L_{BSN} are equivalent for training BSN. For a known noise model (e.g. AWGN model), we can synthesize an infinite number of different noisy images. Thus, BSN can be easily trained. In contrast, since real image datasets often have a limited number of captured images, it is difficult to train BSN directly for real image denoising. Moreover, real images do not satisfy the pixel-wise independent noise assumptions of BSN. Therefore, the limitations of L_{BSN} are significant, and it is not appropriate to use real images directly for BSN.

3.2. AP-BSN and Its Limitation

To make BSN better for the spatially correlated real image noise, AP-BSN [22] introduces PD [47] to break the spatial correlation among real noisy pixels, leading to near pixel-wise independent noise. Therefore, the real image after PD can be considered to meet the BSN assumption [22], and thus can be used as the input for BSN. The original AP-BSN loss is written as $\left\|PD^{-1}(B(PD(y))) - y\right\|_1$, where PD is the PD operator and PD^{-1} is the inverse operator. To facilitate the analysis, we further write it as:

$$L_{AP\text{–}BSN} = E_y\{\|B(PD(y)) - PD(y)\|_1\}. \tag{3}$$

It can be observed that the pattern of pairing is fixed in Eq. (3) (i.e. fixed $PD(y)$ to $PD(y)$). We consider that the training of fixed patterns does not affect the performance of BSN when the sample number is large enough. In contrast, it is well known that insufficient sample number may lead to overfitting [39]. Overfitting of noisy-clean pairing training can only have a superior effect on the training datasets and fail on other datasets. However, BSN requires noisy-noisy pairs for training. The overfitting of BSN will make the denoising result fit the noisy image and affect the denoising performance. For synthetic noise, because the noise modelling can generate an infinite number of noisy images, BSN can avoid fitting the noise, resulting in a clean image. But when BSN is used for real noise, real datasets tend to have a much smaller sample number than synthetic datasets. In this case, BSN may fail to get high-quality clean pixels.

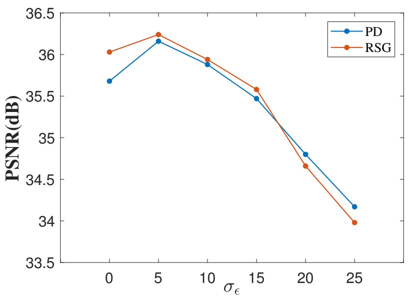
Dataset	Set12	BSD68	Urban100
Fixed seed	28.91/0.8192	27.26/0.7477	27.66/0.8172
Random seed	29.36/0.8399	27.72/0.7728	28.81/0.8632

Table 1: The effect of different BSN training strategies on PSNR (dB)/SSIM.

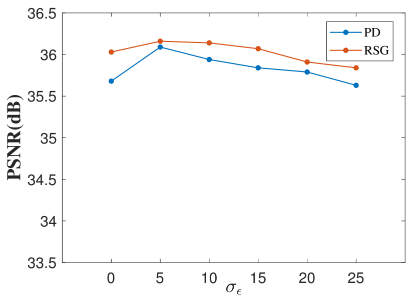
To verify the effect of the number of training images on the denoising results, we train BSN with synthetic images. In this way, we can synthesize any number of noisy images. We employ DIV2K dataset [32] and add Gaussian noise with a level $\sigma = 25$ to synthesize noisy images. Two ways are adopted to train BSN with L_{BSN}: (1) adding noise with fixed seed in each epoch, and (2) adding noise with random seed in each epoch. Then, we perform denoising experiments on the test datasets generated with Gaussian noise (Set12 [44], BSD68 [30], and Urban100 [16]). Table 1 shows the PSNR(dB) and SSIM results in the test datasets for different training strategies. We observe that when a limited number of noisy images (noisy images with fixed seeds) are used to train the BSN , the denoising performance degrades significantly. Therefore, the introduction of PD is practical. Still, considering that the training data of real images are often limited, the direct use of $L_{AP\text{–}BSN}$ may affect the denoising performance for real image denoising.

