Synthesizing Camera Noise using Generative Adversarial Networks

Bernardo Henz, Eduardo S. L. Gastal, and Manuel M. Oliveira

Abstract—We present a technique for synthesizing realistic noise for digital photographs. It can adjust the noise level of an input photograph, either increasing or decreasing it, to match a target ISO level. Our solution learns the mappings among different ISO levels from unpaired data using generative adversarial networks. We demonstrate its effectiveness both quantitatively, using Kullback-Leibler divergence and Kolmogorov-Smirnov test, and qualitatively through a large number of examples. We also demonstrate its practical applicability by using its results to significantly improve the performance of a state-of-the-art trainable denoising method. Our technique should benefit several computer-vision applications that seek robustness to noisy scenarios.

Index Terms—Noise model, GANs, Deep learning

1 Introduction

Noise is a fundamental problem in graphics, image processing, and computer vision, and many image-denoising techniques have been proposed in recent years [1], [2], [3], [4], [5]. While some of these approaches are highly successful in removing artificial additive white Gaussian noise (AWGN), recent works [6], [7], [8] have shown that the performance of such techniques is severely reduced when applied to real photographs. This is particularly true for the case of recent deep learning strategies [9], [10]. The main difficulty faced by these techniques results from the fact that noise found in digital photographs, which we refer to as *natural noise* (in opposition to *synthetic noise*), has multiple sources (*e.g.*, thermal, quantization, etc.), being much more complex than just white Gaussian noise.

While the importance of noise reduction is well understood, increasing noise level is also very useful. It can provide data for training techniques that need to handle noisy scenarios. These include, for instance, improving the performance of denoisers (as we will demonstrate), classifiers for low-light and challenging conditions [11], [12], [13]; performing superresolution in the presence of different noise levels [14]; performing noise reduction during demosaicing [15]; and detecting forgeries based on noise statistics [16]. These and other applications would benefit from the synthesis of realistic noise.

Although many works have studied the nature of noise theoretically [17], [18], [19], [20], no denoising technique seems to be able to directly use such information. We propose a *practical* data-driven solution for synthesizing noise that can, for instance, be used for training/fine-tuning denoising algorithms intended for real-world applications. Our technique can adjust the noise level of an input photograph to match a target ISO level. Fig. 1 illustrates the process: a photograph captured with ISO 100 (left) has its noise level adjusted to ISO 1600 (blue arrow). Likewise, another photograph of the same scene taken with ISO 1600 (right) has its noise level adjusted to ISO 100 (red arrow).

The noise distributions of the highlighted patches for the ISO 1600 photograph (Real 1600) and the one generated by our technique (Synthetic 1600) have a Kullback-Leibler (KL) divergence of 0.0617 and the result of their Kolmogorov-Smirnov (KS) test is 0.0878 (with a p-value of 7.81×10^{-220}), indicating that such distributions are very similar. Since we use the ISO 100 photograph as baseline for estimating the noise distributions, we cannot apply the same tests to ISO 100 patches. For this reason, Fig. 1 shows the PSNR value (29.13) computed for the image generated by our technique (Synthetic 100). Such value indicates good agreement with the patch of the actual ISO 100 photograph (Real 100).

Although it would be preferable to directly use noise variance instead of ISO levels for parameterizing a noise-adjustment process, existing variance-estimation techniques [21], [22], [23], [24] do not provide reliable estimates, as they consider additive white Gaussian noise (AWGN). ISO level, in turn, is a readily available and reliable information, which justifies our choice. As robust noise variance-estimation techniques become available, our approach can be adapted to use them.

To perform noise-level adjustment, we use a convolutional neural network (CNN). Unfortunately, the availability of datasets containing paired photographs captured under different ISO settings is limited, and creating a large one is a non-trivial task. Thus, we designed our technique to use unpaired datasets. We modify the cycle-consistency loss, and introduce a *low-frequency-consistency loss* term to preserve the contrast of the input image in the synthesized one. An ablation study shows how these changes and our modified generator architecture lead to high-frequency content that approximates natural noise (Section 5.1).

Fig. 2 compares the results produced by our technique with the ones generated with AWGN, Gaussian-Poissonian (AWGN+Poisson) and with Noise Flow [25] for various ISO levels and different smartphone camera models. For the three techniques, each noisy result was obtained by corrupting the corresponding *clean patch* provided as part

• *Bernardo Henz, Eduardo S. L. Gastal and Manuel M. Oliveira are with UFRGS. E-mails: bhenz, eslgastal, oliveira@inf.ufrgs.br.*

Manuscript received December 13, 2019.

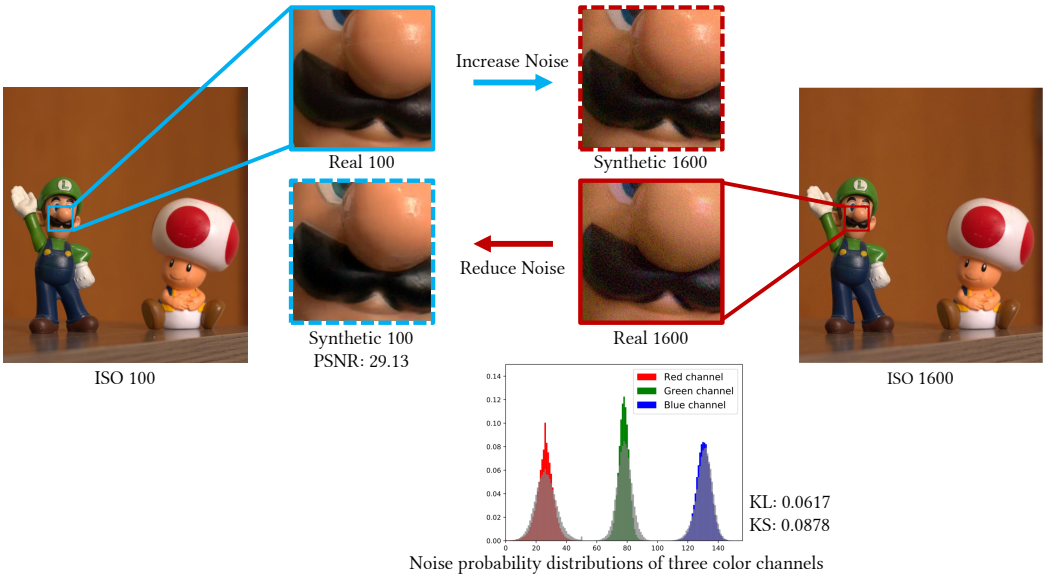

Fig. 1. Using our technique to adjust the noise level of photographs to different ISO values. A photograph captured with ISO 100 (left) has its noise level adjusted to ISO 1600 (blue arrow). Likewise, another photograph of the same scene taken with ISO 1600 (right) has its noise level adjusted to ISO 100 (red arrow). The noise distributions for patches Real 1600 and Synthetic 1600 have a Kullback-Leibler (KL) divergence of 0.0617 and the result of their Kolmogorov-Smirnov (KS) test is 0.0878 (with a p-value of 7.81×10^{-220}), indicating that the two distributions are very similar. The red, green, and blue histograms underneath patch Real 1600 show the noise distributions corresponding to the R, G, and B channels of patch Real 1600, respectively. The superimposed gray histograms are the corresponding noise distributions for the patch Synthetic 1600. The PSNR value computed for patch Synthetic 100 is 29.13, also indicating a good agreement with patch Real 100.

of the SIDD (Smartphone Image Denoising Dataset) dataset. Each *clean image* in SIDD was obtained after processing 150 pictures taken from the same scene [8]. AWGN and our technique were applied in sRGB space, while Noise Flow was applied in raw space. The numbers inside synthesized patches are values of the KL divergence and the KS test computed with respect to the corresponding ground truth, which consists of an actual photograph captured at the target ISO level. For all examples shown in Fig. 2, the images synthesized by our method obtained KL and KS results significantly smaller (better), and textures similar to the corresponding ground truths.

We validate our technique both quantitatively and qualitatively. For this, we use KL divergence, KS test, discriminative evaluation, and t-SNE visualizations. Together, these evaluations show that the proposed model generates noise much closer to natural than existing techniques. Finally, we demonstrate a practical application of our technique: a significant improvement in the performance of a state-of-the-art denoiser (Section 7).

The **contributions** of this work include:

- *A method for adjusting the noise level of an input photograph to match a target ISO level (Section 4). Its results produce significantly better approximations to natural noise than previous synthetic noise generators;*
- *A new loss formulation for use with CycleGANs for allowing the adjustment of noise level* (Section 4.4). Such new loss function results in more realistic noise, whose statistics approximate the ones of real photographs with the same ISO value;
- *A large (unpaired) dataset containing over 2.1 million* 256×256 *patches from photographs captured under different ISO values with a Canon Rebel T3i* (Section 5). The distribution of patches per ISO level is balanced, and the patches did not undergo any denoising, making this a suitable dataset for denoising and noise-synthesis applications.

