Abstract

Generative adversarial networks (GANs) are neural networks that learn data distributions through adversarial training. In intensive studies, recent GANs have shown promising results for reproducing training images. However, in spite of noise, they reproduce images with fidelity. As an alternative, we propose a novel family of GANs called noise robust GANs (NR-GANs), which can learn a clean image generator even when training images are noisy. In particular, NR-GANs can solve this problem without having complete noise information (e.g., the noise distribution type, noise amount, or signal-noise relationship). To achieve this, we introduce a noise generator and train it along with a clean image generator. However, without any constraints, there is no incentive to generate an image and noise separately. Therefore, we propose distribution and transformation constraints that encourage the noise generator to capture only the noise-specific components. In particular, considering such constraints under different assumptions, we devise two variants of NR-GANs for signal-independent noise and three variants of NR-GANs for signal-dependent noise. On three benchmark datasets, we demonstrate the effectiveness of NR-GANs in noise robust image generation. Furthermore, we show the applicability of NR-GANs in image denoising. Our code is available at https://github.com/takuhirok/NR-GAN/.

1. Introduction

In computer vision and machine learning, generative models have been actively studied and used to generate or reproduce an image that is indistinguishable from a real image. Generative adversarial networks (GANs) [21], which learn data distributions through adversarial training, have garnered special attention owing to their ability to produce high-quality images. In particular, with recent advancements [2, 56, 47, 23, 41, 59, 60, 36, 81], the latest GANs (e.g., BigGAN [6] and StyleGAN [37]) have succeeded in generating images indistinguishable for humans.

However, a persistent issue is that recent high-capacity GANs could replicate images faithfully even though the training images were noisy. Indeed, as shown in Figure 1(b)(e), when standard GAN is trained with noisy images, it attempts to recreate them. Although the long-term development of devices has steadily improved image quality, image degradation is unavoidable in real situations. For example, electronic noise is inevitable in digital imaging [63, 1] and estimator variance often appears as noise in graphic rendering [88, 77]. Therefore, susceptibility to noise is practically undesirable for GANs.

The question becomes: “How can we learn a clean image generator even when only noisy images are available for training?” We call this problem noise robust image generation. One solution is to apply a denoiser as pre-
As the noise robustness of GANs has not been sufficiently studied, we first perform a comprehensive study on CIFAR-10 [43], where we compare various models in diverse noise settings (in which we test 152 conditions). Furthermore, inspired by the recent large-scale study on GANs [45], we also examine the performance on more complex datasets (LSUN BEDROOM [79] and FFHQ [37]). Finally, we demonstrate the applicability of NR-GANs in image denoising, where we learn a denoiser using generated noisy images and generated clean images (GN2GC), and empirically examine a chicken and egg problem between noise robust image generation and image denoising.

Overall, our contributions are summarized as follows:

- We provide **noise robust image generation**, the purpose of which is to learn a clean image generator even when training images are noisy. In particular, we solve this problem without full knowledge of the noise.
- To achieve this, we propose a novel family of GANs called **NR-GANs** that train a clean image generator and noise generator simultaneously with a distribution or transformation constraint on the noise generator.
- We provide a comprehensive study on CIFAR-10 (in which we test 152 conditions) and examine the versatility in more complex datasets (LSUN BEDROOM and FFHQ); finally, we demonstrate the applicability in image denoising. The project page is available at [https://takuhirok.github.io/NR-GAN/](https://takuhirok.github.io/NR-GAN/).

### 2. Related work

#### Deep generative models

Image generation is a fundamental problem and has been intensively studied in computer vision and machine learning. Recently, deep generative models have emerged as a promising framework. Among them, three prominent models along with GANs are variational...
autoencoders [40, 67], autoregressive models [74], and flow-based models [14, 15]. Each model has pros and cons. A well-known disadvantage of GANs is training instability; however, it has been steadily improved by recent advancements [2, 56, 47, 4, 69, 23, 41, 59, 60, 36, 81, 6, 9, 37]. In this work, we focus on GANs for their design flexibility, which allows them to incorporate the core of our models, a noise generator and its constraints. Also in other models, image fidelity has improved [75, 64, 58, 39]. Hence, sensitivity to noise can be problematic. Incorporating our ideas into them is a possible direction of future work.

**Image denoising.** Image denoising is also a fundamental problem and several methods have been proposed. They are roughly categorized into two: model-based methods [12, 22, 17, 53, 7, 52, 16, 50, 54] and discriminative learning methods [30, 57, 71, 82, 83, 8, 24, 46, 44, 3]. Recently, discriminative learning methods have shown a better performance; however, a limitation is that most of them (i.e., Noise2Clean (N2C)) require clean images for supervised training of a denoiser. To handle this, self-supervised learning methods (e.g., Noise2Void (N2V) [44] and Noise2Self (N2S) [3]) were proposed. These methods assume the same data setting as ours, i.e., only noisy images are available for training. However, they still have some limitations, e.g., they cannot handle pixel-wise correlated noise, such as shown in Figure 2(G)(H), and their performance is still inferior to supervised learning methods.

Image denoising and our noise robust image generation is a chicken and egg problem and each task can be used as a pre-task for learning the other. In the spirit of AmbientGAN, we aim to learn a clean image generator directly from noisy images. However, examining the performance on (1) learning a generator using denoised images and (2) learning a denoiser using generated clean and noisy images is an interesting research topic. Motivated by this, we empirically examined them through comparative studies. We provide the results in Sections 8.1 and 8.3.

**Noise robust models.** Except for image denoising, noise robust models have been studied in image classification to learn a classifier in practical settings. There are two studies addressing label noise [18, 85, 65, 55, 31, 72, 66, 25, 70, 32, 62, 20] and addressing image noise [87, 13]. For both tasks, the issue is the memorization effect [80], i.e., DNN classifiers can fit labels or images even though they are noisy or fully corrupted. As demonstrated in Figure 1, a similar issue also occurs in image generation.

Pertaining image generation, handling of label noise [35, 34, 33, 73] and image noise [5] has begun to be studied. Our NR-GANs are categorized into the latter. As discussed in Section 1, AmbientGAN [5] is a representative model in the latter category. However, a limitation is that it requires full knowledge of the noise. Therefore, we introduce NR-GANs to solve this problem as they do not have this limitation.

### 3. Notation and problem statement

We first define notation and the problem statement. Hereafter, we use superscripts $r$ and $g$ to denote the real distribution and generative distribution, respectively. Let $y$ be the observable noisy image and $x$ and $n$ be the underlying signal (i.e., clean image) and noise, respectively, where $y, x, n \in \mathbb{R}^{H \times W \times C}$ ($H$, $W$, and $C$ are the height, width, and channels of an image, respectively). In particular, we assume that $y$ can be decomposed additively: $y = x + n$.\(^1\) Our task is to learn a clean image generator that can reproduce clean images, such that $p^r(x) = p^r(x)$, when trained with noisy images $y^r \sim p^r(y)$. This is a challenge for standard GAN as it attempts to mimic the observable images including the noise; namely, it learns $p^r(y) = p^r(y)$.

We assume various types of noise. Figure 2 shows the categorization and examples of the noises that we address in this paper. They include signal-independent noises (A)–(H), signal-dependent noises (I)–(P), pixel-wise correlated noises (G)(H), local noises (C)(D), and their combination (H)(K)(L)(O)(P). We also consider two cases: the noise amount is either fixed or variable across the dataset.