3.3. Perturbation in BSN

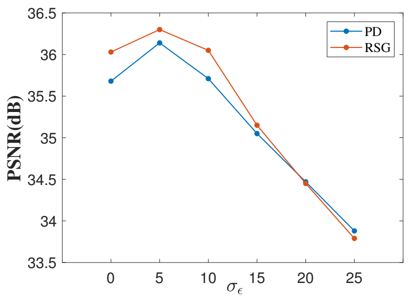
In order to solve the above problems, we suggest adding perturbations to Eq. (3) to increase the number of training data and avoid fitting to the fixed pattern. We should ensure that the sub-images as inputs or targets remain consistent with the BSN assumption after adding perturbations. This is necessary for training the BSN using these sub-images for denoising purposes. A simple solution is using AWGN as the perturbation. Suppose ϵ, ϵ_1, and ϵ_2 are all perturbations of a Gaussian distribution with mean 0 and variance σ_ϵ^2. We propose three perturbation BSN losses $L_{PBSN_{1,2,3}}$



(a) L_{PBSN_1}



(b) L_{PBSN_2}



(c) L_{PBSN_3}

Figure 2: Ablation study of σ_ϵ in $L_{PBSN_{1,2,3}}$ for training on the SIDD validation dataset [1].

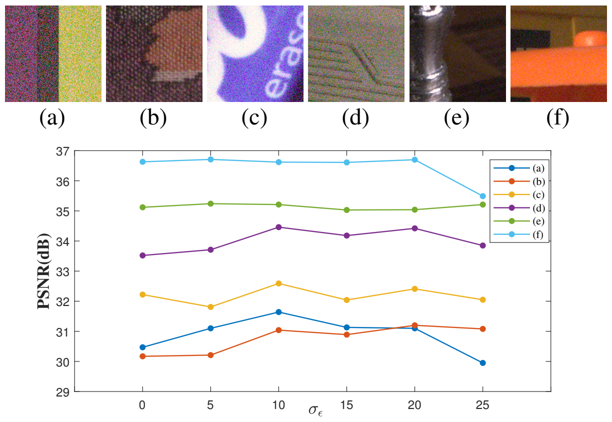


Figure 3: Ablation study of σ_ϵ in L_{PBSN_2} for training in different images.

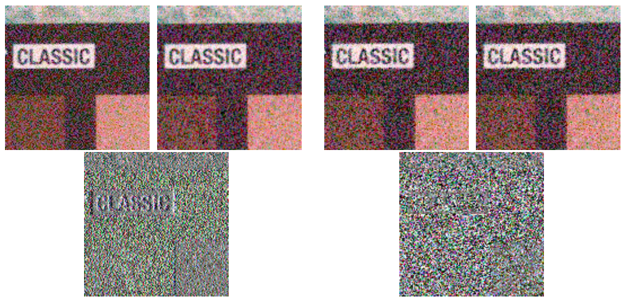
as follows:

$$
\begin{aligned}
L_{PBSN_1} &= E_y\{\|B(PD(y)+\epsilon) - PD(y)\|_1\},\\
L_{PBSN_2} &= E_y\{\|B(PD(y)) - (PD(y)+\epsilon)\|_1\},\\
L_{PBSN_3} &= E_y\{\|B(PD(y)+\epsilon_1) - (PD(y)+\epsilon_2)\|_1\}.
\end{aligned} \tag{4}
$$

We test the effect of different variances of ϵ or (ϵ_1, ϵ_2) in Eq. (4) on the BSN training, and the results are shown in Figures 2 and 3. We observe that adding suitable perturbations can achieve improvements in BSN performance. Therefore, each of the losses in Eq. (4) fits our vision and can obtain a better BSN by training. In Section 4.3, we analyze the influence of perturbations in detail. The results in Figure 2 show that different perturbation levels' effects on BSN performance vary. The results in Figure 3 show that the appropriate optimal perturbation for different levels of noisy images also varies. Therefore, it is a challenge to choose the appropriate perturbation.

3.4. Sampling Difference as Perturbation

PD can produce a series of similar sub-images ($PD_1(y)$, $PD_2(y)$, ..., where $PD_i(\cdot)$ denotes the i-th sub-image obtained after PD). Since the PD sampling process does not



(a) PD (b) RSG

Figure 4: Results of sampling by different methods. The second row is the sampling difference obtained by subtracting the two sub-samples in the first row.

overlap, the pixels in the sub-images that are at the same position are not located at the same position in the original image. As a result, there are certain differences between these sub-images, which we refer to as sampling difference.

