Synthetic2Real: Learning to Generate Realistic Noisy Images via Pixel-Level Adversarial Training

Anonymous NeurIPS 2021 Submission
Paper ID 446

Abstract

Existing deep learning image denoising methods require a large amount of noisy-clean image pairs for supervision. Nonetheless, capturing a real noisy-clean dataset is an unacceptable expensive and cumbersome procedure. Although synthetic noisy images are easy to obtain, there still remains a domain discrepancy between synthetic and real noisy datasets. To narrow this domain discrepancy, we propose a novel framework, namely Pixel-level Noise Generative Adversarial Network (PNGAN) learning to generate more realistic noisy images from their synthetic counterparts. PNGAN treats each pixel as a random variable and aims to estimate its probability distribution. Specifically, PNGAN composes a noise Generator, a denoise Discriminator, and a classification Discriminator. Firstly, the noise Generator maps the synthetic noisy image to its generated counterpart. Secondly, the generated and real noisy images are fed into the denoise Discriminator to minimize the distance between their denoised versions and thus realize soft pixel-wise alignment. Finally, the denoise and classification Discriminators establish an effective pixel-level supervision for the generated noisy image so as to encourage its distribution to approach that of the real noisy version. Comprehensive experiments clearly demonstrate that images generated by PNGAN are much closer to their real noisy counterparts than synthetic images. Moreover, a series of models trained with generated noisy images achieve state-of-the-art (SOTA) results on four benchmarks: SIDD, DND, PolyU, and Nam. Surprisingly, our method ranks the first place in DND RGB denoising competition. Codes and models will be released for research.

1 Introduction

Image denoising is an important yet challenging problem in low-level vision and image processing. It aims to restore a clean image from the noisy counterpart. Noise removal methods are mainly divided into two categories. Most traditional approaches belong to the first category, concentrating on designing a rational maximum a posteriori (MAP) model, containing regularization and fidelity terms, from a Bayesian perspective [1]. Some image priors like low-rankness [2, 3, 4], sparsity [5], and non-local similarity [6, 7] are exploited to customize a better rational MAP model. However, these hand-crafted methods are inferior in representing capacity. With the development of deep learning, image denoising has witnessed significant progress. Deep Convolutional Neural Network (CNN) applies a powerful learning model to eliminate noise and has achieved promising performance [8, 9, 10, 11, 12]. These deep CNN denoisers rely on a large-scale dataset of noisy-clean image pairs. Nonetheless, collecting even small datasets is extremely tedious and labor-intensive. Specifically, the process of acquiring noisy-clean image pairs is to take hundreds of noisy images of the same scene and average them to get the clean image. To obtain more image pairs, researchers try to synthesize noisy images.

In particular, there are two common settings for synthesizing noisy images. As shown in Fig 1 (a), setting1 directly adds the Additive White Gaussian Noise (AWGN) with the clean RGB image. For a long time, single image denoising [13, 14, 15, 16, 17, 10] is performed with setting1. Nevertheless, fundamentally different from AWGN, real camera noise is generally more sophisticated and signal-dependent[18, 19]. The noise produced by photon sensing is further affected by the in-camera signal processing (ISP) pipeline (*e.g.*, Gama correction, compression, and demosaicing). Models trained with setting1 are easily over-fitted to AWGN and fail in real noise removal. Setting2 is based on ISP-modelling CNN [20] and Poisson-Gaussian [19, 21] noise model that modelling photon sensing

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

with Poisson and remaining stationary disturbances with Gaussian has been adopted in RAW imaging. As shown in Fig 1 (a1), setting2 adds a Poisson-Gaussian noise with the clean RAW image and then passes the result through a pre-trained RAW2RGB CNN to obtain the RGB noisy counterpart. Notably, when the clean RAW image is unavailable, a pre-trained RGB2RAW CNN is utilized to transform the clean RGB image to its RAW counterpart [20]. However, setting2 has the following drawbacks : (i) The noise models are all hand-crafted. Because of the randomness and complexity of real camera noise, it's difficult to customize a hand-crafted method to model all the characteristics of real noise. (ii) These approaches lack the supervision over corrupted images to ensure the quality of generated noisy images. (iii) The ISP pipeline is very sophisticated and hard to completely modelled, the RAW2RGB CNN branch learns only the mapping from clean RAW domain to clean RGB space. However, the mapping from Poisson-Gaussian noisy RAW image to real noisy RGB image can not be ensured. (iv) The ISP pipelines of different devices vary significantly, which results in the poor generality and robustness of ISP modelling CNNs. Thus, whether noisy images are synthesized with setting1 or 2, there still remains a discrepancy between synthetic and real noisy datasets. We notice that GAN utilizes the internal information of input image and external information from other images when modelling the image priors, showing great power in image generation. Hence, we suggest using GAN to adaptively narrow the domain discrepancy between synthetic and real noisy datasets.

GAN is firstly introduced in [22] and have been proven successful in image synthesis [23] and translation [24]. Subsequently, GAN is applied to image restoration and enhancement, *e.g.* super resolution [25, 26], style transfer [24, 27], enlighten [28, 29], deraining [30], dehazing [31], image inpainting [32, 33], image editing [34, 35], and mobile photo enhancement [36, 37, 38]. Although GAN is widely applied in low-level vision tasks, few works are dedicated to investigating the realistic noise generation problem. Chen et al. [40] propose a simple GAN that takes Gaussian noise as input to generate noisy patches, but yield poor results. However, as in general, this GAN is image-level, *i.e.*, it treats images as samples and attempts to approximate the probability distribution of noisy images. This image-level GAN does not take into account that each pixel of a real noisy image is a random variable and the real noise is spatially-correlated, thus results in rough distribution approximation.

To tackle the above problems, we propose a framework, namely Pixel-level Noise Generative Adversarial Network (PNGAN) learning to generate realistic noisy images from the synthetic counterparts. Different from prior works, We are not interested in modelling the ISP pipeline. PNGAN treats each pixel of real noisy images as a random variable and is dedicated to approximating its distribution. In particular, PNGAN consists of three components, *i.e.*, a noise Generator, a denoise Discriminator and a dense two-category classification Discriminator. To begin with, customize a lightweight yet efficient CNN architecture, Simple Multi-scale Network (SMNet) as the noise Generator. SMNet learns rich multi-scale auto-correlation for better real noise fitting. The synthetic noisy image undergoes the Generator to produce the generated noisy image. Secondly, PNGAN conducts a soft pixel-wise alignment between the generated and real noisy images. Notably, in the imaging system, noise come from multiple sources (*e.g.*, short noise, thermal noise, dark current noise.) and becomes spatio-chromatically correlated after the transformation of the ISP. All these make the real noise more sophisticated and irregular. Therefore, directly performing pixel-wise alignment between the generated and real noisy images is harmful to the quality of generated noisy image. Hense, we design a denoise Discriminator that is also a real denoiser. The generated and real noisy images undergo the denoise Discriminator, then the distance of denoised results is minimized to realize a soft pixel-wise alignment. Thirdly, PNGAN approximates the probability distribution of each pixel of real noisy images and encourages the noise Generator to model this distribution. To this end, we further propose the pixel-level two-category classification Discriminator aiming to distinguish fake noise from real noise. Different from general GAN, our classification Discriminator outputs a score map. Each location on the score map indicates how realistic the corresponding pixel of the input noisy image is. As analyzed in the previous paragraph, our classification Discriminator is superior to general GAN because it better meets the real noise situation. Finally, the denoise and classification Discriminators establish an effective pixel-level supervision over the noise Generator and encourage it to model the distribution of real noise. In conclusion, our contributions can be summarized into four points:

(1) We propose a reasonable pixel-level RGB image noise model. This model treats each pixel of a real-world noisy image as a random variable and suggests approximating its probability distribution.