As discussed in Section 1, one solution is AmbientGAN [5]; however, it is limited by the need for prior noise knowledge. We plan a solution that will not require that full prior knowledge. Our central idea is to introduce two generators, i.e., a clean image generator and noise generator, and impose a distribution or transformation constraint on the noise generator so that it captures only the noise-specific components. In particular, we explicate such constraints by relying on the signal-noise dependency. We first review our baseline AmbientGAN [5] (Section 4); then detail NR-GANs for signal-independent noise (Section 5) and signal-dependent noise (Section 6).

### 4. Baseline: AmbientGAN

AmbientGAN [5] (Figures 3(a) and 4(a)) is a variant of GANs, which learns an underlying distribution $p^r(x)$ only from noisy images $y^r \sim p^r(y)$.\(^2\) This is a challenging because the desired images $x^r \sim p^r(x)$ are not observable during training. To overcome this challenge, AmbientGAN introduces a noise simulation model $y = F_\theta(x)$ under the assumption that it is priory known. The main idea of AmbientGAN is to incorporate this noise simulation model into the adversarial training framework:

\[
\min_{G_x} \max_{D_y} \mathbb{E}_{y^r \sim p^r(y)} [\log D_y(y^r)] + \mathbb{E}_{z_x \sim p(z_x), \theta \sim p(\theta)} [\log(1 - D_y(F_\theta(G_x(z_x))))].
\]

\(^1\)We decompose additively; however, note that this representation includes signal-independent noise $n \sim p(n)$ and signal-dependent noise $n \sim p(n|x)$.

\(^2\)Strictly, AmbientGAN can handle more general *lossy* data, such as missing data. Here, we narrow the target in accordance with our task.
Assumption 1 (i) The noise is conditionally pixel-wise independent given the signal \( x \). (ii) The noise distribution type (e.g., Gaussian) is priorly known. Note that the noise amount needs not to be known. (iii) The signal \( x \) does not follow the defined noise distribution.

Under this assumption, we develop SI-NR-GAN-I (Figure 3(c)). In this model, we regularize the output distribution of \( G_n \) in a pixel-wise manner using a reparameterization trick [40]. Here, we present the case when the noise distribution type is defined as zero-mean Gaussian:

\[
y = x + n, \quad \text{where} \quad n \sim \mathcal{N}(0, \text{diag}(\sigma)^2),
\]

(3)

where \( \sigma \in \mathbb{R}^{H \times W \times C} \) is the pixel-wise standard deviation. In this case, we redefine the noise generator as \( \sigma = G_n(z_n) \); and introduce an auxiliary pixel-wise random variable \( \mathbf{e} \sim \mathcal{N}(0, I) \), where \( \mathbf{e} \in \mathbb{R}^{H \times W \times C} \); and then calculate the noise \( n \) by multiplying them: \( n = \sigma \cdot \mathbf{e} \), where \( \cdot \) represents an element-wise product. This formulation allows the noise to be sampled as \( n \sim \mathcal{N}(0, \text{diag}(\sigma)^2) \).

In SI-NR-GAN-I, \( \sigma \) is learned through training in a pixel-wise manner. Therefore, the same model can be applied to various noises (e.g., Figure 2(A)–(D), in which each pixel’s noise follows a Gaussian distribution, while the noise amount is different in a sample-wise (e.g., (B)) or pixel-wise (e.g., (D)) manner).

5.2. SI-NR-GAN-II

Two limitations of SI-NR-GAN-I are that it assumes that (i) the noise is pixel-wise independent and (ii) the noise distribution type is pre-defined. The first assumption makes it difficult to apply to a pixel-wise correlated noise (e.g., Figure 2(G)–(H)). The second assumption could cause difficulty when diverse noises are mixed (e.g., Figure 2(F)) or the noise distribution type is different from the pre-defined (e.g., Figure 2(E)). This motivates us to devise SI-NR-GAN-II, which works under a different assumption:

Assumption 2 (i) The noise \( n \) is rotation-, channel-shuffle-, or color-inverse-invariant. (ii) The signal \( x \) is rotation-, channel-shuffle-, or color-inverse-invariant.

3Strictly, our approach is applicable as long as a noise follows a differentiable distribution [40].
Among the noises in Figure 2, this assumption holds in all signal-independent noises (A)–(H). This assumption is reasonable when \( n \) is a zero-mean signal-independent noise and \( x \) is a natural image.\(^4\) Under this assumption, we establish SI-NR-GAN-II (Figure 3(d)). In this model, we redefine the noise generator as \( \hat{n} = G_n(z_n) \) (\( \hat{n} \in \mathbb{R}^{H \times W \times C} \)) and apply transformations to \( \hat{n} \) by \( n = T(\hat{n}) \), where \( T \) is a transformation function. As \( T \), we can use arbitrary transformation as long as it is applicable to \( n \) but not allowable to \( x \). In practice, we use three transformations: (i) rotation – rotating \( \hat{n} \) by \( d \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\} \) randomly, (ii) channel shuffle – shuffling RGB channels randomly, and (iii) color inversion – inverting colors randomly in a channel-wise manner. Each one utilizes one of the invariant and variant characteristics mentioned in Assumption 2. In SI-NR-GAN-II, the noise origin \( n \) is acquired in a data-driven manner; therefore, it is applicable to diverse noises (e.g., Figure 2(A)–(H)) without model modifications.

### 6. Signal-dependent noise robust GANs

Just like in the signal-independent noise case, AmbientGAN is applicable to signal-dependent noise by incorporating the pre-defined noise model (Figure 4(a)). However, it requires prior knowledge about the noise distribution type, signal-noise relationship, and noise amount. To deal with these requirements, we establish three variants that have different assumptions: SD-NR-GAN-I (Section 6.1), SD-NR-GAN-II (Section 6.2), and SD-NR-GAN-III (Section 6.3).

#### 6.1. SD-NR-GAN-I

We first consider the case when the following assumption holds in addition to Assumption 1.

**Assumption 3** The signal-noise relationship is priorly known. Note that the noise amount needs not be known.

Under this assumption, we devise SD-NR-GAN-I (Figure 4(b)), which incorporates a signal-noise relational procedure into SI-NR-GAN-I explicitly. In particular, we devise two configurations for two typical signal-dependent noises: multiplicative Gaussian noise (Figure 2(I)(J)) and Poisson noise (Figure 2(M)(N)).

Multiplicative Gaussian noise is defined as

\[
y = x + n, \quad \text{where } n \sim \mathcal{N}(0, \text{diag}(\sigma \cdot x)^2).
\]

To represent this noise with trainable \( \sigma \), we redesign the noise generator as \( \sigma = G_n(z_n) \). Then, we convert \( \sigma \) using a signal-noise relational function \( R(x, \sigma) = \sigma \cdot x = \hat{\sigma} \). Finally, we obtain \( n \sim \mathcal{N}(0, \text{diag}(\hat{\sigma}^2)) \) by using the reparameterization trick described in Section 5.1.

Poisson noise (or shot noise) is sampled by \( y \sim \text{Poisson}(\lambda \cdot x)/\lambda \), where \( \lambda \) is the total number of events. As this noise is discrete and intractable to construction of a differentiable model, we use a Gaussian approximation [26], which is commonly used for Poisson noise modeling:

\[
y = x + n, \quad \text{where } n \sim \mathcal{N}(0, \text{diag}(\sigma \cdot \sqrt{x})^2),
\]

where \( \sigma = \sqrt{1/\lambda} \). The implementation method is almost the same as that for the multiplicative Gaussian noise except that we redefine \( R(x, \sigma) \) as \( R(x, \sigma) = \sigma \cdot \sqrt{x} = \hat{\sigma} \).