Based on the previous analysis, it is evident that adding perturbation for training can enhance the performance of BSN, but how to add the appropriate perturbations is a challenge. Since there exists a sampling difference among the sub-images acquired through sampling, this difference can be viewed as a type of perturbation. First of all, according to [22], the sub-images obtained through PD sampling are naturally in line with the BSN assumption. Next, the difference between $PD_1(y)$ and $PD_2(y)$ can be considered as the perturbation, so we do not need to add additional perturbations as in Eq. (4). Therefore, $PD_1(y)$ and $PD_2(y)$ can be used as the input and target in the BSN training, respectively. Finally, we propose a new sampling difference BSN loss L_{SDBSN} as follows:

$$L_{SDBSN} = E_y\{\|B(PD_1(y)) - PD_2(y)\|_1\}. \tag{5}$$

3.5. Random Sub-Samples Generation

The sampling difference generated by the PD strategy can be obtained by subtracting the two sub-images generated by the PD. The sub-images and the sampling difference

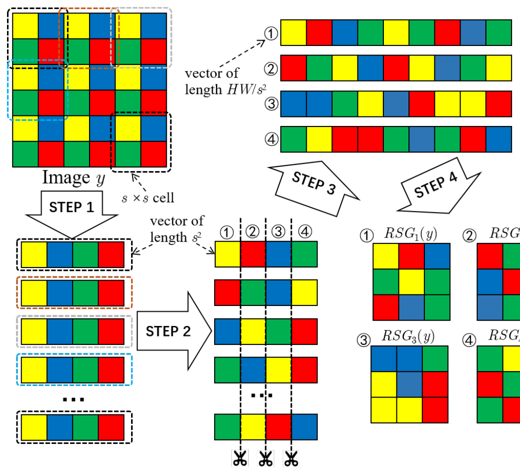
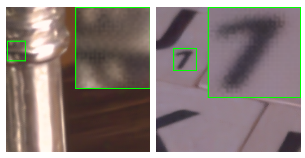
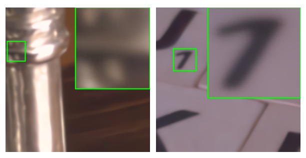


Figure 5: Example of generating sub-samples with RSG.



(a) SDAP



(b) SDAP (E)

Figure 6: Visual comparison between SDAP and SDAP (E).

generated by PD are shown in Figure 4a. It can be observed that the sampling difference generated by PD is similar to the gradient map. Meanwhile, the PD strategy with a stride factor of s can only generate s^2 fixed sub-images for the same image, which still faces the data-hungry issue for BSN training. To obtain more sub-samples and make the sampling difference closer to random perturbation, we propose the random sub-samples generation strategy (RSG).

The diagram of sub-images generation with the RSG is shown in Figure 5. Assume that the noisy image (y) length and width are H and W, respectively. The details of RSG are described below:

1. **Divide.** The image y is divided into HW/s^2 non-overlapping cells with size $s \times s$. Then, we stretch the cells into vectors of length s^2. For better visualization, we set $s = 2$ in Figure 5.

2. **Shuffle.** Randomly shuffle the elements within each vector. Since this process is strictly random, the sub-samples obtained are different each time.

3. **Reform.** Take the elements at the same position of each vector to form s^2 new vectors of length HW/s^2.

4. **Reshape.** Reshape each vector into a sub-sample of size $[W/s]\times[H/s]$. In this way, s^2 similar sub-samples ($RSG_1(y)$, $RSG_2(y)$, ... , $RSG_{s^2}(y)$) are obtained.

To maximize the variance of each sub-sample, our sampling is non-overlapping.

The sub-samples generated by the RSG strategy and their sampling difference is shown in Figure 4b. It can be observed that the use of RSG does make the sampling difference more random. Furthermore, we provide detailed analysis for this phenomenon. Both PD and RSG sample the image directly, so the calculation of the sampling difference can be considered as the deviation in the direction of the line connecting the sampling pixels of the two sub-samples. Since each pixel of the PD sub-sample has a fixed position in the original image, the difference has a fixed offset direction. Due to random sampling, the sampling difference of RSG sub-samples are not fixed in the direction of each pixel deviation, making the difference more random. In addition, as can also be seen from Figure 5, PD is equivalent to obtaining four fixed sub-images of yellow, blue, green and red. While RSG obtains random sub-images of ①②③④ with different results for each sampling, which can provide a large number of samples for training. Therefore, RSG strategy can also be considered as a solution to the data-hungry issue of training BSN with real noisy images.