(2) We propose a novel framework, PNGAN for realistic noise generation. PNGAN establishes a pixel-level adversarial training between a Generator and two Discriminators to encourage the Generator to model the probability distribution of real noise, which can be proved theoretically.

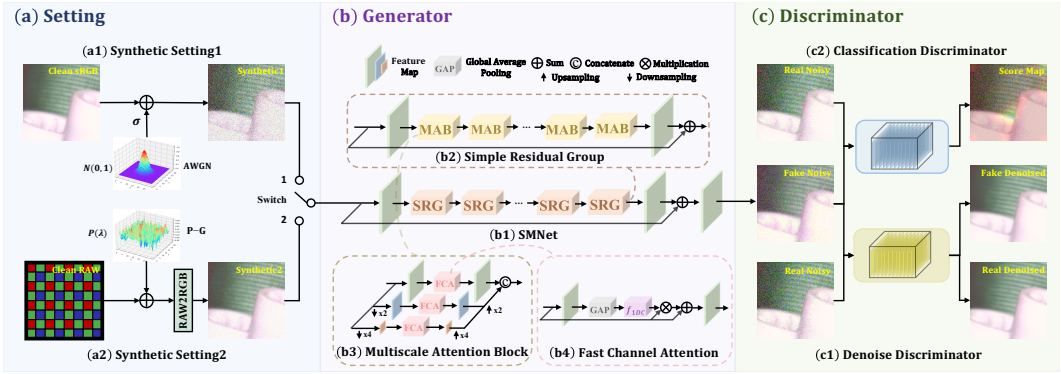


Figure 1: Overview of PNGAN. (a1) and (a2) are two synthetic settings. (b1) is the architecture of SMNet. (b2) depicts the architecture of SRG, which is the basic unit of SMNet. (b3) illustrates the details of MAB, which is the building block of SRG. MAB is equipped with FCA, which is shown in (b4). (c1) is the denoise Discriminator. (c2) depicts a dense two-category classification Discriminator.

(3) We propose an efficient CNN architecture, SMNet learning rich multi-scale auto-correlation for noise fitting. SMNet serves as the Generator in PNGAN costing only 0.8M Params and 20G FLOPS.

(4) Comprehensive experiments demonstrate that PNGAN can effectively narrow the domain discrepancy between synthetic and real noisy datasets. Besides, a series of CNN denoisers are promoted after being finetuned with the generated images and yield SOTA results on four public benchmarks.

2 Proposed Method

The PNGAN pipeline is depicted in Fig 1 , consisting of three parts, *i.e.*, (a) Setting, (b) Generator, and (c) Discriminator. (a1), (a2) are two common settings for synthesizing noisy images. The two kinds of synthetic images are used as the input (selected by the switch) of (b1) the noise Generator G for narrowing the domain discrepancy between synthetic and real noise. Then the generated noisy images are fed into (c1) denoise Discriminator D_d and (c2) classification Discriminator D_c respectively, together with the real noisy counterparts. This framework can be end-to-end trainable.

2.1 Pixel-level Adversarial Training

Noise Modeling. When taking multiple noisy images of the same scene, it can be observed that the noise intensity of the same pixel is very different. Hence, PNGAN treats each pixel of noisy images as a random variable. Specifically, A real noisy image, $\mathbf{I}_{rn} \in \mathbb{R}^{H\times W\times 3}$ at pixel i can be written as:

$$\mathbf{I}_{rn}[i] = \hat{\mathbf{I}}_{clean}[i] + \mathbf{N}[i], \quad D_d(\mathbf{I}_{rn}[i]) = \hat{\mathbf{I}}_{clean}[i], \quad 1 \le i \le H \times W, \tag{1}$$

where $\hat{\mathbf{I}}_{clean} \in \mathbb{R}^{H\times W\times 3}$ is the pseudo clean counterpart of $\mathbf{I}_{rn}$, it's denoised by D_d. Each $\mathbf{N}[i]$ is a random noise variable with unknown distribution. Therefore, each $\mathbf{I}_{rn}[i]$ can also be viewed as a distribution-unknown (denoted as $P_{data}(x_i)$) random variable. Now we aim to design a framework to generate a fake noisy image $\mathbf{I}_{fn} \in \mathbb{R}^{H\times W\times 3}$ such that the distribution of $\mathbf{I}_{fn}[i]$ and $\mathbf{I}_{rn}[i]$ is as close as possible. To this end, we split the problem into two steps: (i) Constant alignment of $\hat{\mathbf{I}}_{clean}[i]$. (ii) Distribution approximation of $\mathbf{N}[i]$. Please note that the mapping learned by D_d is strictly from $\mathbf{I}_{rn}$ to $\hat{\mathbf{I}}_{clean}$. If the constant in Eq 1 is set as the clean image, $\mathbf{I}_{clean} \in \mathbb{R}^{H\times W\times 3}$, the subsequent constant alignment and probability distribution approximation will introduce unnecessary errors.

Constant Alignment. Due to the intrinsic randomness, complexity, and irregularity of real noise, directly deploying $\mathcal{L}_1$ loss between $\mathbf{I}_{fn}$ and $\mathbf{I}_{rn}$ will drastically damage the quality of $\mathbf{I}_{fn}$ and fail in handling pixel-level alignment. Therefore, we feed $\mathbf{I}_{fn}$ and $\mathbf{I}_{rn}$ into D_d to obtain their denoised versions and then perform $\mathcal{L}_1$ loss to realize soft pixel-level alignment, which can be formulated as:

$$\mathcal{L}_1 = \sum_{i=1}^{H\times W} \left|\left|D_d(\mathbf{I}_{fn})[i] - D_d(\mathbf{I}_{rn})[i]\right|\right|_1 = \sum_{i=1}^{H\times W} \left|\left|D_d(\mathbf{I}_{fn})[i] - \hat{\mathbf{I}}_{clean}[i]\right|\right|_1. \tag{2}$$

By employing D_d and minimizing $\mathcal{L}_1$, the procedure of constant alignment can be free from the interference of noise and enjoys a better convergence. We prove this in the experiment part.