In both noise cases, the noise amount \( \sigma \) is trainable; therefore, each configuration of SD-NR-GAN-I is applicable to the noises in Figure 2(I)(J) and those in Figure 2(M)(N), respectively, without model modifications.

#### 6.2. SD-NR-GAN-II

In SD-NR-GAN-II, we consider the case when the noise distribution type is known (i.e., Assumption 1 holds) but the signal-noise relationship is unknown (i.e., Assumption 3 is not required). Under this assumption, we aim to learn \( R(x, \sigma) \) implicitly, which is explicitly given in SD-NR-GAN-I. To achieve this, we develop SD-NR-GAN-II (Figure 4(c)), which is an extension of SI-NR-GAN-I incorporating the image latent vector \( z_n \) into the input of \( G_n \), i.e.,
\[ \sigma = G_n(z_n, z_x). \] Similarly to SI-NR-GAN-I, we sample \( n \sim \mathcal{N}(0, \text{diag}(\sigma)^2) \) using the reparameterization trick described in Section 5.1. Here, we consider the case when the noise distribution type is defined as zero-mean Gaussian.

As discussed in Section 6.1, multiplicative Gaussian noise and Poisson noise are represented (or approximated) as signal-dependent Gaussian noise; therefore, SD-NR-GAN-II is applicable to these noises (e.g., Figure 2(I)(J)(M)(N)). Furthermore, SD-NR-GAN-II can internally learn \( R(x, \sigma) \); therefore, the same model can also be applied to signal-independent noise (Figure 2(A)–(D)), i.e., \( R(x, \sigma) = x \), and the combination of multiple noises (Figure 2(K)(L)(O)(P)), e.g., \( R(x, \sigma_d, \sigma_i) = \sigma_d \cdot x + \sigma_i \).

### 6.3. SD-NR-GAN-III

Finally, we deal with the case when both the noise distribution type and signal-noise relationship are not known. In this case, we impose a similar assumption as Assumption 2. However, rotation and channel shuffle collapse the per-pixel signal-noise dependency that is included in typical signal-dependent noise (e.g., Figure 2(I)–(P)). Therefore, we only induce the assumption regarding color inversion. Under this assumption, we devise SD-NR-GAN-III (Figure 4(d)). Similarly to SD-NR-GAN-II, SD-NR-GAN-III learns the signal-noise relationship implicitly by incorporating \( z_x \) into the input of \( G_n \), i.e., \( \hat{n} = G_n(z_n, z_x) \). Similarly to SI-NR-GAN-II, we impose a transformation constraint on \( G_n \) by applying \( n = T(\hat{n}) \), where \( T \) is defined as color inversion. The noise origin \( \hat{n} \) is learned through training; therefore, SD-NR-GAN-III can be adopted to various noises (e.g., all noises in Figure 2) without modifying the model.

### 7. Advanced techniques for practice

#### 7.1. Alleviation of convergence speed difference

In proposed NR-GANs, \( G_x \) and \( G_n \) are learned simultaneously. Ideally, we expect that \( G_x \) and \( G_n \) would be optimized at the same speed; however, through experiments, we found that \( G_n \) tends to learn faster than \( G_x \) and results in a mode collapse in the early training phase. A possible cause is that the noise distribution is simpler and easier to learn than the image distribution. To address this problem, we apply the diversity-sensitive regularization \[ [78] \] to \( G_n \). Intuitively, this regularization makes \( G_n \) sensitive to \( z_n \) and has an effect to prevent the mode collapse. In the experiments, we incorporate this technique to all NR-GANs. We discuss the effect of this regularization in Appendix A.1.

#### 7.2. Alleviation of approximation degradation

As described in Section 6.1, we apply a Gaussian approximation to the Poisson noise to make it tractable and differentiable. However, through experiments, we found that this approximation causes the performance degradation even using AmbientGAN, which knows all information about the noise. A possible reason is that powerful \( G_x \) attempts to fill in the discretized gap caused by this approximation. To alleviate the effect, we apply an anti-alias (or low-pass) filter \([84]\) to \( x \) before providing to \( D_y \). In particular, we found that applying vertical and horizontal blur filters respectively and providing both to \( D_y \) works well. In the experiments, we apply this technique to all GANs in the Poisson or Poisson-Gaussian noise setting. We discuss the effect with and without this technique in Appendix A.2.

#### 8. Experiments

##### 8.1. Comprehensive study

To advance the research on noise robust image generation, we first conducted a comprehensive study, where we compared various models in diverse noise settings (in which we tested 152 conditions in total).

**Data setting.** In this comprehensive study, we used CIFAR-10 \([43]\), which contains 60k \( 32 \times 32 \) natural images, partitioned into 50k training and 10k test images. We selected this dataset because it is commonly used to examine the benchmark performance of generative models (also in the study of AmbientGAN \([5]\)); additionally, the image size is reasonable for a large-scale comparative study. Note that we also conducted experiments using more complex datasets in Section 8.2. With regard to noise, we tested 16 noises, shown in Figure 2. See the caption for their details.

**Compared models.** In addition to the models in Figures 3 and 4, we tested several baselines. As comparative GAN models, we examined four models: (1) Standard GAN, (2) P-AmbientGAN (parametric AmbientGAN), a straightforward extension of AmbientGAN, which has a single trainable parameter \( \sigma \). As with SI-NR-GAN-I and SD-NR-GAN-I, we construct this model for Gaussian, multiplicative Gaussian, and Poisson noises and generate the noise with \( \sigma \) using a reparameterization trick \([40]\). (3) SI-NR-GAN-0 (Figure 3(b)), which has the same generators as SI-NR-GANs but has no constraint on \( G_n \). (4) SD-NR-GAN-0, which has the same generators as SD-NR-GAN-II and -III but has no constraint on \( G_n \).

We also examined the performance of learning GANs using denoised images (denoiser+GANs). As a denoiser, we investigated four methods. As typical model-based methods, we used (1) GAT-BM3D \([54]\) and (2) CBM3D \([11]\) for Poisson/Poisson-Gaussian noise (Figure 2(M)–(P)) and the other noises, respectively. As discriminative learning

---

For further clarification, we conducted a comparative study on transformations in SD-NR-GAN-III. See Appendix A.8 for details.
methods, we used (3) N2V (Noise2Void) [44] and (4) N2N (Noise2Noise) [46]. N2V can be used in the same data setting as ours (i.e., only noisy images are available for training), while N2N requires noisy image pairs for training. We used N2N because it is commonly used as the upper bound of self-supervised learning methods (e.g., N2V).

**Evaluation metrics.** We used the Fréchet distance (FID) [28] as an evaluation metric because its validity has been demonstrated in large-scale studies on GANs [48, 45], and because the sensitivity to the noise has also been shown [28]. The FID measures the distance between real and generative distributions and a smaller value is better.

**Implementation.** We implemented GANs using the ResNet architectures [27] and trained them using a non-saturating GAN loss [21] with a real gradient penalty regularization [59]. In NR-GANs, we used similar architectures for $G_n$ and $G_m$. As our aim is to construct a general model applicable to various noises, we examined the performance when the training settings are fixed regardless of the noise. We provide the implementation details in Appendix C.1.