3.6. Cyclic Sampling Difference BSN Loss

According to Eq. (5), only two sub-samples are needed. However, because multiple sub-samples are generated by RSG, using the whole sub-images instead of only two samples in training can better utilize the features of the generated sub-samples, leading to higher performance. Moreover, if only two sub-samples are used for training, the loss of each training iteration will fluctuate seriously, and the training of BSN cannot converge. To preserve the stability of training and to make full use of each sample, we introduce the strategy of cyclic and RSG to propose a new cyclic samples difference BSN loss L_{CSDBSN}:

$$L_{CSDBSN} = \sum_{i=1}^{s^2} \underbrace{\|B(RSG_i(y)) - RSG_{i+1}(y)\|_1}_{cross-pairing} \tag{6}$$
$$(RSG_{s^2+1}(y) = RSG_1(y)).$$

Since the sub-samples are generated randomly, the training data is also random. The target corresponding to each input in L_{CSDBSN} is different. Therefore, two sub-samples are avoided to form a fixed mapping during the cycle. This cyclic loss has the following merits: 1) it imposes constraints on the full pixel of the original noisy image; 2) it ensures that all the sub-samples generated by RSG are well exploited; 3) it makes the training of BSN more robust. Figure 7 visually illustrates our training scheme with the cyclic sampling difference BSN loss.

3.7. Proposed SDAP Framework

For the training stage, we first sample the noisy images into s^2 sub-samples by RSG. Then, we denoise sub-samples

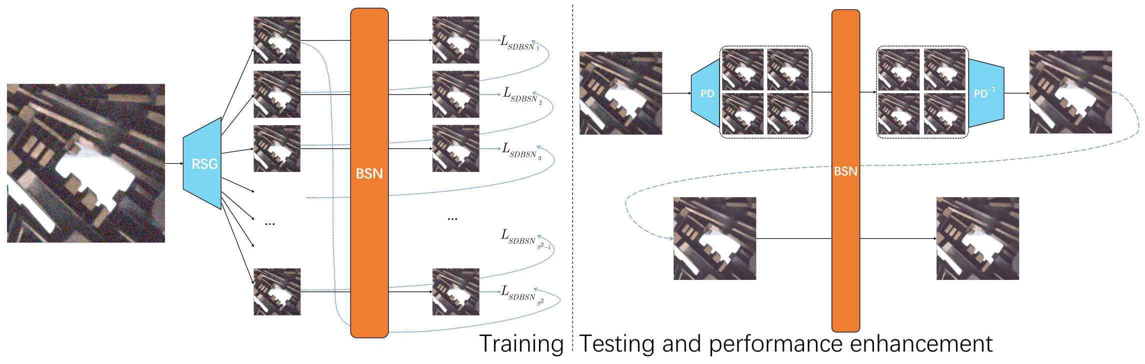


Figure 7: Overview of our proposed SDAP framework.

by BSN. The loss is calculated by cross-pairing the sub-samples after denoising with those before denoising. Finally, the above steps are iterated, and the loss function is updated to optimize the BSN until it converges. The training process is shown in the left half of Figure 7.

For the testing stage, due to the randomness of the single RSG, the test results are not fixed. Since the sampling pattern of PD is fixed, the test results are stable for testing. Therefore, for default setting, we directly adopt the strategy of PD. We denoise the sub-images after PD and then stitch them to obtain the final denoised image.

The PD strategy is introduced in the testing stage to break the spatial correlation of real-world noise. However, this also makes the pixels of the denoised image discontinuous and produces checkerboard artifacts as shown in Figure 6a. To make the denoised image have better visual performance, we propose to denoise the initial denoised image again by BSN to remove the unpleasing artifacts. This performance enhancement method is denoted by "SDAP (E)" and is shown in the right half of Figure 7. The comparison of the final visual effect is shown in Figure 6b. The results show that our proposed test strategy can remove checkerboard artifacts and make the images more natural. More details about PD and RSG for testing are provided in **Supplementary Material**.

4. Experiments

4.1. Implementation Details

We use the general BSN architecture [36, 22] in our method. According to [22], we empirically set the stride factor s of RSG to 5 for training. To ensure the stability of the test, we still use the PD with stride factor 2 for testing and ablation study. During the training, we use Adam as the optimizer with default settings. The initial learning rate is 0.0001, batch size and patch size are set to 16 and 160×160 respectively, and epochs are taken as 15. 25600 patches are used in one epoch. To fine-tune the network, BSN continues to be optimised for 10 epochs. Meanwhile, we decay the learning rate by a factor of 10, reduce batch size to 8, and adjust patch size to 250×250. We use PyTorch to implement our BSN and train it on an NVIDIA RTX 3090 GPU.

4.2. Dataset

SIDD [1]. The SIDD dataset uses five representative mobile phone cameras to capture approximately 300,000 noisy images for 10 scenes under different lighting conditions and obtain the corresponding relatively clean images by specific statistical methods. The SIDD dataset provides an SIDD Medium dataset of 320 data pairs for training, an SIDD validation dataset of 1280 image blocks of size 256×256 for validation, and an SIDD benchmark dataset of 40 high-resolution noisy images for testing.