Distribution Approximation. Note that after finishing the constant alignment of $\hat{\mathbf{I}}_{clean}[i]$, the distribution approximation of $\mathbf{N}[i]$ is equivalent to the distribution approximation of $\mathbf{I}_{rn}[i]$. In addition, the later is more beneficial to capture the correlation between noise and scene. We denote some real noisy pixel samples of $\mathbf{I}_{rn}[i]$ as $\{x_i^1, x_i^2, ..., x_i^m\}$ such that $x_i^k \sim P_{data}(x_i)$ and the distribution of $\mathbf{I}_{fn}[i]$ as $P_G(x_i;\theta_G)$, here θ_G is the parameter of G. We hope that $P_G(x_i;\theta_G)$ is as close to $P_{data}(x_i)$ as possible for each pixel i. We adopt Maximum Likelihood Estimate (MLE) method to handle this problem. The likelihood function controlled by θ_G is constructed as:

$$L = \prod_{i=1}^{H\times W} \prod_{k=1}^{m} P_G(x_i^k;\theta_G). \tag{3}$$

Subsequently, the original problem is transformed as finding a θ_G^* to maximize L, formulated as:

$$\begin{aligned}
\theta_G^* &= \underset{\theta_G}{arg\,max} \prod_{i=1}^{H\times W} \prod_{k=1}^{m} P_G(x_i^k;\theta_G) \\
&= \underset{\theta_G}{arg\,max}\ \texttt{log}(\prod_{i=1}^{H\times W} \prod_{k=1}^{m} P_G(x_i^k;\theta_G)) \\
&= \underset{\theta_G}{arg\,max} \sum_{i=1}^{H\times W} \sum_{k=1}^{m} \texttt{log} P_G(x_i^k;\theta_G) \\
&= \underset{\theta_G}{arg\,max}\ \mathbb{E}_i \left[\ \mathbb{E}_{x_i^k} \left[\ \texttt{log} P_G(x_i^k;\theta_G)\ \right]\right],
\end{aligned} \tag{4}$$

Where $\mathbb{E}$ means taking the average value. To obtain a better θ_G^*, we suggest using D_d and D_c. To begin with, we note that D_d learns the mapping from $P_{data}(x_i)$ to $\hat{\mathbf{I}}_{clean}[i]$ for each pixel i and it is fixed while training G. According to backpropagation, the optimization of G can be formulated as:

$$\begin{aligned}
\theta_G^{*'} &= \underset{\theta_G}{arg\,min} \sum_{i=1}^{H\times W} \left|\left|D_d(\mathbf{I}_{fn})[i] - D_d(\mathbf{I}_{rn})[i]\right|\right|_1 \\
&\approx \underset{\theta_G}{arg\,min} \sum_{i=1}^{H\times W} discrepancy(P_{data}(x_i), P_G(x_i;\theta_G)) \\
&= \underset{\theta_G}{arg\,min} \sum_{i=1}^{H\times W} KL(P_{data}(x_i)||P_G(x_i;\theta_G)) \\
&= \underset{\theta_G}{arg\,min} \sum_{i=1}^{H\times W} \int_{x_i} P_{data}(x_i) \texttt{log} \frac{P_{data}(x_i)}{P_G(x_i;\theta_G)} dx_i \\
&= \underset{\theta_G}{arg\,min} \sum_{i=1}^{H\times W} \int_{x_i} P_{data}(x_i)(\texttt{log} P_{data}(x_i) - \texttt{log} P_G(x_i;\theta_G)) dx_i \\
&= \underset{\theta_G}{arg\,min} \sum_{i=1}^{H\times W} \int_{x_i} -P_{data}(x_i) \texttt{log} P_G(x_i;\theta_G) dx_i \\
&= \underset{\theta_G}{arg\,max} \sum_{i=1}^{H\times W} \int_{x_i} P_{data}(x_i) \texttt{log} P_G(x_i;\theta_G) dx_i \\
&= \underset{\theta_G}{arg\,max} \sum_{i=1}^{H\times W} \mathbb{E}_{x_i \sim P_{data}(x_i)}[\texttt{log} P_G(x_i;\theta_G)] \\
&\approx \underset{\theta_G}{arg\,max}\ \mathbb{E}_i \left[\ \mathbb{E}_{x_i^k} \left[\ \texttt{log} P_G(x_i^k;\theta_G)\ \right]\right] \\
&= \theta_G^*,
\end{aligned} \tag{5}$$

Where KL denotes Kullback-Leibler divergence. Eq 5 indicates that $\theta_G^{*'}$ is an approximation of θ_G^*, which means minimizing $\mathcal{L}_1$ can narrow the discrepancy between $P_{data}(x_i)$ and $P_G(x_i;\theta_G)$ for

each pixel i. Secondly, to further optimize θ_G to maximize L, we employ D_c. D_c is a pixel-level two-category classification Discriminator. D_c outputs a score map $\mathbf{P} \in \mathbb{R}^{H\times W}$. Specifically, the information of $\mathbf{P}[i] \in [0,1]$ is the probability value indicating how realistic $P_G(x_i;\theta_G)$ is. Please refer to the supplementary material for the detailed architecture of D_c. As analyzed in [22], G aims to generate more realistic $\mathbf{I}_{fn}$ to fool D_c while D_c targets at distinguishing between $\mathbf{I}_{fn}$ and $\mathbf{I}_{rn}$. According to Eq 3 , we formulate the adversarial training between G and D_c as a min-max problem:

$$\underset{\theta_G}{min}\ \underset{\theta_{D_c}}{max}\ \mathbb{E}_i\big[\mathbb{E}_{\mathbf{I}_{rn}}\ [\texttt{log}(D_c(\mathbf{I}_{rn};\theta_{D_c})[i])]\big] + \mathbb{E}_i\big[\mathbb{E}_{\mathbf{I}_{fn}}\ [\texttt{log}(1-D_c(\mathbf{I}_{fn};\theta_{D_c})[i])]\big], \tag{6}$$

where $\mathbb{E}_{\mathbf{I}_{fn}}$ and $\mathbb{E}_{\mathbf{I}_{rn}}$ respectively represent the operation of taking the average for all fake and real data in the mini-batch. As analyzed in [41], to make GANs analogous to divergence minimization and produce sensible predictions based on the *a priori* knowledge that half of the samples in the mini-batch are fake, we utilize the recently proposed relativistic discriminator [41] as follow:

$$\begin{aligned} D_c(\mathbf{I}_{rn};\theta_{D_c}) &= \sigma(C_d(I_{rn})), \quad D_{Ra}(\mathbf{I}_{rn},\mathbf{I}_{fn}) = \sigma(C_d(I_{rn}) - \mathbb{E}_{\mathbf{I}_{fn}}(C_d(\mathbf{I}_{fn}))), \\ D_c(\mathbf{I}_{fn};\theta_{D_c}) &= \sigma(C_d(I_{fn})), \quad D_{Ra}(\mathbf{I}_{fn},\mathbf{I}_{rn}) = \sigma(C_d(\mathbf{I}_{fn}) - \mathbb{E}_{\mathbf{I}_{rn}}(C_d(\mathbf{I}_{rn}))), \end{aligned} \tag{7}$$

where D_{Ra} denotes the relativistic discriminator which estimates the probability that real data is more realistic than fake data and also directs the generator to synthesize a fake image that is more realistic than real images. Here σ means the *Sigmoid* activation. C_d represents the non-transformed Discriminator output. The loss functions of D_c and G are then defined in a symmetrical form:

$$\begin{aligned} \mathcal{L}_{D_c} &= -\mathbb{E}_i\big[\mathbb{E}_{\mathbf{I}_{rn}}[\texttt{log}(D_{Ra}(\mathbf{I}_{rn},\mathbf{I}_{fn})[i]) - \mathbb{E}_{\mathbf{I}_{fn}}[1-\texttt{log}(D_{Ra}(\mathbf{I}_{fn},\mathbf{I}_{rn})[i])]\big], \\ \mathcal{L}_G &= -\mathbb{E}_i\big[\mathbb{E}_{\mathbf{I}_{rn}}[1-\texttt{log}(D_{Ra}(\mathbf{I}_{rn},\mathbf{I}_{fn})[i])] - \mathbb{E}_{\mathbf{I}_{fn}}[\texttt{log}(D_{Ra}(\mathbf{I}_{fn},\mathbf{I}_{rn})[i])]\big]. \end{aligned} \tag{8}$$