**Results on signal-independent noises.** The upper part of Table 1 summarizes the results on signal-independent noises. In P-AmbientGAN and SI-NR-GANs, we defined the distribution type as Gaussian for all noise settings and analyzed the effect when the noise is beyond assumption. Our main findings are the following:

1. **Comparison among GAN models.** As expected, AmbientGAN tends to achieve the best score owing to the advantageous training setting, while the best SI-NR-GAN shows a competitive performance (with a difference of 3.3 in the worst case). P-AmbientGAN is defeated by SI-NR-GAN-I in all cases. These results indicate that our two-generator model is reasonable when training a noise generator and image generator simultaneously.

2. **Comparison between SI-NR-GANs and denoiser+GANs.** The best SI-NR-GAN outperforms the best denoiser+GAN in most cases (except for (G)). In particular, pixel-wise correlated noises (G)(H) are intractable for denoiser+GANs except for N2N+GAN, which uses additional supervision, while SI-NR-GAN-II works well and outperforms the baseline models by a large margin (with a difference of over 100).

3. **Comparison among SI-NR-GANs.** SI-NR-GAN-II shows the stable performance across all cases (the difference to the best SI-NR-GAN is within 3.1). SI-NR-GAN-I shows the best or competitive performance in Gaussian (A)–(D) or near Gaussian noise (F); however, the performance degrades when the distribution is beyond assumption (E)(G)(H).

**Results on signal-dependent noises.** The lower part of Table 1 lists the results on signal-dependent noises. In P-AmbientGAN and SD-NR-GAN-I, we defined the distribution type as multiplicative Gaussian and Poisson in (I)–(L) and (M)–(P), respectively. With regard to a comparison among GAN models and comparison between SD-NR-GANs and denoiser+GANs, similar findings (i.e., the best SD-NR-GAN is comparable with AmbientGAN and outperforms the best denoiser+GAN) are observed; therefore, herein we discuss a comparison among SD-NR-GANs. SD-NR-GAN-II and -III stability work better than SD-NR-GAN-0. Among the two, SD-NR-GAN-II, which has a stronger assumption, outperforms SD-NR-GAN-III in all cases (with a difference of over 5.4). SD-NR-GAN-I shows the best or competitive performance when noises are within or a little over assumption (I)–(K)(M)–(O); however, when the unexpected noise increases (L)(P), the performance degrades.

**Summary.** Through the comprehensive study, we confirm the following: (1) NR-GANs work reasonably well comparing to other GAN models and denoiser+GANs. (2) Weakly constrained NR-GANs stability work well across various settings, while (3) strongly constrained NR-GANs show a better performance when noise is within assumption.7

---

7As further analyses, we examined the effect of diversity-sensitive regularization (Appendix A.1); the effect of blur filtering (Appendix A.2); performance on the clean dataset (Appendix A.3); performance on the semi-noisy datasets (Appendix A.4); performance on the mixed datasets

### Table 1. Comparison of FID on CIFAR-10.

| Signal-independent            | (A) | (B) | (C) | (D) | (E) | (F) | (G) | (H) |
|------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| GAN                          |     |     |     |     |     |     |     |     |
| P-AmbientGAN                 | 27.1| 68.7| 39.7| 137.7| 43.8| 100.7| 43.0| 94.2|
| CBM3D+GAN                    | 82.7| 77.4| 93.2| 155.8| 152.4| 160.1| 149.1| 175.8|
| SD-NR-GAN-0                  | 82.7| 59.5| 69.9| 75.1| 71.7| 70.2| 72.0| 69.0|
| SD-NR-GAN-I                  | 22.5| 23.0| 25.3| 112.4| 30.8| 32.0| 31.4| 70.6|
| SD-NR-GAN-II                 | 24.4| 24.2| 23.3| 28.5| 34.0| 33.9| 34.0| 35.4|
| SD-NR-GAN-III                | 37.5| 33.4| 33.5| 33.9| 53.1| 55.1| 52.2| 47.2|
| CBM3D+GAN                    |     |     |     |     |     |     |     |     |
| GAT-BM3D+GAN                 | 26.9| 27.8| 27.6| 40.0|     |     |     |     |
| N2V+GAN                      | 25.8| 26.6| 26.7| 36.4| 37.1| 38.3| 37.5| 41.2|
| N2N+GAN†                     | 24.9| 26.6| 26.2| 34.1| 36.7| 39.7| 36.4| 39.5|

| Signal-dependent (I) (J) (K) (L) (M) (N) (O) (P) |
|-----------------------------------------------|
| AmbientGAN†                                  | 21.4| 21.8| 21.9| 27.4| 31.3| 32.3| 30.9| 35.3|
| P-AmbientGAN                                 | 27.1| 68.7| 39.7| 137.7| 43.8| 100.7| 43.0| 94.2|
| CBM3D+GAN                                    | 82.7| 77.4| 93.2| 155.8| 152.4| 160.1| 149.1| 175.8|
| SD-NR-GAN-0                                  | 82.7| 59.5| 69.9| 75.1| 71.7| 70.2| 72.0| 69.0|
| SD-NR-GAN-I                                  | 22.5| 23.0| 25.3| 112.4| 30.8| 32.0| 31.4| 70.6|
| SD-NR-GAN-II                                 | 24.4| 24.2| 23.3| 28.5| 34.0| 33.9| 34.0| 35.4|
| SD-NR-GAN-III                                | 37.5| 33.4| 33.5| 33.9| 53.1| 55.1| 52.2| 47.2|
| CBM3D+GAN                                    |     |     |     |     |     |     |     |     |
| GAT-BM3D+GAN                                 | 26.9| 27.8| 27.6| 40.0|     |     |     |     |
| N2V+GAN                                      | 25.8| 26.6| 26.7| 36.4| 37.1| 38.3| 37.5| 41.2|
| N2N+GAN‡                                     | 24.9| 26.6| 26.2| 34.1| 36.7| 39.7| 36.4| 39.5|

As further analyses, we examined the effect of diversity-sensitive regularization (Appendix A.1); the effect of blur filtering (Appendix A.2); performance on the clean dataset (Appendix A.3); performance on the semi-noisy datasets (Appendix A.4); performance on the mixed datasets
and more challenging dataset that includes vastly more variation. The reason why we used FFHQ is that FFHQ is the latest
show examples of generated images in Appendix B.1. See Appendices for
generality analysis of SD-NR-GAN-II and -III (Appendix A.10). We also
explicit relational learning in SD-NR-GAN-II and -III (Appendix A.9); and a
tions in SI-NR-GAN-II (Appendix A.7); a comparative study on trans-
appendix A.6). Moreover, we conducted an ablation study on transforma-
Table 2. Comparison of FID on LSUN BEDROOM and FFHQ.
A smaller value is better. Because the training is time-consuming,
experiments were run once. The notation is the same as that in
Table 1.

8.2. Evaluation on complex datasets

Inspired by the resent large-scale study on GANs [45], we also examined
the performance on more complex datasets. Referring to this study, we used the
128 × 128 versions of LSUN BEDROOM [79] and FFHQ [37]. LSUN
BEDROOM contains about 3 million bedroom images, randomly split into training and test sets in the ratio of 99 to
1. FFHQ contains 70k face images, partitioned into 60k training and 10k test images. As these datasets are
calculationally demanding, we selected six noises for LSUN
BEDROOM and two noises for FFHQ. We provide the implemen-
tation details in Appendix C.2.