DND [27]. The DND dataset contains 50 high-resolution test image pairs taken by consumer-grade cameras of various sensor sizes. However, because the images are too large, DND cropped all the images to 512×512 image blocks, giving 1000 image blocks for testing. The DND dataset does not provide training images, and the clean images of the test dataset are not publicly available.

Note that unless otherwise stated, we used SIDD Medium dataset for training in the following experiments.

4.3. Analyzing Perturbation

We first validate the effect of the perturbation in $L_{AP\text{–}BSN}$ for real-world sRGB image denoising. We train the BSN by $L_{PBSN_{1,2,3}}$ with different variances of ϵ, i.e., $\sigma_\epsilon \in \{0, 5, 10, 15, 20, 25\}$. Meanwhile, we also replace PD with RSG for a similar set of experiments. Figure 2 shows the PSNR(dB) results in the SIDD validation set.

We observe that BSN performs best on SIDD validation dataset when $\sigma_\epsilon = 5$ for both PD and RSG. After this, the performance decreases as σ_ϵ increases. Specifically, for $L_{PBSN_{1,2,3}}$, since the perturbation is added to the training data, when the perturbation is too large, it will significantly change the noise distribution of the training image, making it difficult for the BSN to adapt to the original noise. Therefore, the denoising performance of BSN decreases significantly when $\sigma_\epsilon > 15$. Also, since RSG and perturbation

Type of supervision	Training data	Method	SIDD validation	SIDD benchmark	DND benchmark
Non-learning based	-	BM3D [12]	31.75/0.7061	25.65/0.685	34.51/0.8507
		WNNM [13]	26.31/0.5240	25.78/0.809	34.67/0.8646
Supervised	Paired noisy-clean	DnCNN [44]	26.20/0.4414	23.66/0.583	32.43/0.7900
		TNRD [10]	26.99/0.7440	24.73/0.643	33.65/0.8306
		CBDNet [14]	30.83/0.7541	33.28/0.868	38.06/0.9421
		RIDNet [2]	38.76/0.9132	37.87/0.943	39.25/0.9528
		VDN [40]	39.29/0.9109	39.26/0.955	39.38/0.9518
		Zhou *et al.* [47]	-	34.00/0.898	38.40/0.945
		DeamNet [28]	39.40/0.9169	39.35/0.955	39.63/0.9531
Pseudo-supervised	Unpaired noisy-clean	GCBD [8]	-	-	35.58/ 0.9217
		D-BSN [36]	-	-	37.93/0.9373
		C2N [18]	35.36/0.8901	35.35/0.937	37.28/0.9237
	Paired noisy-noisy	R2R [26]	35.04/0.8440	34.78/0.898	37.61/0.9368
Self-supervised	Single noisy	N2V [20]	29.35/0.6510	27.68/0.668	-
		N2S [3]	30.72/0.7870	29.56/0.808	-
		NAC [38]	-	-	36.20/0.9252
		Neighbor2Neighbor [17]	28.00/0.5890	27.96/0.670	31.40/0.7880
		CVF-SID [25]	34.17/0.8719	34.71/0.917	36.50/0.9233
		AP-BSN [22]	34.46/0.8501	34.90/0.900	37.46/0.9244
		SDAP (Ours)	**36.58/0.8630**	**36.54/0.919**	**37.71/0.9278**
		SDAP (S) (Ours)	**36.71/0.8640**	**36.68/0.919**	**38.18/0.9322**
		SDAP (E) (Ours)	**37.30/0.8937**	**37.24/0.936**	**37.86/0.9366**
		SDAP (S)(E) (Ours)	**37.55/0.8943**	**37.53/0.936**	**38.56/0.9402**

Table 2: Quantitative PSNR(dB)/SSIM Results on SIDD and DND Dataset.

have similar roles, both are designed to obtain more training data. When the perturbation is too large, the role of RSG is difficult to be reflected. Therefore, when the perturbation is small, the RSG strategy outperforms the PD strategy. When the perturbation is large, the RSG strategy and the PD strategy are used for BSN to obtain similar denoising results.

In addition, we note that the most suitable perturbation standard deviation σ_ϵ is different for each of these images in Figure 3. Nevertheless, when we consider the sampling difference as the perturbation, the BSN performs the best regardless of the data. In Section 4.5, we analyze the sampling difference as perturbation in detail. Therefore, Eq. (6) is the final loss function of our proposed method.