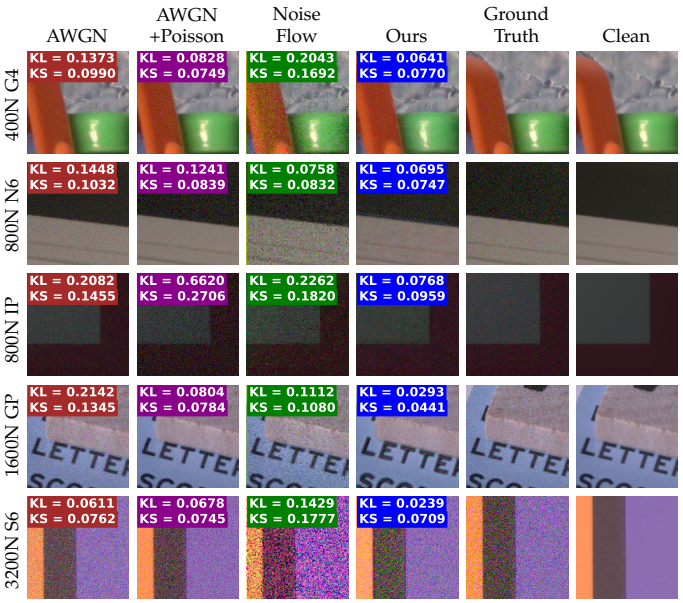

Fig. 2. Comparison of synthesized noise obtained by corrupting a *clean patch* for different ISO values and camera models. AWGN and AWGN+Poisson use the average noise variances computed from the SIDD paired dataset, in sRGB and linear space, respectively. Noise Flow are applied in raw space. Our results achieved the best (smaller) KL divergence and KS values, and exhibit textures similar to the corresponding ground truths (actual photographs taken at the target ISO levels). The clean images are provided as part of the SIDD dataset.

2 Related Work

Our technique focus on adjusting the noise level of an image. Next, we discuss works on denoising, noise synthesis, and paired noise datasets.

2.1 Denoising Methods

Denoising is a well-studied problem and several techniques have been proposed to handle it. Among them, many methods model image priors based on non-local similarities [1], [26], [27], sparse representations [28], [29], [30], [31], total-variation optimization [32], [33], and Markov-Random-Field (MRF) models [34], [35], [36]. Such methods have high-computational costs, heavily relying on the selection of parameter values.

Discriminative learning methods focus on learning inference functions, either based on random-field architectures [37], reaction-diffusion models [10], [38], or conditional random fields [39], [40]. Recently, deep learning has become a trend on denoising methods. Jain and Seung proposed one of the first methods to use CNNs for denoising [41]. Burguer et al. [4] showed how a plain multi-layer perceptron (MLP) trained on large datasets can compete with BM3D [1]. Xie et al. combine sparse coding and denoising autoencoders to address low-level-vision problems such as denoising and inpainting [42]. Mao et al. proposed an autoencoder architecture with symmetric skip connections, training a single model to handle different noise levels [43]. Zhang et al. used a single residual CNN, combined with batch-normalization layers, for blind Gaussian denoising [9]. Later, Guo et al. proposed a convolutional blind denoising network (CBDNet) [44] trained with a noise model more complex than AWGN, surpassing existing methods on benchmarks with real photographs. Lehtinen et al. introduced the idea of learning to denoise using pairs of corrupted images [45]. While it removes the necessity of laboriously collecting noisy-clean pairs for the denoiser training, it still requires at least two realizations of each scene.

The majority of these methods rely on pairs of clean and corrupted images, either to find optimal parameters (for approaches based on image priors), or to fully train discriminative learning techniques. We emphasize that *our technique is not intended to replace denoising methods*. On the contrary, it benefits them by providing more realistic training data, as we demonstrate in the paper.

A recent work by Brooks et al. [46] proposes a technique to invert the transformations performed during the imaging-processing pipeline (gain, color correction, etc.) before performing denoising. Our work, on the other hand, learns how the entire pipeline affects noise. The two techniques could be combined, with our method applied to the "untransformed" images produced by their approach.

2.2 Noise Synthesis

Besides AWGN, a few additional synthetic noise models have been proposed. Foi et al. [47] described one of the first models to try to improve AWGN by combining Poissonian and Gaussian noise. Hwang et al. use a Skellam distribution for modeling Poisson photon noise [48]. More recent approaches propose in-camera imaging models, being able to account for cross-channel noise modeling [49], [50]. Similar to the works described in [49], [50], our data-driven approach takes into account the in-camera pipeline, both in terms of how it can modify the captured noise (*e.g.*, through gamut mapping or demosaicing), as well as accounting for other sources of noise (*e.g.*, quantization). Unlike [49], [50], our method provides a practical solution that learns the marginal distribution of patches with a given ISO setting, allowing it to map an input image from one ISO setting to another. Newson et al. [51] and Eckel et al. [52] presented techniques that try to faithfully model film noise. The work most similar to ours is Noise Flow [25]: a recent machine-learning approach that seeks to minimize the negative log-likelihood (NLL) between the generated and ground-truth noise. We show that our method, besides not needing paired data, is capable of training a denoiser with superior performance compared to Noise Flow.

2.3 Paired Noise Datasets

Recently, researchers have built paired datasets with images captured with different ISO settings. Anaya and Barbu [6] constructed a dataset obtained under low-light conditions, taken with a point-and-shoot (Canon PowerShot S90), a DSLR (Canon EOS Rebel T3i), and a mobile (Xiaomi Mi3) camera. They captured pairs of images (shot at ISO 100 and at a higher ISO value), and showed how to align the pixel intensity values from the pairs of images taken with the same camera. Plotz and Roth [7] propose a similar dataset, capturing images using four different consumer cameras (Sony A7R, Olympus OMD E-M10, Sony RX100 IV, and Nexus 6P), with different sensor sizes. They also propose a post-processing procedure based on the heteroscedastic Tobit regression model to align pixel intensities. Abdelhamed et al. [8] presented the SIDD dataset consisting of 10 scenes, each captured with five smartphone cameras and using different lighting conditions. All these works show that many recent techniques trained for denoising AWGN [4], [30], [37], [38] are outperformed by BM3D [1] when applied to natural noise. This suggests that *the use of AWGN (either for training or evaluation) does not generalize well for the case of natural noise.*

For training our generative model, we have built a large *unpaired dataset* containing over 2.1 million 256×256 image patches taken from photographs captured under different ISO settings (from 100 to 3200) using a Canon Rebel T3i camera. We refer to it as *our_T3i dataset* (to avoid confusion with Renoir T3i [6]). *our_T3i* has a balanced distribution of ISO levels, and consists of sRGB images with no denoising applied. It is a valuable resource for training denoising and noise-synthesis applications. We intend to make this dataset publicly available.

3 Generative Adversarial Networks

This section provides a brief review of Generative Adversarial Networks (GANs), touching on concepts that will be relevant for following sections. GANs were introduced by Goodfellow et al. [53], with many variations appearing in the following years [54], [55], [56]. A GAN consists of two neural networks – a *generator* and a *discriminator* – that are trained together, often aiming to learn to generate images from a specific domain. While the *generator* focuses on learning how to map data from a latent space to the target domain, the *discriminator* learns to distinguish actual elements of the target domain from ones synthesized by the generator. This strategy offers training benefits for both networks: the discriminator helps the generator to synthesize

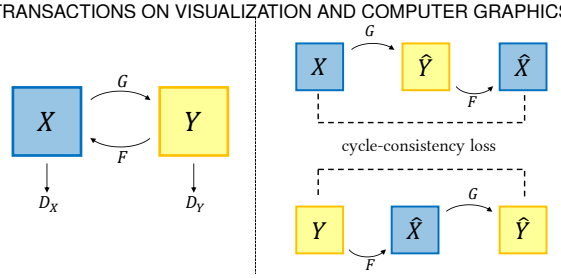

Fig. 3. Unpaired training and cycle-consistency loss. *Discriminators* D_X and D_Y tell if a translated image belongs (or not) to a given domain (left). The cycle-consistency loss enforces G and F to be inverse-like (right).

images that better fit the target domain (via backpropagation), while the generator produces unseen samples to train the discriminator.