According to the analysis in [41] and the definition of $\mathbf{P}$. When given a $\theta^*_{D_c}$, we can obtain:

$$\begin{aligned} \theta_G^{*''} &= \underset{\theta_G}{arg\,min} - \mathbb{E}_i\big[\mathbb{E}_{\mathbf{I}_{rn}}[1-\texttt{log}(D_{Ra}(\mathbf{I}_{rn},\mathbf{I}_{fn})[i])] - \mathbb{E}_{\mathbf{I}_{fn}}[\texttt{log}(D_{Ra}(\mathbf{I}_{fn},\mathbf{I}_{rn})[i])]\big]\big|_{\theta^*_{D_c}} \\ &= \underset{\theta_G}{arg\,max}\ \mathbb{E}_i\big[\mathbb{E}_{\mathbf{I}_{rn}}[1-\texttt{log}(D_{Ra}(\mathbf{I}_{rn},\mathbf{I}_{fn})[i])] + \mathbb{E}_{\mathbf{I}_{fn}}[\texttt{log}(D_{Ra}(\mathbf{I}_{fn},\mathbf{I}_{rn})[i])]\big]\big|_{\theta^*_{D_c}} \\ &= \underset{\theta_G}{arg\,max}\ \mathbb{E}_i\big[\mathbb{E}_{\mathbf{I}_{rn}}[1-\texttt{log}(D_c(\mathbf{I}_{rn};\theta^*_{D_c})[i])] + \mathbb{E}_{\mathbf{I}_{fn}}[\texttt{log}(D_c(\mathbf{I}_{fn};\theta^*_{D_c})[i])]\big]\big|_{\theta^*_{D_c}} \\ &= \underset{\theta_G}{arg\,max}\ \mathbb{E}_i\big[\mathbb{E}_{\mathbf{I}_{fn}}[\texttt{log}(D_c(\mathbf{I}_{fn};\theta^*_{D_c})[i])]\big]\big|_{\theta^*_{D_c}} \\ &\approx \underset{\theta_G}{arg\,max}\ \mathbb{E}_i\ \big[\ \mathbb{E}_{x_i^k}\ [\ \texttt{log}P_G(x_i^k;\theta_G)\ ]\ \big]\big|_{\theta^*_{D_c}} \\ &= \theta^*_G\big|_{\theta^*_{D_c}}. \end{aligned} \tag{9}$$

This means $\theta_G^{*''}$ is an approximation of θ^*_G under the condition that θ_{D_c}= $\theta^*_{D_c}$. Notely, $D_c(\mathbf{I}_{fn};\theta_{D_c})[i]$ encodes how realistic $P_G(x_i^k;\theta_G)$ is. With the adversarial training, the discriminant ability of D_c is improved, thus $D_c(\mathbf{I}_{fn};\theta_{D_c})[i]$ becomes more accurate, and thus the error in Eq 9 decreases. So far, we have proved the adversarial training between G and D_d, G and D_c can narrow the discrepancy between $P_{data}(x_i)$ and $P_G(x_i;\theta_G)$ and facilitate G to generate more realistic noisy images.

2.2 Noise Generator

In sec 2.1 , we denote the probability distribution of $\mathbf{I}_{fn}[i]$ as $P_G(x_i;\theta_G)$. Now we customize a light-weight yet efficient CNN architecture, SMNet as G to generate $P_G(x_i;\theta_G)$. In this section, we firstly introduce the input setting of G and subsequently the detailed architecture of SMNet.

Input Setting. To derive a better constant alignment of $\hat{\mathbf{I}}_{clean}$, the input of G should be in a neighborhood of $\hat{\mathbf{I}}_{clean}$. Hence, we adopt a synthetic noisy image $\mathbf{I}_{syn} \in \mathbb{R}^{H\times W\times 3}$ as input. As depicted in Fig 1 (a), $\mathbf{I}_{syn}$ can be synthesized with the two common synthetic settings. Which synthetic setting is selected is controlled by the switch. As analyzed in Sec 1 , there still remains a significant domain discrepancy between the synthetic and real noisy RGB images. Therefore, to narrow this domain discrepancy and generate more realistic noisy images, we propose SMNet.

SMNet. The architecture of SMNet is illustrated in Fig 1 (b1). SMNet involves t Simple Residual Groups (SRG) and each SRG contains n Multi-scale Attention Blocks (MAB). The synthetic input continuously undergoes a convolution (*conv*) layer f_1, t SRGs, and a *conv* layer f_2, then added

with a long identity mapping for efficient residual learning to pass through the final *conv* layer f_3 to eventually generate the fake noisy counterpart, $\mathbf{I}_{fn} \in \mathbb{R}^{H\times W\times 3}$. This process can be formulated as:

$$\mathbf{I}_{fn} = f_3(\mathbf{I}_{syn} + f_2(S_t(\mathbf{F}_{S_t}))),\ \ \mathbf{F}_{S_{j+1}} = S_j(\mathbf{F}_{S_j}),\ \ \mathbf{F}_{S_1} = f_1(\mathbf{I}_{fn}), \tag{10}$$

where S_j denotes the j_{th} SRG, $1 \leq j \leq t-1$. The components of SRG are depicted in Fig 1 (b2). We define the input feature of the j_{th} SRG as $\mathbf{F}_{S_j} \in \mathbb{R}^{H\times W\times C}$ and its channel as C. $\mathbf{F}_{S_j}$ continuously undergoes a *conv* layer, n MABs, and a *conv* layer to add with an identity mapping:

$$\mathbf{F}_{S_{j+1}} = \mathbf{F}_{S_j} + M_n^j(\mathbf{F}_{M_n^j}),\ \ \mathbf{F}_{M_{k+1}^j} = M_k^j(\mathbf{F}_{M_k^j}),\ \ \mathbf{F}_{M_1^j} = \mathbf{F}_{S_j}, \tag{11}$$

where M_k^j denotes the k_{th} MAB of the j_{th} SRG, $1 \leq k \leq n-1$. MAB is the basic building block and the most significant component of SMNet. The details of MAB are depicted in Fig 1 (b3). We customize MAB with the following motivations: (i) Multi-scale feature fusion can increase the receptive field and multi-resolution contextual information can cover rich auto-correlation, which provides a more accurate spatial representation for noise fitting. (ii) The noise level decreases as the scale increases, and nonlinear sampling operations can increase the richness of the mapping in the potential space of real noise. Therefore, we exploit parallel multi-resolution branch aggregation from top to bottom and bottom to top to facilitate the learning of complex real noise. (iii) Specifically, during the feature downsampling, general downsample operation damages the image information, resulting in pixel discontinuity and jagged artifact. To alleviate these issues, we exploit Shift-Invariant Downsample [42] that copes with the discontinuity by using continuous pooling and filtering operation, preserving rich cross-correlation information between original and downsampled images. (iv) To efficiently capture continuous channel correlation and avoid information loss, we use 1D channel attention module, Fast Channel Attention (FCA) instead of the general 2D convolution attention module. The input feature, $\mathbf{F}_{M_k^j} \in \mathbb{R}^{H\times W\times C}$ is feed into three parallel multi-scale paths:

$$\mathbf{F}^1_{M_k^j} = FCA(\mathbf{F}_{M_k^j}),\ \ \ \mathbf{F}^2_{M_k^j} = f^2_{up}(FCA(f^2_{sid}(\mathbf{F}_{M_k^j}))),\ \ \ \mathbf{F}^4_{M_k^j} = f^4_{up}(FCA(f^4_{sid}(\mathbf{F}_{M_k^j}))), \tag{12}$$

where FCA denotes Fast Channel Attention. f^2_{up} denotes a *conv* layer after bilinear interpolation up-sampling, 2 is the scale factor. f^4_{up} is similarly defined. f^2_{sid} means Shift-Invariant Downsample [42], 2 is also the scale factor. f^4_{sid} is similarly defined. Subsequently, the output feature is derived by:

$$M_k^j(\mathbf{F}_{M_k^j}) = \mathbf{F}_{M_k^j} + f([\mathbf{F}^1_{M_k^j}, \mathbf{F}^2_{M_k^j}, \mathbf{F}^4_{M_k^j}]), \tag{13}$$

where f represents the last *conv* layer, $[\cdot,\cdot,\cdot]$ denotes concatenating operation. The architecture of FCA is shown in Fig 1 (b4). We define the input feature as $\mathbf{F}_d$, then FCA can be formulated as:

$$FCA(\mathbf{F}_d) = \mathbf{F}_d \cdot \big(1 + \sigma\big(f_{1DC}(GAP(\mathbf{F}_d))\big)\big), \tag{14}$$

here σ represents the *Sigmoid* activation function, GAP means global average pooling along the spatial wise, f_{1DC} denotes 1-Dimension Convolution. In this work, we set t = 3, n = 2, and C = 64.

Training Objective. In addition to the aforementioned losses, we employ a perceptual loss to yield a more visual-pleasing image and improve the structural contents and detailed textures of $\mathbf{I}_{fn}$:

$$\mathcal{L}_p = \big|\big|VGG(\mathbf{I}_{fd}) - VGG(\mathbf{I}_{rd})\big|\big|_2^2,\ \ \mathbf{I}_{fd} = D_d(\mathbf{I}_{fn}),\ \ \mathbf{I}_{rd} = D_d(\mathbf{I}_{rn}), \tag{15}$$

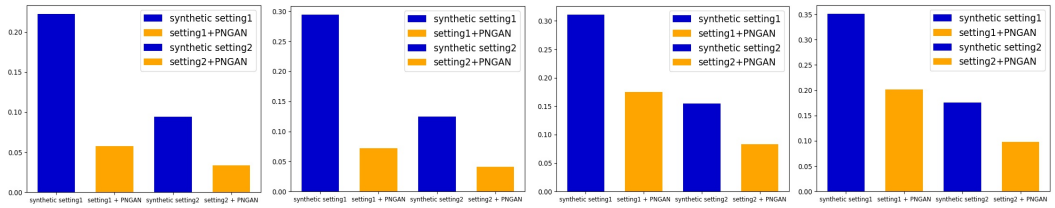
Where VGG denotes the last feature map of vgg16. Eventually, the overall training objective is:

$$\mathcal{L} = \mathcal{L}_1 + \lambda_p \cdot \mathcal{L}_p + \lambda_{Ra} \cdot (\mathcal{L}_{D_c} + \mathcal{L}_G), \tag{16}$$

Where λ_p and λ_{Ra} are two hyper-parameters controlling the importance balance. The proposed PNGAN framework is end-to-end trained by minimizing $\mathcal{L}$. Note that the parameters in D_d and vgg16 are fixed. Each mini-batch training procedure is divided into two steps: (i) Fix D_c and train G. (ii) Fix G and train D_c. This pixel-level adversarial training scheme promotes D_c the ability to distinguish fake noisy images from real noisy images and allows G to learn to create the solutions that are highly similar to real camera noisy images and thus difficult to be classified by D_c.

3 Experiment

Datasets. We first use SIDD `train` set to train D_d. Then we fix D_d to train G on the same set. Subsequently, G uses DIV2K [51], Flickr2K [52], BSD68 [53], Kodak24 [54], and Urban100 [55] to



SIDD DND PolyU Nam

Figure 2: Domain discrepancy. We use the metric, Maximum Mean Discrepancy (MMD) to measure the domain discrepancy between synthetic and real noisy datasets, PNGAN generating and real noisy datasets. Under both setting1 and 2, the discrepancy decreases significantly when PNGAN is applied.

benchmark	DnCNN-B [10]	CBDNet [43]	RIDNet [44]	AINDNet [45]	VDN [46]	CycleISP [20]	MPRNet [47]	MIRNet [48]	**RIDNet***	**MPRNet***	**MIRNet***
SIDD	23.66	33.28	38.71	39.15	39.23	39.52	39.71	39.72	**39.25**	**40.06**	**40.07**
	0.583	0.868	0.914	0.955	0.955	0.957	0.958	0.959	**0.956**	**0.960**	**0.960**
DND	32.43	38.06	39.26	39.53	39.38	39.56	39.80	39.88	**39.55**	**40.18**	**40.25**
	0.790	0.942	0.953	0.956	0.952	0.956	0.954	0.956	**0.955**	**0.961**	**0.962**
benchmark	RDN [12]	FFDNet+ [17]	TWSC [49]	CBDNet [43]	RIDNet [44]	VDN [46]	MPRNet [47]	MIRNet [48]	**RIDNet***	**MPRNet***	**MIRNet***
PolyU	37.94	38.17	38.68	38.74	38.86	39.04	39.07	39.18	**39.54**	**40.48**	**40.55**
	0.946	0.951	0.958	0.961	0.962	0.965	0.969	0.973	**0.971**	**0.982**	**0.983**
Nam	38.16	38.81	38.96	39.08	39.20	39.68	39.41	39.57	**39.69**	**40.72**	**40.78**
	0.956	0.957	0.962	0.969	0.973	0.976	0.974	0.979	**0.975**	**0.984**	**0.986**

Table 1: Comparison on four benchmarks. * denotes finetuned with images generated by PNGAN.

generate realistic RGB noisy images. Finally, we finetune our models with the large-scale generated noisy-clean image pairs and evaluate them on four benchmarks: SIDD, DND, PolyU, and Nam. The images in SIDD [56] benchmark are collected using five smartphone cameras in 10 static scenes. They have various camera settings and lighting conditions. There are 320 image pairs for training and 1280 image patch pairs for validation. The DND [57] dataset composes 50 noisy-clean image pairs captured by 4 consumer cameras. 1000 patches at size 512×512 are cropped from the collected images. PolyU [58] consists of 40 real camera noisy images of different scenes. Nam [59] is composed of real noisy images of 11 static scenes. We respectively collect 20 clean images from DIV2K, Flickr2K, BSD68, Kodak24, and Urban100 to generate extended noisy-clean image pairs.