Table 2 lists the results. Just like the CIFAR-10 results, we found that the best NR-GAN outperforms standard GAN
and its performance is closer to that of AmbientGAN. In contrast, differently from the CIFAR-10 results, we found that in complex datasets, some weakly constrained SD-NR-GANs suffer from learning difficulty (e.g., SD-NR-GAN-
III in LSUN BEDROOM (M)). This is undesirable but understandable because in complex datasets it is highly challenging to isolate noise from the dependent signal without an explicit knowledge about their dependency. This is related
to GAN training dynamics and addressing this limitation is our future work. As reference, we provide qualitative results in Appendix B.2.

Table 3. Comparison of PSNR on LSUN BEDROOM and FFHQ. A larger value is better. We report the median score across three random seeds. The symbols ‡ indicate that the
models are trained in advantageous conditions (∗clean target images and †noisy image pairs are given, respectively). The other models are trained using only noisy images (not including pairs). Bold font indicates the best score except for the models denoted by ‡.

8.3. Application to image denoising

NR-GANs can generate an image and noise, respectively. By utilizing this, we create clean and noisy image pairs synthetically and use them for learning a denoiser. We call this method GeneratedNoise2GeneratedClean (GN2GC). In particular, we employed the generators that achieve the best FID in Table 2 (denoted by bold font). Note that NR-GANs are trained only using noisy images; therefore, GN2GC can be used in the same data setting as self-supervised learning methods (N2V [44] and N2S [3]). We used the same training and test sets used in Section 8.2. We present the implementation details in Appendix C.3.

We summarize the results in Table 3. We found that
GN2GC not only outperforms the state-of-the-art self-supervised learning methods (N2V and N2S) but also is comparable with N2N, which learns in advantageous conditions. The requirement for pre-training GANs could narrow the applications of GN2GC; however, we believe that its potential for image denoising would increase along with rapid progress of GANs. We show examples of denoised images in Appendix B.3.

9We provide other case results in Appendix A.11.
Acknowledgement

We thank Naoya Fushishita, Takayuki Hara, and Atsuhiro Noguchi for helpful discussions. This work was partially supported by JST CREST Grant Number JP-MJCR1403, and partially supported by JSPS KakenHI Grant Number JP19H01115.

References

[1] Josue Anaya and Adrian Barbu. RENOIR–A dataset for real low-light image noise reduction. *J. Vis. Commun. Image Represent.*, 51:144–154, 2018.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

[3] Joshua Batson and Loic Royer. Noise2Self: Blind denoising by self-supervision. In *ICML*, 2019.

[4] Marc G. Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The Cramer distance as a solution to biased Wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017.

[5] Ashish Bora, Eric Price, and Alexandros G. Dimakis. AmbientGAN: Generative models from lossy measurements. In *ICLR*, 2018.

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[7] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *CVPR*, 2005.

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

[9] Ting Chen, Mario Lucic, Neil Houlsby, and Sylvain Gelly. On self modulation for generative adversarial networks. In *ICLR*, 2019.

[10] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised GANs via auxiliary rotation loss. In *CVPR*, 2019.

[11] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space. In *ICIP*, 2007.

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

[13] Steven Diamond, Vincent Sitzmann, Stephen Boyd, Gordon Wetzstein, and Felix Heide. Dirty pixels: Optimizing image classification architectures for raw sensor data. *arXiv preprint arXiv:1701.06487*, 2017.

[14] Laurent Dinnh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. In *ICLR Workshop*, 2015.

[15] Laurent Dinnh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *ICLR*, 2017.

[16] Weisheng Dong, Lei Zhang, Guangming Shi, and Xin Li. Nonlocally centralized sparse representation for image restoration. *IEEE Trans. Image Process.*, 22(4):1620–1630, 2012.

[17] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Process.*, 15(12):3736–3745, 2006.

[18] Aritra Ghosh, Himanshu Kumar, and P.S. Sastry. Robust loss functions under label noise for deep neural networks. In *AAAI*, 2017.

[19] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

[20] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural networks using a noise adaptation layer. In *ICLR*, 2017.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[22] Shuhang Gu, Lei Zhang, Wangmeng Zu, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *CVPR*, 2014.

[23] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In *NIPS*, 2017.

[24] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zu, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *CVPR*, 2019.

[25] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, 2018.

[26] Samuel W. Hasinoff. Photon, Poisson noise. *Computer Vision: A Reference Guide*, pages 608–610, 2014.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[28] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a Nash equilibrium. In *NIPS*, 2017.

[29] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[30] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In *NIPS*, 2009.

[31] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

[32] Ishan Jindal, Matthew Nokleby, and Xuewen Chen. Learning deep networks from noisy labels with dropout regularization. In *ICDM*, 2016.

[33] Takuhito Kaneko and Tatsuya Harada. Label-noise robust multi-domain image-to-image translation. *arXiv preprint arXiv:1905.02185*, 2019.
[73] Kiran Koshy Thekumparampil, Ashish Khetan, Zinan Lin, and Sewoong Oh. Robustness of conditional GANs to noisy labels. In NeurIPS, 2018. 3
[74] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In ICML, 2016. 3
[75] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In NIPS, 2017. 3
[76] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. In ICML Workshop, 2015. 27
[77] Bing Xu, Junfei Zhang, Rui Wang, Kun Xu, Yong-Liang Yang, Chuan Li, and Rui Tang. Adversarial Monte Carlo denoising with conditioned auxiliary feature modulation. ACM Trans. Graph., 38(6):1–12, 2019. 1
[78] Bing Xu, Junseok Hong, Yunseok Jang, Tianchen Zhao, and Honglak Lee. Diversity-sensitive conditional generative adversarial networks. In ICLR, 2019. 6, 12, 27, 29
[79] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365, 2015. 2, 8
[80] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In ICLR, 2017. 3
[81] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In ICML, 2019. 1, 3
[82] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. IEEE Trans. Image Process., 26(7):3142–3155, 2017. 3
[83] Kai Zhang, Wangmeng Zuo, and Lei Zhang. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. IEEE Trans. Image Process., 27(9):4608–4622, 2018. 3
[84] Richard Zhang. Making convolutional networks shift-invariant again. In ICML, 2019. 6
[85] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In NeurIPS, 2018. 3
[86] Qian Zhao, Deyu Meng, Zongben Xu, Wangmeng Zuo, and Lei Zhang. Robust principal component analysis with complex noise. In ICML, 2014. 2
[87] Stephen Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In CVPR, 2016. 3
[88] Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. Comput. Graph. Forum, 34(2):667–681, 2015. 1
A. Further analyses

In this section, we provide further analyses mentioned in the main text.

A.1. Effect of diversity-sensitive regularization

We first discuss the impact of the diversity-sensitive regularization (DS regularization) [78] introduced in Section 7.1. As described, we used this regularization to prevent $G_n$ from resulting in a mode collapse. In this section, we first detail DS regularization, followed by the experimental results.