Image-to-image translation seeks to obtain two mapping functions: $G : X \mapsto Y$, and $F : Y \mapsto X$, where X and Y are distinct image domains. Normally, for a neural network to learn such functions, it would require paired training data, consisting of the same sample on both domains. Zhu et al. show how the use of discriminators can enable the training on unpaired data [57]. Their technique, called CycleGAN, makes use of two *discriminators*: D_X, which aims at telling whether a given image belongs to the X domain, and D_Y, doing the same for the Y domain. Fig. 3 (left) illustrates how such process works: D_Y encourages G to translate from X to Y, and D_X enforces F to take an image from Y and output the corresponding image inside the image-distribution of domain X. Note that, in this case, G and F are the *generators*, but instead of mapping from a latent space, they map between domains with different image distributions. To constrain such an ill-posed problem, the authors use an additional loss function term called *cycle-consistency loss*. This term enforces that G and F should be inverses (conceptually), making both mappings bijective. In practice, it enforces that given $x \in X$, then $F(G(x)) \approx x$. Likewise, given $y \in Y$, then $G(F(y)) \approx y$ (Fig. 3 (right)). A similar idea was independently proposed by Yi et al. [58].

Existing alternatives for performing unpaired training of image-to-image translation include CoGAN [59], DualGAN [60], and MUNIT [61]. We selected CycleGANs due to its superior results and flexibility for working on many tasks. However, the original CycleGAN formulation is not appropriate for our problem. Thus, we have designed a new loss formulation suited for adjusting the noise level of a given image to match the noise level of a target ISO value.

4 Adjusting Image Noise Levels

Our method uses a GAN architecture inspired by the work of Zhu et al. [57]. Its goal is to learn mapping functions $G : X \mapsto Y$ and $F : Y \mapsto X$ between the domains X and Y. X represents the domain of images captured under a lower ISO setting, and Y represents images captured with a higher ISO. Thus, the mapping function G should learn to increase the noise level, while F should learn to reduce it. During training, two discriminators D_X and D_Y learn to discriminate images belonging to domains X and Y, respectively. Our loss function consists of three terms: an adversarial loss [53], a modified cycle-consistency loss [57], and a novel *low-frequency-consistency loss*.

4.1 Adversarial Loss

Adversarial losses are used to enforce that the output of generators G and F match the image distributions learned by the discriminators D_Y and D_X, respectively. Instead of using the original adversarial loss presented in [53], we use the Least-Squares adversarial loss [62] due to its increased stability during training. For G and D_Y, the adversarial loss is defined as:

$$\mathcal{L}_{\text{LSGAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[D_Y(y)^2] + \mathbb{E}_{x \sim p_{data}(x)}[(1 - D_Y(G(x)))^2], \quad (1)$$

where D_Y outputs 1 when it accepts its input as part of distribution of domain Y, and 0 otherwise. During training, G aims to minimize this loss, while D_Y aims at maximizing it. This way, D_Y is trained to accept images from domain Y and reject the ones generated by $G(x)$. In turn, G is trained to make its output similar to the ones in domain Y (*i.e.*, make its output to be accepted by D_Y). A similar adversarial loss is used for F and D_X, *i.e.*, $\min_F \max_{D_X} \mathcal{L}_{\text{LSGAN}}(F, D_X, Y, X)$.

4.2 Cycle-consistency Loss

Zhu et al. argue that both generators should be cycle-consistent, *i.e.*, for each image $x \in X$, $F(G(x)) \approx x$ [57]. Likewise, for each image $y \in Y$, $G(F(y)) \approx y$. However, *such a constraint is not entirely suited to our problem due to the stochastic nature of noise*. In particular, two noisy photographs of the same scene have significantly different pixel values, despite being corrupted by noise realizations coming from *the same* ISO-level noise distribution. As such, in combination with an l_1 or l_2 norm, the optimization would smooth images x and y to maximize their similarity, being unable to increase noise level.

To overcome this limitation, we propose a new cycle-consistency loss, based on the observation that the noise-to-signal ratio is higher in the high-frequency Fourier components of natural images. Thus, our loss isolates these components, and considers only their total energy (in contrast to exact pixel values). More precisely, it is defined as the l_2-norm $\|\cdot\|_2$ of the channel-wise (R, G, and B) variance (var) of the high-frequencies of the patches x against $F(G(x))$, and likewise for the patches y against $G(F(y))$. Thus, *our cycle-consistency loss* is defined as:

$$\mathcal{L}_{cyclic}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|\text{var}(\mathbb{H}(F(G(x)))) - \text{var}(\mathbb{H}(x))\|_2] + \mathbb{E}_{y \sim p_{data}(y)}[\|\text{var}(\mathbb{H}(G(F(y)))) - \text{var}(\mathbb{H}(y))\|_2], \quad (2)$$

where $\mathbb{H}(x) = x - \mathbb{G}_\sigma(x)$ is the high-frequency content of image patch x, and $\mathbb{G}_\sigma(\cdot)$ denotes a blur using a Gaussian kernel with standard deviation σ. Appendix C discusses how to choose σ values for different ISO values. The low-frequency component of x, $\mathbb{G}_\sigma(x)$, is a key part of our low-frequency-consistency loss term (Section 4.3).

During training, we have noticed no major changes when computing variance over the entire patch or computing moving variance over small windows across the patch. Thus, to speed up the training, the variance is computed once over the entire patch.

4.3 Low-frequency-consistency Loss

The adversarial loss terms (Eq. 1) make the outputs of generators to be accepted by the corresponding discriminators, and the cycle-consistency loss term (Eq. 2) provides additional constraints, enforcing a given input to maintain high frequencies after passing through both G and F. Despite these terms, the problem of image-to-image translation is still ill-posed. For instance, we have found that only using adversarial and cycle-consistency loss terms, the resulting mapping functions tend to not preserve the original colors and global contrast of the input image. We thus introduce another novel loss term that addresses this problem. It is conceptually a dual of the loss from Eq. 2, based on the observation that *given a scene photographed using two different ISO levels, the low-frequency content of both images should be the same, regardless of the used ISO levels*. This happens because, as mentioned earlier, low-frequency components are not significantly corrupted by the relative noise amplitude. Thus, for a given image $x \in X$, the low-frequency contents of both x and $G(x)$ should be similar. The same goes for y and $F(y)$. *Our novel low-frequency-consistency loss* is defined as:

$$\begin{aligned}\mathcal{L}_{low\text{-}freq}(G,F) = \mathbb{E}_{x\sim p_{data}(x)}[\|\mathbb{G}_\sigma(G(x)) - \mathbb{G}_\sigma(x)\|_2] \\ + \mathbb{E}_{y\sim p_{data}(y)}[\|\mathbb{G}_\sigma(F(y)) - \mathbb{G}_\sigma(y)\|_2].\end{aligned} \quad (3)$$

4.4 The Complete Loss Function

The complete loss is then given by:

$$\begin{aligned}\mathcal{L}(G,F,D_X,D_Y) = \mathcal{L}_{\text{LSGAN}}(G,D_Y,X,Y) \\ + \mathcal{L}_{\text{LSGAN}}(F,D_X,Y,X) \\ + \lambda_1\mathcal{L}_{cyclic}(G,F) \\ + \lambda_2\mathcal{L}_{low\text{-}freq}(G,F),\end{aligned} \quad (4)$$

where λ_1 and λ_2 control the importance of each term. After some experimentation, we empirically found that $\lambda_1 = \lambda_2 = 10$ works best. For all results described in the paper we used this setup. The optimization process is guided by

$$G, F = \arg\min_{G,F}\max_{D_X,D_Y} \mathcal{L}(G,F,D_X,D_Y). \quad (5)$$

4.5 Network Architectures

Fig. 4 illustrates the architectures for both generators (top) and discriminators (bottom). For discriminators D_X and D_Y, we use the architecture described by Zhu et al. [57], which consist of a convnet classifier based on PatchGANs [63], [64], with patch size of 70×70 pixels. For the generators G and F, however, we have opted for different architectures.

Different from architectures like U-Net [65], we do not use any downscaling/upsampling layers (nor any convolutional layer with stride different from 1). Our generator G adds the input image x to a residual produced by a modified version of the architecture described by Zhu et al. [57] obtained by removing its `tanh` layer. The output of G for a given input image x is given by

$$G(x) = x + \mathrm{G}_{\text{residual}}(x'), \quad (6)$$

where $\mathrm{G}_{\text{residual}}(x')$ is the residual to be added to the input image x, and x' is the concatenation of x with a random noise image of same size (*i.e.*, for a 3-channel $W \times H$ image x, x' is a 6-channel $W \times H$ image). The noise in x' works like a seed for the generator, guaranteeing that different versions of the output will be produced even if the same image is provided as input multiple times. A similar use of seed for modeling a distribution of possible outputs appears in [66].