Implementation Details. We set the hyper-parameter λ_p = 6e-3, λ_{Ra} = 8e-4. For synthetic setting1, we set the noise intensity, σ_n = 50. For synthetic setting2, we directly exploit CycleISP to generate the synthetic noisy input. We adopt a pre-trained RIDNet as D_d. It can also be replaced by other alternatives. All the sub-modules (noise Generator, denoise Discriminator, and classification Discriminator) are trained with the Adam optimizer ($\beta_1 = 0.9$ and $\beta_1 = 0.9999$) for 7×10^5 iterations. The initial learning rate is set as 2×10^{-4}. The cosine annealing strategy [60] is employed to steadily decrease the learning rate from the initial value to 10^{-6} during the training procedure. Patches at size 128×128 cropped from training images are fed into the models. The batch size is set as 8. The horizontal and vertical flips are performed for data augmentation. All the models are trained on RTX8000 GPUs. While finetuning, we set the learning rate as 1e-6, other settings remain unchanged.

Domain Discrepancy Validation. We use the widely applied metric, Maximum Mean Discrepancy (MMD) [61] to measure the domain discrepancy between synthetic and real-world noisy images, PNGAN generating and real-world noisy images on four benchmarks. For DND, we derive a pseudo clean version by denoising the real noisy counterparts with a pre-trained MIRNet. Then we use the pseudo clean version to synthesize noisy images. The results are depicted as a histogram in Fig 2 . For setting1, the domain discrepancy decreases by 74 %, 75 %, 44 %, and 43 % on SIDD, DND, PolyU, and Nam when PNGAN is exploited. For setting2, the domain discrepancy decreases by 64 %, 67 %, 46 %, and 44 %. These results demonstrate that PNGAN can narrow the discrepancy between synthetic and real noisy datasets. Please refer to the supplementary for detailed calculation process.

Comparison with SOTA Methods. We use the generated noisy-clean image pairs to finetune a series of CNN denoisers. We compare our models and SOTA algorithms on four benchmarks: SIDD, DND, PolyU, Nam. The results are reported in Tab 1 . * denotes models finetuned with image pairs generated by PNGAN. We make the following observations: (i) Our methods outperform SOTA methods by a large margin. Specifically, MPRNet* and MIRNet* exceed the recent best method MIRNet by 0.34 and 0.35 dB on SIDD, 0.30 and 0.37 dB on DND. RIDNet*, MPRNet*, and MIRNet* surpass the best performers by 0.36, 1.30, and 1.37 dB on PoyU and 0.01, 1.04, and 1.10 dB on Nam. (ii) Compared with the counterparts that are not finetuned, our models achieve a significant promotion. In particular, RIDNet* is 0.54, 0.29, 0.68 and 0.49 dB higher than RIDNet on SIDD, DND, PolyU and Nam. MPRNet* achieves 0.35, 0.38, 1.41 and 1.31 dB gain than MPRNet on SIDD, DND, PolyU, and Nam. MIRNet* is improved by 0.35, 0.35, 1.37, and 1.21 dB. This evidence clearly suggests the high similarity between PNGAN generating and real noisy images. Models adapted with our fake noisy-clean image pairs generalize better across different benchmarks.

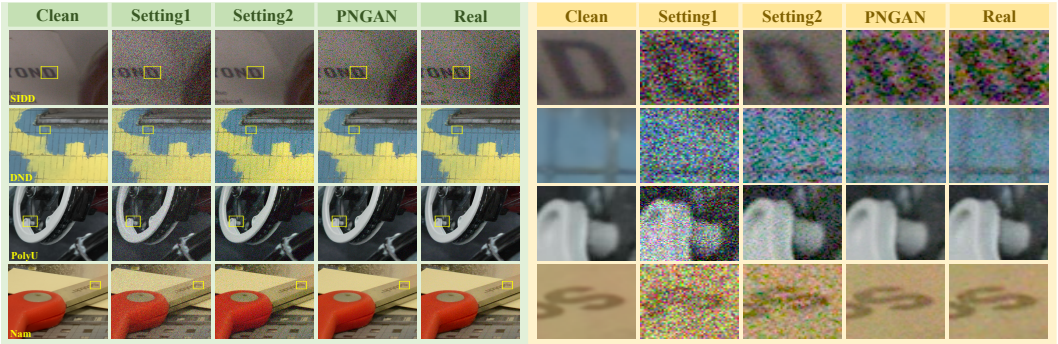


Figure 3: Visual comparisons of noisy images on SIDD, DND, PolyU, and Nam. Please zoom-in.

Method	SIDD					DF2K			
	setting1	setting1 + PNGAN	setting2	setting2 + PNGAN	real	setting1	setting1 + PNGAN	setting2	setting2 + PNGAN
EDSR [50]	22.18	**37.87 (+16.59)**	36.15	**38.43 (+2.28)**	38.45	22.74	**32.38 (+9.64)**	34.62	**38.23 (+3.61)**
RIDNet [44]	22.55	**37.92 (+15.37)**	36.13	**38.71 (+2.58)**	38.69	22.55	**32.10 (+9.55)**	33.98	**38.14 (+4.16)**
MPRNet [47]	22.86	**38.52 (+15.66)**	36.52	**39.53 (+3.01)**	39.45	22.85	**32.94 (+10.09)**	34.19	**38.61 (+4.42)**
MIRNet [48]	22.83	**38.76 (+15.93)**	36.55	**39.57 (+3.02)**	39.58	23.08	**32.34 (+9.26)**	34.26	**38.72 (+4.46)**

Table 2: More strong comparison by training models with different noisy-clean data from scratch.

Train from Scratch. For more strong comparison, we use the fake noisy images generated from clean SIDD `train` and DF2K (DIV2K+Flicker2K) to train CNN denoisers from scratch. The PSNR and SSIM results on SIDD `test` are listed in Tab 2 . For fair comparisons, all models are trained with the same experiment schedule except the training data. It can be observed: (i) On SIDD `train`, when PNGAN is applied to setting1, models are promoted by $\sim$ 16.79 dB and only $\sim$ 0.78 dB lower than their counterparts that trained by real data. While applying PNGAN to setting2 (CycleISP), models are improved by $\sim$ 2.72 dB. Surprisingly, in this case, models achieve almost the same performance as their counterparts trained by real data, the relative error is 0.083 $\%$. (ii) There is a significant domain discrepancy between DF2K and SIDD. To validate the generality of PNGAN, we also adopt synthetic DF2K noisy-clean image pairs to train models. As shown in the right part of Tab 2 , when PNGAN is applied to setting1, models are promoted by $\sim$ 9.64 dB. While applying PNGAN to setting2, models are improved by $\sim$ 4.16 dB and only $\sim$ 0.62 dB lower than their counterparts trained with SIDD real `train` set. These experiment results convincingly demonstrate: (i) The generated noise is very close to the real camera noise especially when PNGAN is applied to synthetic setting2. (ii) PNGAN can significantly narrow the domain discrepancy between synthetic and real-world noise.