DS regularization is defined as follows:

$$\max_{G_n} \lambda \mathbb{E}_{z_n^1, z_n^2 \sim p(z_n)} \left[ \min \left( \frac{\|G_n(z_n^1) - G_n(z_n^2)\|}{\|z_n^1 - z_n^2\|}, \tau \right) \right],$$

(6)

where $\lambda$ controls the importance of DS regularization, and $\tau$ is a boundary for ensuring numerical stability. Intuitively, when a mode collapse occurs and $G_n$ produces a deterministic output, Equation 6 reaches close to its minimum because $G_n(z_n^1) \approx G_n(z_n^2)$ for all $z_n^1, z_n^2 \sim p(z_n)$. To avoid this, we regularize $G_n$ to maximize Equation 6 and promote $G_n$ to produce diverse outputs depending on $z_n$.

During the experiments described in Section 8.1, we set $\tau$ to zero, following the implementation of a study on DS regularization [78]. Furthermore, we fixed $\lambda = 0.02$ and applied DS regularization to all NR-GANs to investigate the performance when the training parameters were set.

To further analyze the effect of DS regularization, herein we examined the performance when $\lambda$ is changed. We show qualitative results in Figure 5. As shown, the diversity of the noise increases as $\lambda$ increases (in particular, in Figure 5(c)).

We also analyzed quantitatively. In particular, we examined both a case in which the images are noisy and a case in which the images are clean. In the former case, DS regularization will be useful because the diversity of the noise is significant, while in the latter case, DS regularization will be superfluous because there is no diversity in the noise.

Figure 6 shows a comparison of FID using a different strength of DS regularization. We found that, for the noisy images (Figure 2(A)(B)(I)(M)), all models improve the FID when using the appropriate strength of DS regularization (in particular, $\lambda \in [0.01, 0.02]$). We also confirm that when using $\lambda \in [0.01, 0.02]$, the negative effect is relatively small (the difference is within 1.0) in the clean dataset, where the diversity in the noise is not required. Based on these results, we argue that the incorporation of appropriate DS regularization is reasonable for the NR-GAN framework. As mentioned in the original study on DS regularization [78], learning $\lambda$ from the training data remains an interesting area for future research.
A.2. Effect of blur filtering

As described in Section 7.2, we apply blur (i.e., low-pass) filters to Poisson noise to alleviate the difficulty in learning a discrete distribution. In particular, we apply both vertical and horizontal blur filters to an image \( y \); concatenate the two filtered results; and provide the concatenation to \( D_y \), as shown in Figure 7(c). We call this approach \textit{Blurvh}. By focusing the filter in one direction, we can alleviate the discretized effect in that direction while preserving the original structure in another direction.

To clarify the effect of \textit{Blurvh}, we compared it with a non-filtered model (No filter; Figure 7(a)) and a standard blur filtering model (Blur; Figure 7(b)). Table 4(a) lists the results for Poisson noise (Figure 2(M)(N)) and Poisson-Gaussian noise (Figure 2(O)(P)). We observed that, under Poisson noise (M)(N) and Poisson-noise dominated Poisson-Gaussian noise (O), the proposed \textit{Blurvh} outperforms \textit{No filter} for all GAN models including AmbientGAN (the difference is more than 3.3 in AmbientGAN and more than 5.3 in SD-NR-GANs). By contrast, standard \textit{Blur} is outperformed by \textit{No filter} in most cases (i.e., 10/12, except for (M)(O) in SD-NR-GAN-III). These results verify the superiority of the proposed \textit{Blurvh}. We also found that under Gaussian-noise dominated Poisson-Gaussian noise (P), \textit{No filter} works well in certain cases (i.e., 2/4, in AmbientGAN and SD-NR-GAN-III). We consider this to be due to the additive Gaussian noise helping alleviating the discretization.

To further understand \textit{Blurvh}, we also examined the performance under non-Poisson noise (namely, a continuously distributed noise). In particular, we conducted a test under multiplicative Gaussian noise (Figure 2(I)(J)) and under additive and multiplicative Gaussian noise (Figure 2(K)(L)). Table 4(b) shows the results. Unsurprisingly, we found that \textit{No filter} performs the best in most cases (i.e., 15/16, except for (L) in SD-NR-GAN-I, where the noise is beyond the assumption of SD-NR-GAN-I). This is because, under a continuously distributed noise, blur filtering is superfluous and results in a loss of the original structure. A remarkable finding here is that the negative effect of \textit{Blurvh} is constantly small compared to that of \textit{Blur}, and the degradation from \textit{No filter} is relatively small (with an average difference of 2.5 when excluding (L) in SD-NR-GAN-I, where SD-NR-GAN-I fails to learn a clean image generator because the noise is beyond assumption). Based on these results, we consider that applying \textit{Blurvh} is a reasonable option when the noise distribution is unknown.

A.3. Performance on clean dataset

In the main text, we provide the results on noisy datasets. An interesting question is the performance on a clean dataset. Ideally, it is expected that \( G_{\alpha} \) learns \textit{no noise} and outputs a value of zero in this case. To examine this, we conducted experiments on the original (clean) CIFAR-10.

Table 5 lists the results. We found that the performance is almost the same (with a difference of within 1.3) except for SD-NR-GAN-0. To examine the reason for the degra-
Table 5. **Comparison of FID on clean CIFAR-10.** A smaller value is better. We omit SD-NR-GAN-I because it is equal to SI-NR-GAN-I under the assumption that it knows the signal-noise relationship (i.e., no relationship here). We give the median score of three random seeds. Italic font indicates the worst score. Bold font indicates the best score except for the models denoted by †‡.

| Model          | Clean | (A) AGF | (A) AGF | (A) AGF | (G) BG | (G) BG | (G) BG |
|----------------|-------|---------|---------|---------|-------|-------|-------|
| GAN            | 18.8  | 0.25    | 0.5     | 0.75    | 1     | 0.25  | 0.5   |
| P-AmbientGAN   | 18.3  |         |         |         |       |       |       |
| SI-NR-GAN-0    | 19.1  | 20.7    | 22.2    | 24.1    | 26.7  | 21.4  | 23.3  |
| SI-NR-GAN-I    | 19.5  | 30.5    | 55.9    | 90.3    | 145.8 | 35.9  | 63.7  |
| SI-NR-GAN-II   | 18.7  | 21.5    | 23.9    | 25.9    | 26.7  | 36.3  | 65.2  |
| SI-NR-GAN-III  | 19.3  | 20.4    | 23.5    | 27.2    | 29.7  | 20.8  | 23.2  |
| SD-NR-GAN-0    | 57.0  | 21.5    | 24.9    | 29.2    | 34.6  | 39.0  | 65.8  |
| SD-NR-GAN-II   | 19.6  | 22.3    | 25.0    | 29.0    | 33.5  | 21.0  | 23.6  |
| SD-NR-GAN-III  |       |         |         |         |       |       |       |

Table 6. **Comparison of FID on semi-noisy datasets.** A smaller value is better. The third row denotes the noise rate r, i.e., what percentage of images include noise. We report the median score across three random seeds. The symbol † indicates that the ground-truth noise models are given. The symbol ‡ denotes that noisy image pairs are given during the training. The other models are trained using only noisy images (not including pairs) without complete noise information. Bold font indicates the best score except for the models denoted by †‡.