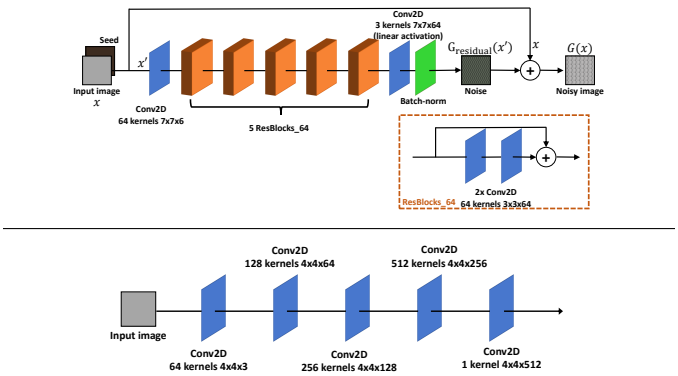

Fig. 4. Architecture of our GANs: (top) our generators consist of five residual blocks, two Conv2D and one Batch-Normalization layers. Each residual block consists of two Conv2D layers. All Conv2D layers (including the ones in the residual blocks) are preceded by a reflection padding and followed by an Instance-Normalization and ReLU layers; (bottom) the discriminators consist of five Conv2D layers, all followed by an Instance-Normalization and leaky-ReLU layer.

The generator F follows the same idea, with the exception that $x' = x$, as F does not require a stochastic behavior to reduce noise. Section 5.1 shows that our architecture is more suited to our problem, being capable of increasing (or attenuating) high-frequency noise. Appendix A presents implementation details, while Appendix B provides a textual description of these architectures.

5 Experiments and Results

We present a series of experiments demonstrating the effectiveness of our technique. Our CNN was trained using our unpaired dataset. The training and test datasets consist of sRGB photographs captured under different ISO levels (ISO 100, ISO 200, ISO 400, ISO 800, ISO 1600, and ISO 3200). All images (total of 13,224) were acquired in raw mode using a Canon EOS Rebel T3i, including photos from many places and object types. The images were then post-processed (demoisaicing, white balancing, gamma correction) using the *rawpy* library with default parameters for outputting images in the sRGB color space in PNG format (lossless compression). This intentionally makes our CNN account for cross-channel effects that might come from demosaicing and gamut mapping, as well as for quantization noise. To speed-up the reads from disk during training, each 5184×3456-pixel image was split into 256×256 patches. We removed patches with mean intensity above 0.5 to avoid over-exposed areas with little noise, which would not contribute to training. The remaining patches were split into training and test sets, consisting of 1,883,501 and 235,318 patches, respectively. During training, we randomly crop a 128×128-pixel region from each 256×256 patch. Data augmentation, in the form of horizontal/vertical flips, and $\pm 90°$ rotations, is used for each cropped region.

Fig. 5. Ablation study, mapping from ISO 100 to ISO 1600. (a) input image patch (captured w/ ISO 100). (b) Result obtained with the model trained with Zhu et al.'s [57] architecture and loss formulation. Note the significant color shift and change in global contrast. (c) output of model trained with Zhu et al.'s architecture and **our** loss formulation. Colors have been better preserved, but contrast has not. (d) output of model trained with the proposed architecture (Section 4.5) and our loss formulation; (e) a patch taken from an image captured with ISO 1600 (ground truth). *We encourage the reader to zoom in for better visualization of the noise.*

5.1 Ablation Study

We present an ablation study that shows the impact of each of our decisions (regarding the loss function and architecture designs). For this study, we have trained our entire CNN (G, F, D_X, and D_Y), X being the domain of photographs captured with ISO 100, and Y the domain of images captured with ISO 1600. The number of training patches was 297,211 for ISO 100, and 303,368 for ISO 1600.

Fig. 5 compares three training experiments: Fig. 5(b) shows results obtained using Zhu et al.'s [57] loss functions and architecture. Fig. 5(c) shows results obtained using our modified cycle-consistency loss (Eq. (2)) and our low-frequency-consistency loss (Eq. (3)), but still using Zhu et al.'s [57] generator architecture. Fig. 5(d) shows results obtained using our loss formulation (Eq. (5)) and architectures for generators G and F (see Section 4.5). Fig. 5(e) shows the corresponding patch taken from an image captured with ISO 1600 (ground truth). All the models were trained using the same training set and with the same number of iterations (4 epochs each). Notice how the additional constrains imposed by the low-frequency-consistency loss enforce color and global scene contrast preservation. Our convolutional architecture, combined with residual strategy (Eq. (6)) enables the network to better learn how to synthesize (in the case of G) and remove (in the case of F) high-frequency noise.

5.2 Multi-ISO Mapping

One possible strategy for transforming between arbitrary pairs of ISO values is to define a chain of mappings between pairs of adjacent ISO levels. However, mappings between adjacent ISO values tend to undershoot the noise, behaving like an identity function. This happens because the domains of neighbor ISO-levels are very similar, making it very hard for discriminators to learn how to separate such domains. This is illustrated in Fig. 6, where the confusion matrix for

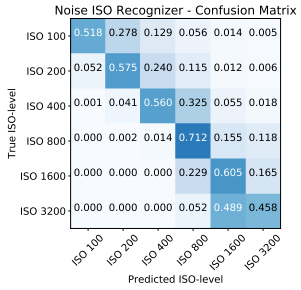

Fig. 6. Difficulty in distinguishing adjacent ISO levels: normalized confusion-matrix obtained for an ISO-level classifier. Note the higher confusion values around the main diagonal.

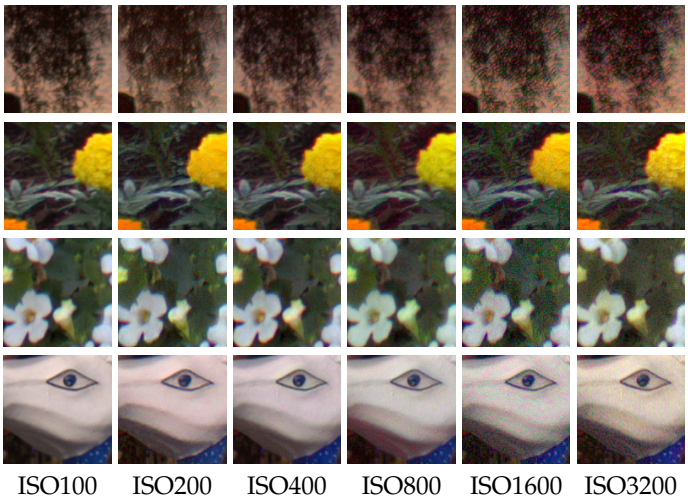

Fig. 7. Our CNN can adjust the noise level of input images to match that of a target ISO value. In all these examples, our CNN takes an input patch captured with ISO 100 and maps it to various target ISO values: 200, 400, 800, 1600, and 3200. We encourage the reader to zoom in these images to inspect the noise levels.

Fig. 8. Photographs mapped from ISO 1600 to ISO 100 using our technique. Zoom in to see the reduced noise in the synthetic versions.

a classifier that detects the ISO of a given patch got an average accuracy of only 59.03%. This shows how hard it is to distinguish between adjacent ISO values. Details about the classifier's architecture can be found in Appendix D.

An inspection of the confusion matrix in Fig. 6 shows that it is virtually impossible to distinguish between ISO 1600 and ISO 3200 photographs, as well as between ISO 100 and ISO 200 pictures. The inspection also reveals that one can confidently map between ISO 100 and all other ISO values, except ISO 200. Thus, we designed a multi-ISO mapping scheme that takes advantage of this observation. The mapping between ISO 100 and ISO 200 is done indirectly through ISO 1600. By transitivity, it allows mappings among all pairs of ISO values.

Fig. 7 illustrates the use of our technique to adjust the

noise levels of several photographs taken at ISO 100 to match different ISO values, from 200 to 3200. Fig. 8 shows the use of our method to reduce the noise level of five photographs originally taken with ISO 1600 to ISO 100. We encourage the reader to zoom in for better visualization of the increased and reduced noise.

6 Evaluation

To evaluate our method we use the small version of SIDD [8], which consists of paired noisy-clean images captured under different ISO values with five different smartphone cameras: LG G4 (G4), Google Pixel (GP), iPhone 7 (IP), Motorola Nexus 6 (N6), and Samsung Galaxy S6 Edge (S6). Having a paired dataset allows us to compare our results both quantitatively and qualitatively. It also enables the computation of the standard deviation of the noise for each ISO value, which is used to obtain a fair implementation of AWGN. Moreover, since Noise Flow [25] was trained in such dataset, the use of SIDD also allows for a fair comparison with this method.