Visual Examination. To more intuitively evaluate the quality of generated noisy images, we firstly provide visual comparisons of noisy images on four benchmarks: SIDD, DND, PolyU, and Nam, as shown in Fig 3 . Please zoom in for a better view. Note that the clean image of DND is pseudo, denoised from its noisy version by a MIRNet. The synthetic noisy images is synthesized with setting1, σ_n = 50. The left part depicts noisy images from SIDD, DND, PolyU , and Nam (top to down). The right part exhibits the small patches cropped by the yellow bounding boxes, from left to right: clean, synthetic setting1, setting2 (CycleISP), PNGAN generating, and real noisy images. As can be seen from the zoom-in patches: (i) The distribution and intensity of synthetic noise remain similar across diverse scenes, which differs from the real noise counterparts. (ii) The noise generated by CylceISP and PNGAN can change with the scene. Nonetheless, PNGAN can generate more realistic structural contents and the characteristics of the generated noise are more similar to those of the real noise counterpart. For instance, in line1 of Fig 3 , pictures of letter "D", the noise generated by PNGAN is highly similar to the real noise in intensity and distribution while the intensity of the noise generated by CycleISP is much lower and the edges of letter "D" is smoothed. These results suggest the superiority and generality of PNGAN in generating realistic noise. PNGAN performs robustly and reliably across different datasets photoed by different devices. Additionally, we compare the visual results of denoisers before and after being finetuned (denoted with *) with generated data in Fig 4 . It is observed that models finetuned with generated data are effective in real noise removal. Furthermore, they are capable of preserving the original RGB color, structural content, textural details, and the spatial smoothness of the homogeneous regions without introducing redundant artifacts. In contrast, original models either yield over-smooth images sacrificing fine textural details and structural content, produce RGB color distortion, or introduce redundant blotchy texture and chroma artifacts.

Break-down Ablations. We perform break-down ablations to evaluate the effects of PNGAN components and SMNet architecture. We select setting1 to synthesize the noisy input. The results on SIDD `test` are reported in Tab 3 . (i) Firstly, the Generator is set as SMNet to validate the effects

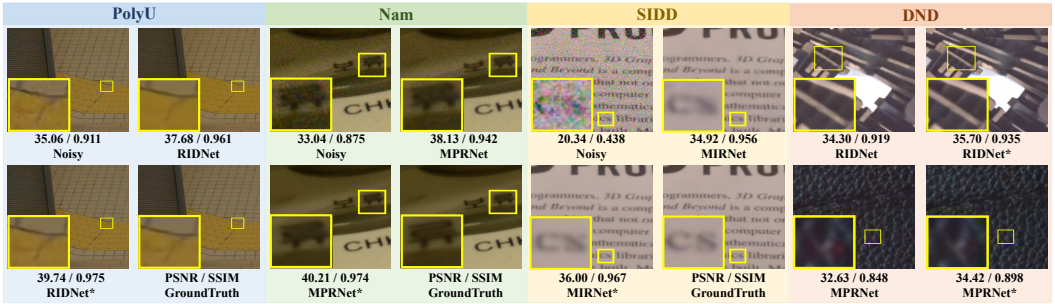


Figure 4: Visual results of denoisers before and after being finetuned with fake data. Please zoom-in.

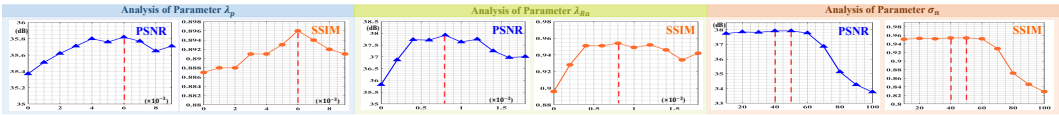


Figure 5: Parameter Analysis. The optimal setting is λ_p = 6e-3, λ_{Ra} = 8e-4, and σ_n = 40 or 50.

Method	PNGAN component					Generator Architecture			
	Baseline1	+ D_d	+ image-level D_c	+ pixel-level D_c	+ $\mathcal{L}_p$	Baseline2	+ Multi-scale Branch	+ SID	+ FCA
EDSR [50]	13.99 / 0.514	35.44 / 0.874	35.42 / 0.874	37.35 / 0.949	37.87 / 0.952	35.87 / 0.898	36.96 / 0.921	37.14 / 0.938	37.87 / 0.952
RIDNet [44]	14.54 / 0.500	35.37 / 0.887	35.39 / 0.888	37.49 / 0.950	37.92 / 0.954	35.62 / 0.888	37.01 / 0.943	37.23 / 0.947	37.92 / 0.954
MPRNet [47]	14.25 / 0.472	36.26 / 0.931	36.23 / 0.931	38.27 / 0.943	38.52 / 0.945	36.28 / 0.929	37.47 / 0.937	37.86 / 0.941	38.52 / 0.945
MIRNet [48]	13.57 / 0.468	36.15 / 0.921	36.17 /0.922	38.28 / 0.941	38.76 / 0.947	36.39 / 0.926	37.66 / 0.938	37.89 / 0.940	38.76 / 0.947

Table 3: Break-down ablation study of PNGAN component and the noise Generator Architecture.

of PNGAN components. We start from Baseline1, no Discriminator is used and $\mathcal{L}_1$ loss is directly performed between generated and real noisy images. Models trained with these fake images collapse dramatically. When D_d is applied, models trained with the generated images are promoted by 21.71 dB on average. In addition, the PSNR and SSIM between the denoised counterparts of generated and real noisy images are 39.14 dB and 0.928 on average respectively. This evidence demonstrates that D_d enables the soft pixel-level alignment between generated and real noisy images. Subsequently, we use an image-level D_c with stride *conv* layers to classify whether the whole generated image is real. Nonetheless, the performance of denoisers remains almost unchanged. After deploying D_c, the models are improved by 2.04 dB, suggesting that the pixel-level noise model is more in line with real noise scenes and benefits generating more realistic noisy images. When the perceptual loss is used, the denoisers gain a slight improvement by about 0.42 dB, indicating $\mathcal{L}_p$ facilitates yielding perceptually-pleasing results. (ii) Secondly, we only change the Generator architecture and fix other PNGAN components. We start from Baseline2 that doesn't exploit multi-scale feature fusion and FCA. When we add two different scale branches and use bilinear interpolation to downsample and upsample, models trained by the generated images are promoted by about 1.24 dB. After applying SID and FCA, the trained models further gain 0.26 and 0.74 dB improvement on average. These results convincingly demonstrate the superiority of the proposed SMNet in real-world noise fitting.

Parameter Analysis. We respectively validate the effects of parameters, λ_p , λ_{Ra} in Eq 16 , and the noise intensity of synthetic setting1, *i.e.*, σ_n. We change the parameters, train the Generator, and evaluate the performance of denoise models (RIDNet) trained with the fake noisy images on SIDD. When analyzing one parameter, the others are fixed at their optimal values. The results are shown in Fig 5 and the optimal setting is λ_p = 6e-3, λ_{Ra} = 8e-4, and σ_n = 40 or 50. It's noteworthy that when σ_n is too high, the performance of denoise models degrades drastically. However, when using different parameters, our model stably outperforms the baseline model trained with synthetic data (setting1). This evidence clearly suggests the robustness and reliability of the PNGAN framework.