4.4. Denoising of Real Image

We use SIDD validation, SIDD benchmark, and DND benchmark datasets to evaluate the performance of our SDAP in real-world denoising tasks. For the SIDD validation dataset, because the paired noisy-clean images are provided, we can directly test the PSNR/SSIM results. For the SIDD and DND benchmarks, we submit the denoised images to websites for server-side evaluation to obtain the final results (PSNR/SSIM values). To evaluate our model on the three different datasets, we adopt two different strategies for training.

Training on SIDD Medium dataset. We train our self-supervised method with noisy images from the SIDD Medium dataset. Then, the obtained model is evaluated on different datasets. This training method is represented as "SDAP" in Table 2.

Training on test dataset. Since our method is fully self-supervised [25], instead of using the fixed SIDD Medium dataset for training, we can use the test images for self-supervised training as well. Therefore, we train our method on three test datasets in self-supervised manner, respectively. The trained models are evaluated with the corresponding test datasets. Since the noise distribution of training and test images are identical, this strategy can obtain a BSN model with better performance. Fully self-supervised training method is represented as "SDAP (S)" in Table 2.

It should be noted that "(S)" denotes the method to improve the performance in training, while "(E)" denotes the way to improve the performance in testing. Thus "SDAP (S)(E)" is the way in which the proposed method can achieve the best performance.

The results in Table 2 show that this method achieves significantly better results than some traditional methods, such as BM3D, on the SIDD Benchmark test data. Even when compared with some supervised deep learning-based

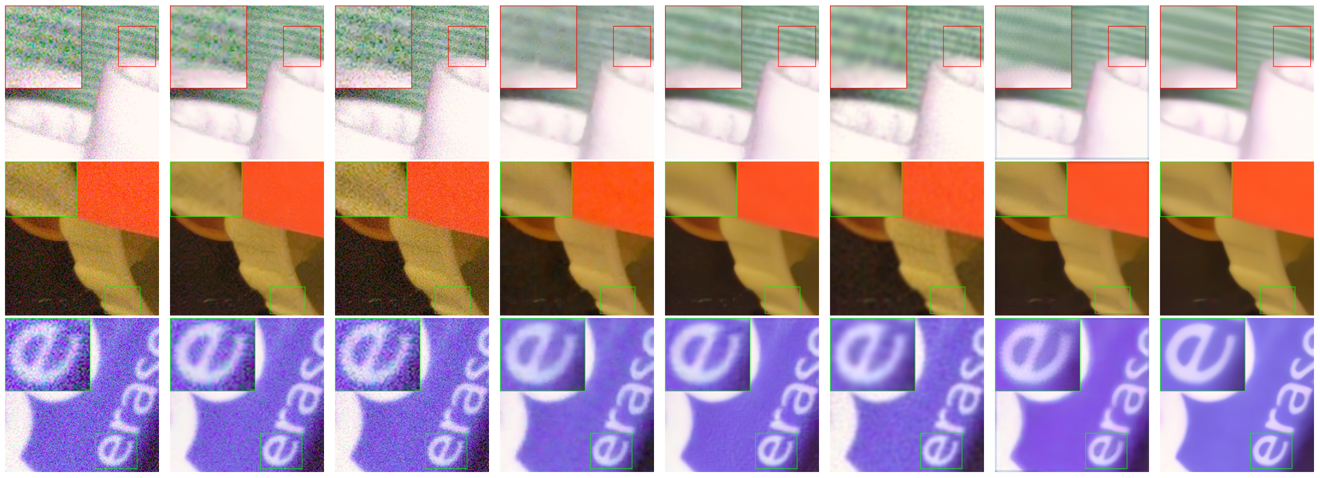


Figure 8: Visual comparison of our method against other competing methods on the SIDD validation dataset [1].

PD	RSG	$L_{AP\text{–}BSN}$	L_{CSDBSN}	PSNR(dB)/SSIM
✓	-	✓	-	35.68/0.8382
✓	-	-	✓	36.19/0.8472
-	✓	✓	-	36.03/0.8533
-	✓	-	✓	**36.58/0.8630**

Table 3: Investigation of RSG and L_{CSDBSN} on SIDD Validation Dataset.

image denoising methods, this method achieves superior denoising results. For example, the PSNR and SSIM results of this method are 3.40dB and 0.051 higher compared with CBDNet. When we compared with the SOTA self-supervised image denoising approaches, our method also has obvious advantages. For example, compared with AP-BSN, which has the same network structure as us, SDAP achieves 1.78dB and 0.019 increases in PSNR and SSIM metrics, respectively. The performance of SDAP on the DND dataset has similar findings as those on SIDD. Figures 1 and 8 provides visual comparisons of several methods, and the results further validate the visual superiority of our method to other methods.