Section 6.1 describes the five noise models compared to ours. In Section 6.2 we use a discriminative model trained to classify noise between natural and synthetic (produced by previous models). Section 6.3 presents another classifier trained to distinguish among natural noise and the several noise models described in Section 6.1, including ours. Finally, Section 6.4 compares the performance of our technique against the most competitive ones using KL divergence and KS test as objective metrics. Such experiment measures the similarities between each model's noise distributions and the noise distributions of actual photographs taken at the target ISO levels (ground truth). We also qualitatively compare results obtained with these three techniques for different ISO values and camera models.

6.1 Noise Models

We call an algorithm used to corrupt a clean patch a *noise model*. We consider five popular/recent noise models for comparisons against ours:

AWGN: the most traditional noise model, consisting of zero-mean Gaussian noise[1].. By having paired noisy-clean patches from the SIDD, we compute the average noise-variance for each ISO value. Such value is used when synthesizing noise using AWGN for the corresponding ISO level;

Poisson: another traditional noise model, synthesizes noise following a Poissonian distribution[6.1];

GaussianPoissonian: a composition of the two previous noise models (AWGN+Poisson), inspired by [47]. First, we apply AWGN to the input, then apply Poisson noise to the intermediate result.

AWGN followed by Bayer sampling and demosaicing: consisting of the application of AWGN followed by sampling (simulating a Bayer color filter array – CFA) and demosaicing (using bilinear interpolation). A similar noise model is used by CBDNet [44];

1. Generated with the *random_noise* function from the python package *skimage*

Noise Flow [25]: a deep learning based model for synthesizing noise for different types of cameras and ISO values. As the provided models were trained in raw space, we synthesize (and add) noise in raw before converting the images to sRGB.

We refer to our models trained on the small SIDD dataset as $\mathbf{GAN_{SIDD}}$. Although SIDD provides noisy-clean pairs, the training of our generative models was unsupervised (*i.e.*, did not use such paired information). The small version of the SIDD dataset contains 160 pictures, and only their most representative ISO values were used for training our models: ISO 400 (17 pictures), ISO 800 (36 pictures), ISO 1600 (22 pictures), and ISO 3200 (13 pictures). This resulted in 88 images and a total of 18,951 256×256 training patches distributed over these four ISO levels and five camera models. We trained one GAN_{SIDD} for each combination of ISO level and camera model. Training one model per ISO value regardless of camera model results in *mode collapsing* [53], where the generator learns to mimic only one camera model. Each model was trained for 40 epochs using 128×128 random crops from the training patches (to expedite training).

Except for Noise Flow, the remaining algorithms operate directly in the sRGB space. For each scene, SIDD provides a so called *clean image*, obtained after processing 150 pictures taken from the same scene [8]. For the comparisons described in the following sections, all noise models take as input one 256×256 *clean patch* at a time. Their outputs are used as input to discriminative models, and the images and noise distributions are compared to patches from actual photographs (ground truth) taken at the target ISO level.

6.2 Classifying Natural and Artificial Noise

An evaluation network was trained for classifying a given input into one of the two classes: (i) corrupted with natural noise, or (ii) corrupted by traditional noise models (*i.e.*, AWGN, Poissonian, GaussianPoissonian, and AWGN followed by Bayer sampling and demosaicing). During training, each patch had a 20% probability of containing natural noise (*i.e.*, taken directly from an actual noisy photography in the SIDD dataset). There was also a 20% probability that a training patch is obtained by corrupting a clean patch with each one of the four noise models (adding up to the remaining 80%). Details of the architecture of this binary-classification network can be found in Appendix D. The obtained classifier achieved an accuracy above 99.75% on the test set on all ISO levels. This simple experiment shows how easily separable the sets of images corrupted by natural and by artificially-generated noise are. It also shows how *current synthetic noise models fail to mimic natural noise found in digital photographs*.

6.3 Discriminating among Noise Models

We also trained classifiers for discriminating images containing natural noise as well as ones corrupted by the six noise models described in Section 6.1, which include ours. We trained a single classifier for each ISO level (400, 800, 1600, and 3200), regardless of the used cameras models. Although the trained classifiers are capable of discriminating between natural and our GAN-generated noise, we show that the noise produced by our generative models is much

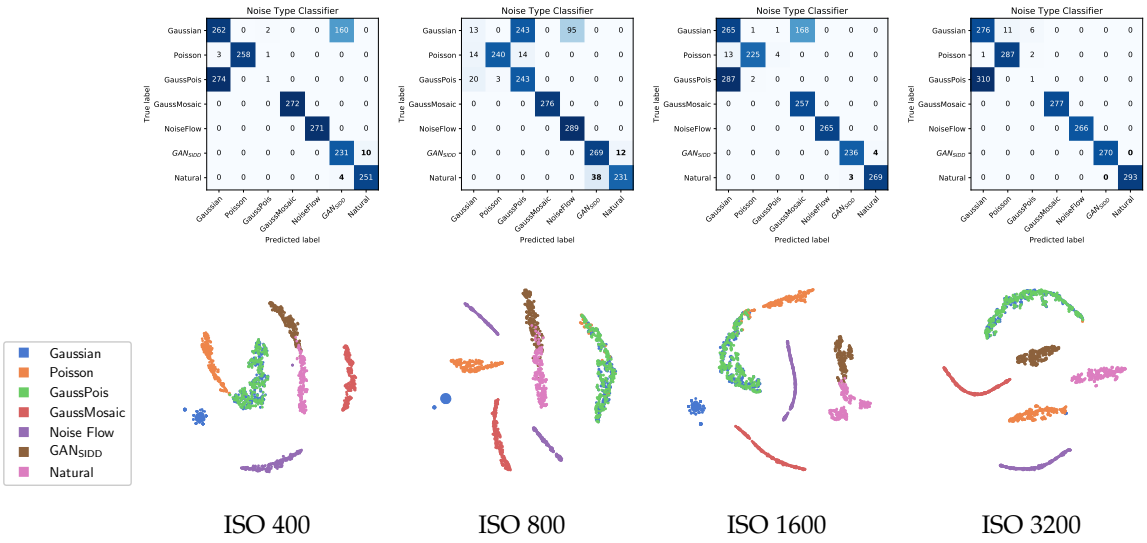

Fig. 9. Classification of images corrupted by several noise models, including our GAN_{SIDD} trained on the small SIDD dataset. We trained one classifier for each ISO level. (top) Confusion matrices obtained with a 2,000-patch testset for each ISO level, including multiple camera models per ISO. These matrices exhibit noticeable confusion between natural noise and our GAN_{SIDD} noise, specially for the ISO levels 800. (bottom) Projections using t-SNE for the same test sets. The projections of the noise generated by our GAN_{SIDD} overlap with natural for most ISO values.

TABLE 1
KL divergences and KS values for different noise models, ISO values, and lighting conditions ([L]ow and [N]ormal light brightness levels). Best (lower) values are highlighted in bold. These values were measured over the whole population of image patches (see the text for details). Notice how our method achieves better values for these metrics across all ISO values.

		ISO 400		ISO 800		ISO 1600		ISO 3200	
		L	N	L	N	L	N	L	N
KL divergence	AWGN	0.0910	0.1063	0.0938	0.1108	0.1780	0.1478	0.0765	0.0796
	AWGN+Poisson (linear)	0.1358	0.0781	0.2053	0.1010	0.0732	0.1786	0.0784	0.0383
	NoiseFlow	0.1052	0.1557	0.0593	0.1140	0.0517	0.0634	0.0801	0.0520
	GAN_{SIDD}	**0.0091**	**0.0323**	**0.0104**	**0.0249**	**0.0228**	**0.0129**	**0.0339**	**0.0188**
KS value	AWGN	0.0693	0.0787	0.0842	0.0909	0.1096	0.1237	0.1100	0.0896
	AWGN+Poisson (linear)	0.1000	0.0668	0.1520	0.0726	0.0937	0.1277	0.0784	**0.0628**
	NoiseFlow	0.1138	0.1422	0.0900	0.1057	0.0929	0.0821	0.1270	0.0925
	GAN_{SIDD}	**0.0333**	**0.0564**	**0.0404**	**0.0550**	**0.0600**	**0.0643**	**0.0761**	0.0677

closer to natural than the ones created by previous noise models. Fig. 9 shows the results of these classifiers. The confusion matrices on the top were created with 2,000 test patches for each ISO level. Notice that the confusion between natural and our GAN generated noise is higher than with the previous noise generators. The t-SNE visualizations at the bottom of Fig. 9 show the output of the last layer of each classifier. t-SNE is a visualization technique that uses a non-linear transformation to project the data onto a 2D plane, trying to keep the pairwise distance between samples [67]. Note that the projections of other noise models are further away than ours (brown) from natural noise (pink). The projections of the noise generated by our GAN_{SIDD} overlap with natural for most ISO values. The labels of each sample (AWGN, Poisson, natural, etc.) were not used for learning the t-SNE projections, but only for coloring the visualizations *after* t-SNE has been applied.