| Model          | (A) MGF | (B) MGF | (C) MGF | (D) MGF | (A) A+I | (B) A+I | (C) A+I | (D) A+I |
|----------------|---------|---------|---------|---------|---------|---------|---------|---------|
| AmbientGAN†    | 19.7    | 20.8    | 21.2    | 21.4    | 21.3    | 23.6    | 25.0    | 27.4    |
| GAN            | 23.4    | 36.0    | 56.4    | 82.7    | 32.6    | 58.2    | 95.5    | 155.8   |
| SI-NR-GAN-I    | 20.4    | 20.6    | 21.8    | 22.5    | 33.4    | 60.4    | 93.9    | 112.4   |
| SI-NR-GAN-II   | 21.0    | 22.6    | 23.8    | 24.4    | 20.9    | 23.4    | 27.3    | 28.5    |
| SI-NR-GAN-III  | 21.4    | 23.0    | 27.4    | 37.5    | 22.4    | 25.6    | 30.7    | 33.9    |
| N2V+GAN        | 20.5    | 21.8    | 23.1    | 25.8    | 21.6    | 25.7    | 29.9    | 36.4    |
| N2N+GAN†‡      | 20.3    | 21.6    | 23.4    | 24.9    | 21.5    | 25.6    | 30.5    | 34.1    |

Figure 8. **Examples of images generated for clean CIFAR-10.** We show 5 × 5 samples for each case. (a) In SD-NR-GAN-0, which does not regularize $G_n$, both $G_x$ and $G_n$ attempt to generate an image. (b) By contrast, in SD-NR-GAN-II, which imposes a distribution constraint on $G_n$, $G_x$ generates an image and $G_n$ outputs a value of zero (i.e., learns no noise).

Table 7. **Performance on semi-noisy datasets.**

| Model          | (A) AGF | (A) AGF | (A) AGF | (G) BG | (G) BG | (G) BG |
|----------------|---------|---------|---------|-------|-------|-------|
| AmbientGAN†    | 20.7    | 22.2    | 24.1    | 26.7  | 21.4  | 23.3  |
| GAN            | 30.5    | 55.9    | 90.3    | 145.8 | 35.9  | 63.7  |
| SI-NR-GAN-I    | 21.5    | 23.9    | 25.9    | 26.7  | 36.3  | 65.2  |
| SI-NR-GAN-II   | 20.4    | 23.5    | 27.2    | 29.8  | 20.8  | 23.2  |
| N2V+GAN        | 21.4    | 24.9    | 29.2    | 34.6  | 39.0  | 65.8  |
| N2N+GAN†‡      | 22.3    | 25.0    | 29.0    | 33.5  | 21.0  | 23.6  |

A.4. **Performance on semi-noisy datasets**

In the main text, we conducted experiments on datasets in which all images are noisy. Besides, a practically important question is the performance on a semi-noisy dataset in which parts of images are noisy while the remaining are clean. To answer this question, we conducted experiments on the semi-noisy datasets. In particular, we tested two signal-independent noises, additive Gaussian noise (Figure 2(A)) and Brown Gaussian noise (Figure 2(G)), and two signal-dependent noises, multiplicative Gaussian noise (Figure 2(I)) and additive and multiplicative Gaussian noise (Figure 2(L)). We selected the noise rate r from $r \in \{0.25, 0.5, 0.75, 1\}$.

Table 6 summarizes the results. We found a similar tendency as that described in Section 8.1. (1) The best NR-GAN is comparable with AmbientGAN. Note that in this case, AmbientGAN requires additional knowledge regarding the rate of noisy samples, whereas NR-GANs can obtain this knowledge in a data-driven manner. (2) The best NR-GAN shows better or competitive performance compared to the best denoiser+GAN in most cases (except for (G) with $r = 0.75$ and $r = 1$, where N2N+GAN, trained under advantageous conditions, performs the best). (3) Weakly constrained NR-GANs (i.e., SI-NR-GAN-II, SD-NR-GAN-II, and SD-NR-GAN-III) stability work well across diverse settings, while strongly constrained NR-GANs (i.e., SI-NR-GAN-I and SD-NR-GAN-I) show a better performance when noise is within assumption (i.e., (A) and (I), respectively) in most cases (except for (A) with $r = 0.25$ and $r = 0.5$, where SI-NR-GAN-II works slightly better than SI-NR-GAN-I). These results validate the usefulness of NR-GANs on semi-noisy datasets.

A.5. **Performance on mixed datasets**

In the main text, we conducted experiments on the datasets in which we assume that the noise parameters, excluding the amount of noise, are fixed across all training images. An interesting question is the performance on the mixed datasets, i.e., partial images containing a certain type of noise and the remaining images including a different type of noise. To investigate this, we conducted experiments on the mixed datasets. In particular, we tested
two cases: (1) **AGF/U**, where partial images contain additive Gaussian noise (Figure 2(A)) while the remaining include uniform noise (Figure 2(E)), and (2) **AGF/BG**, in which partial images contain additive Gaussian noise (Figure 2(A)) while the remaining include Brown Gaussian noise (Figure 2(G)). We selected the mixture rate $\mu$ from $\mu \in \{0, 0.25, 0.5, 0.75, 1\}$.

Table 7 summarizes the results. With respect to a comparison among GAN models and comparison between SI-NR-GANs and denoiser+GANs, we observed a similar tendency as that described in Section 8.1, i.e., the best SI-NR-GAN is comparable with AmbientGAN and outperforms the best denoiser+GAN in most cases (except for (A)/(G) with $\mu = 0.75$ and (G), where N2N+GAN, trained under advantageous conditions, performs the best). Hence, herein we discuss a comparison among SI-NR-GANs in greater detail. We found that in both mixed datasets, SI-NR-GAN-II, which can be applied to all noises, shows a stable performance (with a difference of within 3.5). By contrast, SI-NR-GAN-I, which assumes Gaussian noise, degrades the performance as the rate of unexpected noise (i.e., uniform or Brown Gaussian noise) increases. In particular, we observed that the mixing of Brown Gaussian noise has a larger impact. The reason for this is that Brown Gaussian noise has a correlation between pixels, which is more difficult to approximate by a Gaussian distribution, compared to uniform noise which is pixel-wise independent. These results indicate the importance of a model selection based on the noise type that may be contained.

**A.6. Performance of end-to-end denoiser+GAN**

In the main text, we used a pretrained denoiser when implementing N2V+GAN and N2N+GAN because it performs better than the end-to-end formulation.

Specifically, we considered two types of end-to-end denoiser+GAN schemes (Figure 9). Here, for ease of explanation, we focus on the case when N2V [44] is used as a denoiser. In the main text, we used the same network architectures presented in Figure 9(a), but adopted a pre-trained denoiser (N2V) because there is no merit in training a denoiser from scratch jointly with GAN because the denoiser is optimized using only real noisy images. In this case, we can obtain better denoised images from the start of GAN training by adopting a pre-trained denoiser. As empirical evidence, we show the comparison of FID on CIFAR-10 with additive Gaussian noise (Figure 2(A)) in Table 8.