4.5. Ablation Study

We examine two major determinants of our model: a) RSG; b) L_{CSDBSN}. If there is a "✓" in the column of L_{CSDBSN} (or L_{APBSN}), it means that L_{CSDBSN} (or L_{APBSN}) is used for training, and vice versa. If there is a "✓" in the column of RSG, it means that the RSG strategy is used for training, and we replace all PD in the loss with RSG. If there is a "✓" in the column of PD, it means that the PD strategy is used for training, and we replace all RSG in the loss with PD.

Study of RSG. The performance of training with RSG in Table 3 is always higher than training with PD. A comparison between the first and third rows of Table 3 shows that replacing PD with RSG increases the PSNR/SSIM result by 0.35dB/0.0151. The comparison between the second and fourth rows shows a similar conclusion. These results verify the effectiveness for RSG.

Study of L_{CSDBSN}. Rows 2 and 4 of Table 3 show the effect of our proposed L_{CSDBSN}, where our proposed loss improves the denoising performance by 0.51dB/0.0090 in the case of training with PD and 0.55dB/0.0097 in the case of training with RSG. It is worth noting that, irrespective of whether PD or RSG is used in L_{CSDBSN} for BSN training, the performance of BSN is superior to that achieved by directly adding Gaussian perturbations in $L_{AP\text{–}BSN}$ (i.e., using L_{CSDBSN} for training is better than using $L_{PBSN_{1,2,3}}$).

5. Conclusion

In this paper, we first analyze the reasons for the limited performance of BSN when used for real image denoising. Based on this, we propose to add perturbations to the training data and consider sampling difference as perturbation. Further, we propose SDAP framework with random sub-samples generation and cyclic sampling difference loss. Our SDAP does not require clean images for training and outperforms existing pseudo-supervised/self-supervised methods. We believe that our approach can provide promising inspirations for various self-supervised real-world denoising methods.

Acknowledgement. This work was supported by the National Natural Science Foundation of China under Grant 62171304.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR*, pages 1692–1700, 2018.

[2] Saeed Anwar and Nick Barnes. Real image denoising with feature attention. In *ICCV*, pages 3155–3164, 2019.

[3] Joshua Batson and Loic Royer. Noise2self: Blind denoising by self-supervision. pages 524–533. PMLR, 2019.

[4] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. Unprocessing images for learned raw denoising. In *CVPR*, pages 11036–11045, 2019.

[5] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *CVPR*, volume 2, pages 60–65. Ieee, 2005.

[6] Yuanhao Cai, Xiaowan Hu, Haoqian Wang, Yulun Zhang, Hanspeter Pfister, and Donglai Wei. Learning to generate realistic noisy images via pixel-level noise-aware adversarial training. In *NeurIPS*, volume 34, pages 3259–3270, 2021.

[7] Sungmin Cha, Taeeon Park, and Taesup Moon. Gan2gan: Generative noise learning for blind image denoising with single noisy images. *arXiv preprint arXiv:1905.10488*, 3, 2019.

[8] Jingwen Chen, Jiawei Chen, Hongyang Chao, and Ming Yang. Image blind denoising with generative adversarial network based noise modeling. In *CVPR*, pages 3155–3164, 2018.

[9] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. *arXiv preprint arXiv:2204.04676*, 2022.

[10] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE TPAMI*, 39(6):1256–1272, 2016.

[11] Shen Cheng, Yuzhi Wang, Haibin Huang, Donghao Liu, Haoqiang Fan, and Shuaicheng Liu. Nbnet: Noise basis learning for image denoising with subspace projection. In *CVPR*, pages 4896–4906, 2021.

[12] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE TIP*, 16(8):2080–2095, 2007.

[13] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *CVPR*, pages 2862–2869, 2014.

[14] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *CVPR*, pages 1712–1722, 2019.

[15] Zhiwei Hong, Xiaocheng Fan, Tao Jiang, and Jianxing Feng. End-to-end unpaired image denoising with conditional adversarial networks. In *AAAI*, volume 34, pages 4140–4149, 2020.

[16] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015.

[17] Tao Huang, Songjiang Li, Xu Jia, Huchuan Lu, and Jianzhuang Liu. Neighbor2neighbor: Self-supervised denoising from single noisy images. In *CVPR*, pages 14781–14790, 2021.