The use of classifiers is a known strategy for measuring the quality of generative methods [68]. According to Lopez-Paz and Oquab, the lower the classifier's accuracy when discriminating between natural and synthesized samples, the higher the quality of the generative model. It is important to mention that when evaluating trained generative models [68], all experiments got near-perfect results (real and synthetic samples being easily distinguishable). In a similar experiment presented in this section (Fig. 9), the classifier failed to distinguish our results from natural noise in several occasions.

6.4 Quantitative Metrics

For quantitative evaluation, we compared patches containing natural noise with synthesized ones produced using AWGN, AWGN+Poisson, Noise Flow, and our GAN models. We use the KL divergence and KS test as objective metrics for assessing the similarity of the noise distributions generated by each technique with the noise distributions obtained from natural noise. Lower values for KL and KS represent better results (more similar to real noise).

Fig. 10 shows additional quantitative and qualitative comparisons (besides the ones in Fig. 2) of patches from different ISO values and lighting conditions (L for low and N for normal light-brightness levels) from the small SIDD dataset. The variances used for AWGN and AWGN+Poisson-linear are mean variances measured in sRGB (post-gamma) and in linear space (obtained by applying inverse gamma to sRGB), respectively. For each ISO level, the variances were computed from all noisy-clean image pairs for the given ISO value. Noise Flow were applied in raw space, with metadata (camera and ISO value)

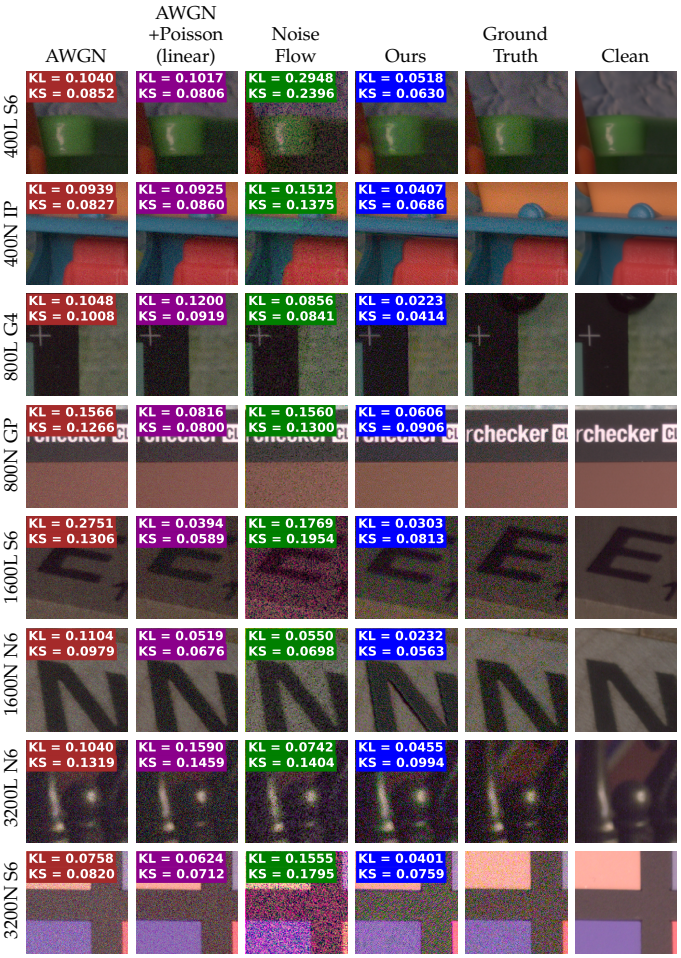

Fig. 10. Additional comparisons of synthesized noise obtained by corrupting *clean patches* for different ISO values and camera models. Our results achieved the best (smaller) KL divergence and KS values. Ground truth is an actual photograph taken at the target ISO level. The clean images are provided as part of the SIDD dataset.

taken from the patch[2], using a trained model provided by the authors. AWGN+Poisson-linear was applied in linear space. Both AWGN and our GAN_{SIDD} models are applied directly in the sRGB space. Both metrics (KL divergence and KS value) are computed in the sRGB space.

Note in Fig. 10 how our model achieves smaller values of KL divergence, despite of not using paired data at all. Table 1 compares these metrics, for the most competitive methods, for the whole population of patches from SIDD. The metrics were obtained by computing the Red, Green and Blue histograms of all patches as a group, resulting in three numbers per population (KL_R, KL_G, KL_B), which are averaged together. The same procedure is performed for KS values. For more details please refer to Appendix E. Appendix F provides a similar Table comparing additional methods. As seen in Table 1, our approach obtains the best metrics across practically all scenarios, attesting to its effectiveness in synthesizing noise that is statistically similar to natural noise. Appendix G provides more comparison cases for visual inspection.

2. Our statistical metrics are computed in sRGB space, and we noticed that the raw-to-sRGB postprocessing implemented by Noise Flow differs from the one implemented by SIDD. To be fair in our comparisons, the metrics for Noise Flow were computed against the clean sRGB images provided in the Noise Flow package.

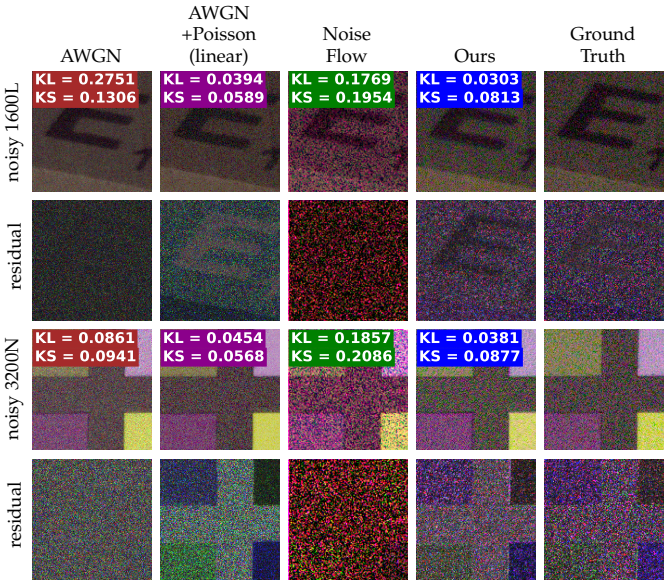

Fig. 11. Comparison of synthesized noisy images and corresponding noise (residual) produced by the various methods. For better visualization, contrast of the residual images has been enhanced by factor of $3\times$. Our results better mimic the noise found in digital photographs.

Fig. 11 compares the synthesized noise (residual), *i.e.*, the difference between the noisy and clean image versions, generated by the noise models in Fig. 10. Notice how our method better mimics camera noise, specially regarding the size of noise grain and color distribution. Appendix H provides additional examples of residual images.

7 Applications

To demonstrate a potential application of our noise model, we trained several versions of the DnCNN method for blind denoising [9]. We have used the authors' implementation, using the DnCNN-S architecture, which consists of 17 Conv2D layers with 64 filters each. Each version of the DnCNN denoiser is trained on an extensive noisy-clean paired dataset created with a different realization of our GAN. More precisely, we select a set of clean (noise-free) images from the dataset of Gharbi et al. [69], and compute the corresponding noisy images using our GAN_{D} noise generator (for several ISO values). The subscript D indicates on which (unpaired) dataset the GAN_{D} noise model was trained: $\text{D} \in \{\text{our\_T3i}, \text{SIDD}\}$. Although we describe a denoising application, in principle, our method could be used to generate input to any trainable computer-vision task that seeks robustness to noisy scenarios.

To compare our noise generator against existing alternatives, we also trained several DnCNN denoisers using paired datasets created by the AWGN and Poissonian-Gaussian models (both applied in linear and post-gamma spaces - see Table 2), CBDNet [44] and Noise Flow [25] (for these last two, we used implementations provided by their authors). For all experiments, we trained the denoisers past convergence: taking the checkpoint with highest PSNR on the validation set. We used the same data-augmentation in the training of all models (see Appendix I for details).

Table 2 summarizes the results of the DnCNN denoisers trained with our GAN noise models versus the alternatives.

TABLE 2
PSNR results of evaluating a denoiser (DnCNN) trained using noise generated by existing techniques, and by using the generators trained with our Canon dataset, and with SIDD dataset. The version of DnCNN trained using images corrupted by both GANs (trained with Canon and SIDD datasets) achieved higher PSNR (in bold) in all natural-image benchmarks.