4 Conclusion

Too much research focuses on designing a CNN architecture for real noise removal. By contrast, this paper is dedicated to investigating how to generate more realistic noisy images from synthetic counterparts so as to boom the denoisers' performance. We propose a reasonable framework, PNGAN that treats each noisy pixel as a random variable and attempts to approximate its probability distribution. We provide theoretical proof that the pixel-level adversarial training between the Generator and Discriminators can encourage the Generator to model the distribution of real noise. In addition, for better noise fitting, we customize an efficient multi-scale CNN, SMNet as the Generator. Comprehensive experiments demonstrate that PNGAN can effectively narrow the domain discrepancy between synthetic and real noisy datasets. Besides, denoisers finetuned with the generated noisy images outperform SOTA methods by a large margin on four datasets. Surprisingly, the SSIM of our method ranks the $\mathbf{1_{st}}$ place and the PSNR ranks the $\mathbf{2_{nd}}$ place on DND RGB denoising competition.

References

[1] C. M. Bishop, "Pattern recognition and machine learning," in *springer*, 2006.

[2] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *CVPR*, 2014.

[3] F. Zhu, G. Chen, and P.-A. Heng, "From noise modeling to blind image denoising," in *CVPR*, 2016.

[4] J. Xu, L. Zhang, and D. Zhang, "A trilateral weighted sparse coding scheme for real-world image denoising," in *ECCV*, 2018.

[5] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, 2006.

[6] A. Buades, B. Coll, and J. . Morel, "A non-local algorithm for image denoising," in *CVPR*, 2005.

[7] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "Nonlocal transform-domain filter for volumetric data denoising and reconstruction," *TIP*, 2013.

[8] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?," in *CVPR*, 2012.

[9] S. Lefkimmiatis, "Non-local color image denoising with convolutional neural networks," in *CVPR*, 2017.

[10] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *TIP*, 2017.

[11] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *CVPR*, 2017.

[12] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image restoration," *TPAMI*, 2020.

[13] Y. Chen, W. Yu, and T. Pock, "On learning optimized reaction diffusion processes for effective image restoration," in *CVPR*, 2015.

[14] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *TIP*, 2007.

[15] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *CVPR*, 2014.

[16] J. Xu, L. Zhang, D. Zhang, and X. Feng, "Multi-channel weighted nuclear norm minimization for real color image denoising," in *ICCV*, 2017.

[17] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *TIP*, 2018.

[18] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang., "Waterloo exploration database: New challenges for image quality assessment models," *TIP*, 2014.

[19] A. Foi, M. Trimeche, V. Katkovnik, and K. O. Egiazarian, "Practical poissonian-gaussian noise modeling and fitting for single-image raw-data," *TIP*, 2008.

[20] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "Cycleisp: Real image restoration via improved data synthesis," in *CVPR*, 2020.

[21] Y. Hwang, J.-S. Kim, and I.-S. Kweon, "Difference-based image noise modeling using skellam distribution," *TPAMI*, 2012.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[23] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "Autogan: Neural architecture search for generative adversarial networks," in *ICCV*, 2019.

[24] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networkss," in *ICCV*, 2017.

[25] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *CVPR*, 2017.

[26] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *ECCVW*, 2018.

[27] C. Li and M. Wand, "Combining markov random fields and convolutional neural networks for image synthesis," in *CVPR*, 2016.

[28] Y. Jiang, X. Gong, D. Liu, Y. Cheng, C. Fang, X. Shen, J. Yang, P. Zhou, and Z. Wang, "Enlightengan: Deep light enhancement without paired supervision," *TIP*, 2021.

[29] W. Yang, S. Wang, Y. Fang, Y. Wang, and J. Liu, "From fidelity to perceptual quality: A semi-supervised approach for low-light image enhancement," in *CVPR*, 2020.

[30] R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, "Attentive generative adversarial network for raindrop removal from a single image," in *CVPR*, 2018.

[31] R. Li, J. Pan, Z. Li, and J. Tang, "Single image dehazing via conditional generative adversarial network," in *CVPR*, 2018.

[32] C. Zheng, T.-J. Cham, and J. Cai, "Pluralistic image completion," in *CVPR*, 2019.

[33] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *CVPR*, 2018.

[34] S. Yang, Z. Wang, Z. Wang, N. Xu, J. Liu, and Z. Guo, "Controllable artistic text style transfer via shape-matching gan," in *ICCV*, 2019.

[35] S. Yang, Z. Wang, J. Liu, and Z. Guo, "Deep plastic surgery: Robust and controllable image editing with human-drawn sketches," in *ECCV*, 2020.

[36] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *CVPRW*, 2018.

[37] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *CVPRW*, 2018.

[38] Y.-S. Chen, Y.-C. Wang, M.-H. Kao, and Y.-Y. Chuang, "Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans," in *CVPR*, 2018.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[40] J. Chen, J. Chen, H. Chao, and M. Yang, "Image blind denoising with generative adversarial network based noise modeling," in *CVPR*, 2018.

[41] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," *arXiv preprint arXiv:1807.00734*, 2018.

[42] R. Zhang, "Making convolutional networks shift-invariant again," in *ICML*, 2019.

[43] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, "Toward convolutional blind denoising of real photographs," in *CVPR*, 2019.

[44] S. Anwar and N. Barnes, "Real image denoising with feature attention," in *ICCV*, 2019.

[45] Y. Kim, J. W. Soh, G. Y. Park, and N. I. Cho, "Transfer learning from synthetic to real-noise denoising with adaptive instance normalization," in *CVPR*, 2020.

[46] Z. Yue, H. Yong, Q. Zhao, D. Meng, and L. Zhang, "Variational denoising network: Toward blind noise modeling and removal," in *NIPS*, 2019.

[47] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "Multi-stage progressive image restoration," in *CVPR*, 2021.

[48] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "Learning enriched features for real image restoration and enhancement," in *ECCV*, 2020.

[49] J. Xu, L. Zhang, and D. Zhang, "A trilateral weighted sparse coding scheme for real-world image denoising," in *ECCV*, 2018.

[50] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *CVPRW*, 2017.

[51] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, and L. Zhang, "Ntire 2017 challenge on single image super-resolution: Methods and results," in *CVPRW*, 2017.

[52] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *CVPRW*, 2017.

[53] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *CVPR*, 2015.

[54] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and e. a. Michael Bernstein, "Imagenet large scale visual recognition challenge," in *IJCV*, 2015.

[55] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *CVPR*, 2015.

[56] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *CVPR*, 2018.

[57] T. Plotz and S. Roth, "Benchmarking denoising algorithms with real photographs," in *CVPR*, 2017.

[58] J. Xu, H. Li, Z. Liang, D. Zhang, and L. Zhang, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *arXiv:1804.02603*, 2018.

[59] S. Nam, Y. Hwang, Y. Matsushita, and S. J. Kim, "A holistic approach to cross-channel image noise modeling and its application to image denoising," in *CVPR*, 2016.

[60] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," in *ICLR*, 2017.

[61] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," in *JMLR*, 2012.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the supplementary
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the supplementary
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sec 2.1
   (b) Did you include complete proofs of all theoretical results? [Yes] See Sec 2.1
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplementary
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See **Implementation Details** in Sec 3
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See **Parameters Analysis** in Sec 3
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See **Implementation Details** in Sec 3
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The used data are all publicly avaliable.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The used data doesn't contain any personally identifiable information or offensive content.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]