[18] Geonwoon Jang, Wooseok Lee, Sanghyun Son, and Kyoung Mu Lee. C2n: Practical generative noise modeling for real-world denoising. In *ICCV*, pages 2350–2359, 2021.

[19] Yoonsik Kim, Jae Woong Soh, Gu Yong Park, and Nam Ik Cho. Transfer learning from synthetic to real-noise denoising with adaptive instance normalization. In *CVPR*, pages 3482–3492, 2020.

[20] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void-learning denoising from single noisy images. In *CVPR*, pages 2129–2137, 2019.

[21] Samuli Laine, Tero Karras, Jaakko Lehtinen, and Timo Aila. High-quality self-supervised deep image denoising. *NeurIPS*, 32, 2019.

[22] Wooseok Lee, Sanghyun Son, and Kyoung Mu Lee. Ap-bsn: Self-supervised denoising for real-world images via asymmetric pd and blind-spot network. In *CVPR*, pages 17725–17734, 2022.

[23] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. pages 2965–2974. PMLR, 2018.

[24] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *NeurIPS*, 29, 2016.

[25] Reyhaneh Neshatavar, Mohsen Yavartanoo, Sanghyun Son, and Kyoung Mu Lee. Cvf-sid: Cyclic multi-variate function for self-supervised image denoising by disentangling noise from image. In *CVPR*, pages 17583–17591, 2022.

[26] Tongyao Pang, Huan Zheng, Yuhui Quan, and Hui Ji. Recorrupted-to-recorrupted: unsupervised deep learning for image denoising. In *CVPR*, pages 2043–2052, 2021.

[27] Tobias Plotz and Stefan Roth. Benchmarking denoising algorithms with real photographs. In *CVPR*, pages 1586–1595, 2017.

[28] Chao Ren, Xiaohai He, Chuncheng Wang, and Zhibo Zhao. Adaptive consistency prior based deep network for image denoising. In *CVPR*, pages 8596–8606, 2021.

[29] Chao Ren, Yizhong Pan, and Jie Huang. Enhanced latent space blind model for real image denoising via alternative optimization. In *NeurIPS*, 2022.

[30] Stefan Roth and Michael J Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–229, Jan. 2009.

[31] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, pages 4539–4547, 2017.

[32] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPRW*, pages 114–125, Jul. 2017.

[33] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxim: Multi-axis mlp for image processing. In *CVPR*, pages 5769–5780, 2022.

[34] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. Uformer: A gen-

eral u-shaped transformer for image restoration. In *CVPR*, pages 17683–17693, 2022.

[35] Zejin Wang, Jiazheng Liu, Guoqing Li, and Hua Han. Blind2unblind: Self-supervised image denoising with visible blind spots. In *CVPR*, pages 2027–2036, 2022.

[36] Xiaohe Wu, Ming Liu, Yue Cao, Dongwei Ren, and Wangmeng Zuo. Unpaired learning of deep image denoising. In *ECCV*, pages 352–368. Springer, 2020.

[37] Yaochen Xie, Zhengyang Wang, and Shuiwang Ji. Noise2same: Optimizing a self-supervised bound for image denoising. *NeurIPS*, 33:20320–20330, 2020.

[38] Jun Xu, Yuan Huang, Ming-Ming Cheng, Li Liu, Fan Zhu, Zhou Xu, and Ling Shao. Noisy-as-clean: Learning self-supervised denoising from corrupted image. *IEEE TIP*, 29:9316–9329, 2020.

[39] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing, 2019.

[40] Zongsheng Yue, Hongwei Yong, Qian Zhao, Deyu Meng, and Lei Zhang. Variational denoising network: Toward blind noise modeling and removal. *NeurIPS*, 32, 2019.

[41] Zongsheng Yue, Qian Zhao, Lei Zhang, and Deyu Meng. Dual adversarial network: Toward real-world noise removal and noise generation. In *ECCV*, pages 41–58. Springer, 2020.

[42] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, pages 5728–5739, 2022.

[43] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *CVPR*, pages 14821–14831, 2021.

[44] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE TIP*, 26(7):3142–3155, 2017.

[45] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE TIP*, 27(9):4608–4622, 2018.

[46] Hongyi Zheng, Hongwei Yong, and Lei Zhang. Deep convolutional dictionary learning for image denoising. In *CVPR*, pages 630–641, 2021.

[47] Yuqian Zhou, Jianbo Jiao, Haibin Huang, Yang Wang, Jue Wang, Honghui Shi, and Thomas Huang. When awgn-based denoiser meets real noises. In *AAAI*, volume 34, pages 13074–13081, 2020.