		Renoir			Darmstadt	SIDD	
id	Noise Model	T3i	Mi3	S90		Low lighting	Normal lighting
1	CBDNet [44]	27.35	25.08	26.62	32.58	24.48	28.99
2	AWGN	29.12	25.07	28.91	30.63	27.07	29.31
3	AWGN-linear	29.64	25.48	29.48	32.13	29.17	31.20
4	AWGN+Poisson	29.61	25.18	29.31	31.15	27.30	29.71
5	AWGN+Poisson-linear	30.02	25.94	29.70	32.11	29.93	31.67
6	Noise Flow [25]	30.62	26.15	29.93	32.84	29.53	31.68
7	GAN_{our_T3i}	**33.51**	28.09	31.60	32.24	30.33	31.48
8	GAN_{SIDD}	32.16	27.38	31.04	32.79	31.85	33.02
9	GAN_{our_T3i} + GAN_{SIDD}	33.49	**28.14**	**31.69**	**35.32**	**33.27**	**34.67**

We evaluate each denoiser using the Renoir [6], Darmstadt [7], and SIDD [8] denoising benchmarks. In these, we discriminate each camera model in the Renoir dataset, as well as between scenes captured under low (L) and normal (N) light brightness levels in the SIDD dataset. As can be seen by the higher PSNR values in Table 2, using our GAN noise models lead to improved performance of the DnCNN denoiser across all benchmarks.

Compared to previous noise generators (rows 1 to 6 in Table 2), the denoiser trained only with noisy images generated by GAN_{our_T3i} (row 7 in Table 2) got a significant improvement in PSNR performance, especially in the Renoir dataset. Similarly, when trained only with GAN_{SIDD} (row 8 in Table 2), the denoiser also performed significantly better than previous approaches, in particular on the SIDD dataset. Intriguingly, a more generic denoiser trained using randomly selected patches generated by GAN_{our_T3i} and GAN_{SIDD} got the overall best results (row 9 in Table 2). We conjecture that the combined use of the two models might help to fight overfitting, thus explaining the improved performance. Indeed, while the denoisers separately trained with GAN_{our_T3i} and GAN_{SIDD} both converged between epochs 3 and 4, the denoiser trained with both noise models (GAN_{our_T3i} + GAN_{SIDD}) converged after epoch 5.

8 Discussion

The classifiers for identifying the ISO level of a given patch (Section 5.2), for distinguishing natural from artificial noise (Section 6.2), and for noise-model classification (Section 6.3) all share similar architectures. The architectures of the GAN generators (Section 4.5) and of these classifiers were obtained after experimenting with different designs (*e.g.*, residual network [70], U-Net [65]) and different deep-learning layers (*e.g.*, batch-normalization [71], half-strided convolutions, and instance normalization [72]). We have also tried different metrics for the cycle-consistency loss (Section 4.2), such as l_1 and l_2 norm, and SSIM.

Using a residual design (Eq. (6)) for our generators greatly improved the convergence of the training and the quality of the results. In this same residual design, we include a batch-normalization as the last layer of the generator, initializing Γ (the scaling parameter) as 0.1, making Eq. (6) to be close (at initialization) to an identity function. This actually speeds up the training convergence, while reducing generated artifacts. As mentioned in Section 4.1, we replaced the traditional adversarial loss by the least-squares adversarial loss, as this has been reported to increase training stability [62]. While the *tanh* activation function is often used as the last layer of generators, for our application it tends to saturate highlights and darken low-lit regions. Removing *tanh* improved our results.

Adding noise to raw image values more closely simulates most noise sources, besides being simpler to implement compared to simulating these processes in demosaiced, white-balanced, quantized, and compressed images. However, doing so would preclude our method from being used with images post-processed by such transformations (such as JPEG, PNG, etc.), which are used by most applications. Therefore, we perform all training in the sRGB color space. This was a design decision where we favored wider applicability over slightly better precision. Nonetheless, by retraining our models for input/output raw pixel values, our method can be used with RAW images.

A key factor for AWGN-based noise-models is the choice of the variance. In our KL/KS comparison, as well as in the denoiser experiment, the variance for AWGN was computed for each ISO level using the paired noisy-clean images from SIDD[3]. When such paired dataset is unavailable, the variance must be estimated or guessed, resulting in a poorer representation of the natural noise. Our method, on the other hand, does not rely on paired datasets, thus, the gap between results obtained with our and AWGN- based methods should increase in practical applications.

Training GANs is hard not only due to problems during training (*e.g.*, mode collapsing, non-convergence, diminished gradients, and sensibility to hyperparameters), but specifically to potential artifacts introduced by image-generative models, like ringings and blurring. The fact that our GAN architecture is capable of generating high-frequency noise illustrates its potential. Nonetheless, color shifts and halos may happen in some situations. The cause of these artifacts are not quite clear, and its understanding and correction is a subject for future investigation. Despite such artifacts, our model provides a better alternative to existing noise generators, as shown in Table 2.

Noise encountered in photographs of different camera models might exhibit different noise statistics [8]. We show how our GAN model can be trained for generating noise mimicking multiple camera models (*e.g.*, Canon T3i and several smartphone cameras from SIDD). We show that the performance of a denoiser trained using such networks is

3. We computed the variance for the post-gamma and linear spaces.

greatly improved on several benchmarks, which demonstrates the potential and generalization of our method. For instance, the denoiser trained with GAN_{our_T3i} + GAN_{SIDD} got improved PSNR on cameras Mi3 and S90 (row 9 in Table 2), even when our generative models were not trained with images from such cameras.

We have demonstrated that our generative models can be trained in an unsupervised way using small, unpaired datasets. Thus, it can be easily applied to other camera models. Its main limitation is the mode collapsing observed when training a single generator for several camera models. While we have dealt with this issue by training one generator for each camera model, ideally this problem should be addressed directly in the GAN architecture. This is the subject of future investigation.

9 Conclusion

We presented a practical data-driven technique for adjusting the noise level of input photographs to match target ISO levels. Our solution learns the mappings among different ISO levels from unpaired data using GANs, for which we defined a new loss formulation and network architectures tailored to the problem. An ablation study justifies our decisions, confirming the superior results achieved when using our method. By not requiring a paired noisy-clean dataset, our technique can be trained for any camera model by just collecting photographs for the various ISO levels.

We demonstrate the effectiveness of our approach both quantitatively and qualitatively, by demonstrating its superior performance over previous methods. As a practical application, we have shown that images generated by our technique greatly improve the performance of a state-of-the-art trainable denoiser. Our source code and trained models are available at our project repository[4].

Several applications can benefit from our technique. This includes not only trainable denoising methods, but also demosaicing [15], [69] and image-reconstruction [14], forgery detection [16], as well as computer-vision tasks seeking noise robustness [11], [12], [13].

Acknowledgments

This work was sponsored by CNPq-Brazil (fellowships and grants 312975/2018-0, 436932/2018-0, 423673/2016-5), and financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. We would like to thank NVIDIA for donating the GeForce GTX Titan X GPU used for this research.

References

[1] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE TIP*, vol. 16, no. 8, pp. 2080–2095, 2007.

[2] C. Liu, R. Szeliski, S. B. Kang, C. L. Zitnick, and W. T. Freeman, "Automatic estimation and removal of noise from a single image," *IEEE TPAMI*, vol. 30, no. 2, pp. 299–314, 2008.

[3] A. Barbu, "Training an active random field for real-time image denoising," *IEEE TIP*, vol. 18, no. 11, pp. 2451–2462, 2009.

[4] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?" in *IEEE CVPR*, 2012, pp. 2392–2399.

4. https://github.com/bernardohenz/synt_noise_GANs

[5] Y. Chen, T. Pock, R. Ranftl, and H. Bischof, *Revisiting Loss-Specific Training of Filter-Based MRFs for Image Restoration*, 2013, pp. 271–281.

[6] J. Anaya and A. Barbu, "Renoir - a dataset for real low-light noise image reduction," *arXiv preprint arXiv:1409.8230*, 2014.

[7] T. Plotz and S. Roth, "Benchmarking denoising algorithms with real photographs," in *IEEE CVPR*, vol. 00, 2017, pp. 2750–2759.

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

[9] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE TIP*, vol. 26, no. 7, pp. 3142–3155, 2017.

[10] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE TPAMI*, vol. 39, no. 6, pp. 1256–1272, 2017.

[11] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *QoMEX*, 2016, pp. 1–6.

[12] S. Diamond, V. Sitzmann, S. P. Boyd, G. Wetzstein, and F. Heide, "Dirty pixels: Optimizing image classification architectures for raw sensor data," *CoRR*, vol. abs/1701.06487, 2017.

[13] S. S. Roy, S. I. Hossain, M. A. H. Akhand, and K. Murase, "A robust system for noisy image classification combining denoising autoencoder and convolutional neural network," *IJACSA*, vol. 9, no. 1, 2018.

[14] S. Mandal, A. Bhavsar, and A. K. Sao, "Noise adaptive super-resolution from single image via non-local mean and sparse representation," *Signal Proc.*, vol. 132, pp. 134 – 149, 2017.

[15] B. Henz, E. S. L. Gastal, and M. M. Oliveira, "Deep joint design of color filter arrays and demosaicing," *CGF*, vol. 37, no. 2, pp. 389–399, 2018.

[16] M. Kaur and S. Walia, "Forgery detection using noise estimation and hog feature extraction," *Int. J. Multimed. Ubiquitous Eng*, vol. 11, no. 4, pp. 37–48, 2016.

[17] F. Alter, Y. Matsushita, and X. Tang, *An Intensity Similarity Measure in Low-Light Conditions*, 2006, pp. 267–280.

[18] S. W. Hasinoff, F. Durand, and W. T. Freeman, "Noise-optimal capture for high dynamic range photography," in *IEEE CVPR*, 2010, pp. 553–560.

[19] S. W. Hasinoff, "Photon, poisson noise," in *Computer Vision*. Springer US, 2014, pp. 608–610.

[20] EMVA, "Emva standard 1288 release 3.1," 2016.

[21] D. Makovoz, "Noise variance estimation in signal processing," in *IEEE ISSPIT*, 2006, pp. 364–369.

[22] X. Liu, M. Tanaka, and M. Okutomi, "Single-image noise level estimation for blind denoising," *IEEE TIP*, vol. 22, no. 12, pp. 5226–5237, 2013.

[23] G. Chen, F. Zhu, and P. A. Heng, "An efficient statistical method for image noise level estimation," in *IEEE ICCV*, 2015, pp. 477–485.

[24] M. Petrovic, D. Petrovic, and G. Nikolic, "An approach to noise variance estimation in very low signal-to-noise ratio stochastic signals," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 10, pp. 407–410, 2016.

[25] A. Abdelhamed, M. A. Brubaker, and M. S. Brown, "Noise flow: Noise modeling with conditional normalizing flows," in *IEEE ICCV*, 2019.

[26] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *IEEE CVPR*, vol. 2, 2005, pp. 60–65 vol. 2.

[27] A. Buades, B. Coll, and J.-M. Morel, "Nonlocal image and movie denoising," *IJCV*, vol. 76, no. 2, pp. 123–139, 2008.

[28] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE TIP*, vol. 15, no. 12, pp. 3736–3745, 2006.

[29] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *IEEE ICCV*, 2009, pp. 2272–2279.

[30] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *IEEE TIP*, vol. 22, no. 4, pp. 1620–1630, 2013.

[31] R. Giryes and M. Elad, "Sparsity-based poisson denoising with dictionary learning," *IEEE TIP*, vol. 23, no. 12, pp. 5057–5069, 2014.

[32] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259 – 268, 1992.

[33] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 460–489, 2005.

[34] X. Lan, S. Roth, D. Huttenlocher, and M. J. Black, "Efficient belief propagation with learned higher-order markov random fields," in *ECCV*, 2006, pp. 269–282.
[35] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. Springer-Verlag London, 2009.
[36] S. Roth and M. J. Black, "Fields of experts," *International Journal of Computer Vision*, vol. 82, no. 2, p. 205, Jan 2009.
[37] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *IEEE CVPR*, 2014, pp. 2774–2781.
[38] Y. Chen, W. Yu, and T. Pock, "On learning optimized reaction diffusion processes for effective image restoration," in *IEEE CVPR*, 2015.
[39] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth, "Discriminative non-blind deblurring," in *IEEE CVPR*, 2013, pp. 604–611.
[40] U. Schmidt, J. Jancsary, S. Nowozin, S. Roth, and C. Rother, "Cascades of regression tree fields for image restoration," *IEEE TPAMI*, vol. 38, no. 4, pp. 677–689, 2016.
[41] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *NIPS*, 2009, pp. 769–776.
[42] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *NIPS*, 2012, pp. 341–349.
[43] X. Mao, C. Shen, and Y. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *NIPS*, 2016, pp. 2802–2810.
[44] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, "Toward convolutional blind denoising of real photographs," *IEEE CVPR*, 2019.
[45] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2Noise: Learning image restoration without clean data," in *ICML*, vol. 80, 2018, pp. 2965–2974.
[46] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron, "Unprocessing images for learned raw denoising," in *IEEE CVPR*, 2019.
[47] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, "Practical poissonian-gaussian noise modeling and fitting for single-image raw-data," *IEE TIP*, vol. 17, no. 10, pp. 1737–1754, 2008.
[48] Y. Hwang, J. Kim, and I. S. Kweon, "Difference-based image noise modeling using skellam distribution," *IEEE TPAMI*, vol. 34, no. 7, pp. 1329–1341, 2012.
[49] S. J. Kim, H. T. Lin, Z. Lu, S. Süsstrunk, S. Lin, and M. S. Brown, "A new in-camera imaging model for color computer vision and its application," *IEEE TPAMI*, vol. 34, no. 12, pp. 2289–2302, 2012.
[50] 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 *IEEE CVPR*, 2016, pp. 1683–1691.
[51] A. Newson, J. Delon, and B. Galerne, "A stochastic film grain model for resolution-independent rendering," *CGF*, vol. 36, no. 8, pp. 684–699, 2017.
[52] S. Eckel, P. Huthwaite, U. Zscherpel, A. Schumm, and N. Paul, "Realistic film noise generation based on experimental noise spectra," *IEEE TIP*, vol. 29, pp. 2987–2998, 2020.
[53] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS 27*, 2014, pp. 2672–2680.
[54] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.
[55] A. Odena, "Semi-Supervised Learning with Generative Adversarial Networks," *ArXiv e-prints*, 2016.
[56] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *CoRR*, vol. abs/1606.03657, 2016.
[57] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE ICCV*, 2017, pp. 2242–2251.
[58] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *IEEE ICCV*, vol. 00, 2017, pp. 2868–2876.
[59] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *NIPS 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 469–477.
[60] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *IEEE ICCV*, 2017, pp. 2868–2876.
[61] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *ECCV*, 2018.
[62] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, and Z. Wang, "Multi-class generative adversarial networks with the L2 loss function," *CoRR*, vol. abs/1611.04076, 2016.
[63] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.
[64] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks." in *ECCV*, vol. 9907, 2016, pp. 702–716.
[65] O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, ser. LNCS, vol. 9351, 2015, pp. 234–241.
[66] J. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *NIPS*, 2017, pp. 465–476.
[67] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *JMLR*, vol. 9, pp. 2579—2605, 2008.
[68] D. Lopez-Paz and M. Oquab, "Revisiting classifier two-sample tests," in *ICLR*, 2017.
[69] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, "Deep joint demosaicking and denoising," *ACM TOG*, vol. 35, no. 6, pp. 191:1–191:12, 2016.
[70] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.
[71] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML 37*, 2015, pp. 448–456.
[72] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016.

Bernardo Henz is an Assistant Professor at the Farroupilha Federal Institute (IFFar) in Brazil. He is also a Computer Science Ph.D. candidate at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS). His research interests include image processing, computational photography, and machine/deep learning.

Eduardo S. L. Gastal is an Assistant Professor of Computer Science at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS), in Brazil. His work focuses on efficient and novel filters for images and videos, and he is generally interested in image and signal processing, computer graphics, and applied mathematics. His work on efficient high-dimensional filtering with Dr. Manuel M. Oliveira was awarded the inaugural ACM SIGGRAPH Outstanding Doctoral Dissertation Award (2016) and first place in the Brazilian Computer Society's Doctoral Dissertation Contest. He is associate editor for Elsevier Computers & Graphics.

Manuel M. Oliveira is a Full Professor at the Federal University of Rio Grande do Sul (UFRGS) in Brazil. He received his Ph.D. from the University of North Carolina at Chapel Hill in 2000. Before joining UFRGS in 2002, he was an assistant professor at SUNY Stony Brook (2000—2002). In the 2009–2010 academic year, he was a visiting associate professor at the MIT Media Lab. His research interests include computer graphics, image processing, pattern recognition, computational photography, machine learning, and vision (both human and machine). He is an associate editor of ACM TOG and a former associate editor of IEEE TVCG and IEEE CG&A.