In our preliminary experiments, we also tested the scheme presented in Figure 9(b). However, we found that it yields poor performance, as listed in Table 8. We consider that this is because in this scheme, the denoiser (N2V) is optimized using generated noisy images that do not necessarily satisfy the assumption of N2V (i.e., noise is pixel-wise independent) until the generator is able to recreate a noisy image completely.
Table 9. Comparison of FID on CIFAR-10 with signal-independent noises using different transformations. A smaller value is better. SI-NR-GAN-0 with all transformations (the last row) is used as SI-NR-GAN-III in other experiments. The median score across three random seeds is given. Bold font indicates the best score for each noise.

| Model                  | (A) | (B) | (C) | (D) | (E) | (F) | (G) | (H) | (I) | (J) | (K) | (L) | (M) | (N) | (O) | (P) |
|------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| AGF AGV LGF LGV U Mix |     | AGF | AGV | LGF | LGV | U   | Mix |     |     |     |     |     |     |     |     |     |
| SI-NR-GAN-0            | 40.7| 39.5| 23.1| 24.3| 38.6| 38.6| 32.7| 71.6| 139.7|
| + Rotation             | 28.3| 29.8| 24.0| 23.9| 32.2| 32.7| 27.6| 36.1| 47.7 |
| + Shuffle              | 30.2| 30.4| 24.1| 24.1| 32.2| 32.7| 27.2| 36.1| 47.1 |
| + Inversion            | 29.5| 30.3| 23.2| 23.0| 32.1| 26.8| 32.7| 33.6| 44.3 |
| + Rotation & shuffle   | 29.6| 30.2| 23.1| 23.3| 32.1| 26.1| 36.2| 46.1 |     |
| + Rotation & inversion | 29.8| 30.1| 22.1| 21.9| 32.3| 26.8| 33.4| 44.0 |     |
| + Shuffle & inversion  | 30.5| 31.0| 22.2| 22.6| 32.6| 26.7| 33.0| 44.4 |     |
| + All (SI-NR-GAN-III)  | 29.8| 29.7| 21.1| 21.7| 31.6| 26.5| 32.2| 44.0 |     |

Table 10. Comparison of FID on CIFAR-10 with signal-dependent noises using different transformations. A smaller value is better. SD-NR-GAN-0 with color inversion (the sixth row) is used as SD-NR-GAN-III in other experiments. We report the median score for three random seeds. Bold font indicates the best score for each noise.

| Model                  | (I) | (J) | (K) | (L) | (M) | (N) | (O) | (P) | (Q) | (R) | (S) | (T) | (U) | (V) |
|------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| MGF MGV                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| SI-NR-GAN-0            | 82.7| 59.5| 69.9| 75.1| 71.7| 70.2| 72.0| 69.0 |     |
| + Rotation             | 91.4| 82.5| 102.0| 75.4| 88.9| 82.5| 86.3| 63.2 |     |
| + Shuffle              | 100.8| 92.6| 94.1| 76.9| 81.3| 76.7| 78.1| 81.0 |     |
| + Inversion (SD-NR-GAN-III) | 37.5| 33.4| 33.5| 33.9| 53.1| 55.1| 52.4| 47.2 |     |
| + All (SI-NR-GAN-III)  | 63.7| 58.4| 57.6| 38.5| 38.2| 32.7| 31.3| 70.1 |     |

Table 11. Comparison of FID on CIFAR-10 with signal-dependent noises with and without implicit relational learning. A smaller value is better. SD-NR-GAN-II without implicit relational learning (the fourth row) is equal to SI-NR-GAN-I. SD-NR-GAN-III without implicit relational learning (the sixth row) is equal to a variant of SI-NR-GAN-II, in which only color inversion is used as a transformation. Herein, we report the median score for three random seeds.

Table 10 lists the results. From these results, we found that SD-NR-GAN-0 with color inversion (namely, SD-NR-GAN-III) achieves the best performance in all cases, while SD-NR-GAN-0 with any other transformation (i.e., rotation, channel shuffle, or all transformations) does not necessarily improve the performance compared to SD-NR-GAN-0. Refer to Appendix A.7, where we discuss the effect of the transformations in SI-NR-GAN-II.

A.7. Ablation study on transformations in SI-NR-GAN-II

As described in Section 5.2, we use three transformations, i.e., rotation, channel shuffle, and color inversion, in SI-NR-GAN-II. We conducted an ablation study to reveal the impact of the transformations.

Table 9 summarizes the results. We observed that, in most cases (6/8), the score becomes the best when using all transformations. When focusing on the individual transformation, we found that a dataset dependency occurs: under additive Gaussian noise (A)/(B), rotation is the most useful, whereas under local Gaussian noise (C)/(D) and pixel-wise correlated noise (G)/(H), color inversion is the most effective. Finding the best transformation from the training data would be an interesting area for future study. Refer to Appendix A.8, where we discuss the effect of the transformations in SD-NR-GAN-III.

A.8. Comparative study on transformations in SD-NR-GAN-III

As discussed in Section 6.3, we only use color inversion in SD-NR-GAN-III because rotation and channel shuffle collapse the per-pixel signal-noise dependency that is included in typical signal-dependent noise (e.g., Figure 2(I)–(P)). For further clarification, we conducted a comparative study on the transformations in SD-NR-GAN-III.

A.9. Ablation study on implicit relational learning in SD-NR-GAN-II and -III

We conducted an ablation study to clarify the effectiveness of implicit relational learning in SD-NR-GAN-II and -III for signal-dependent noises. Table 11 lists the results. SD-NR-GAN-II without implicit relational learning is equal to SI-NR-GAN-I, and SD-NR-GAN-III without implicit relational learning is equal to a variant of SI-NR-GAN-II, in which only color inversion is used as a transformation. We found that, although SD-NR-GAN-II and -III without implicit relational learning (the fourth and sixth rows) outperform standard GAN in all cases, they are defeated by SD-NR-GAN-II and -III with implicit relational learning (the third and fifth rows) by a large margin in all cases (with a difference of over 10). These results verify the effectiveness of implicit relational learning for signal-dependent noises. We discuss the generality of SD-NR-GAN-II and -III for signal-independent noises in Appendix A.10.

A.10. Generality analysis of SD-NR-GAN-II and -III

In the main text, we demonstrate the effectiveness of SD-NR-GAN-II and -III for signal-dependent noises. However, as discussed in Sections 6.2 and 6.3, SD-NR-GAN-II and -III can also be applied to signal-independent noises because they can implicitly learn the signal-noise relationship, i.e., learn \( R(x, \sigma) = x \) in this case. To verify this claim, we examined the performance of SD-NR-GAN-II and -III on signal-independent noises (Figure 2(A)–(H)). To focus on...
an analysis of implicit relational learning, we compared SD-NR-GAN-II with SI-NR-GAN-I and SD-NR-GAN-III with a variant of SI-NR-GAN-II, in which only color inversion is used as a transformation. Using these settings, the differences among these two models are whether $G_n$ incorporates $z_n$ into an input, i.e., whether an implicit relational learning is conducted.

Table 12 summarizes the results. We found that, although in both cases SD-NR-GANs (the third and fifth rows) are defeated by SI-NR-GANs (the fourth and sixth rows), the difference is relatively small (the worst degradation is 3.6 in SD-NR-GAN-II and 2.2 in SD-NR-GAN-III). Furthermore, we observed that SD-NR-GANs outperform standard GAN by a large margin. From these results, we conclude that, although the use of SI-NR-GANs is the best when it is known that the noise is signal-independent, using SD-NR-GAN-II and -III is also reasonable when the noise type is unknown. We discuss the limitations of SI-NR-GANs for signal-dependent noises in Appendix A.9.

### A.11. Detailed analysis on GN2GC

In the main text, we employed those generators achieving the best FID for GN2GC, as described in Section 8.3. In this section, we provide other case results.

Table 13(a) shows the results for signal-independent noises. We can see that most of the models improve the PSNR compared to the original noisy input. An exception is GN2GC with SI-NR-GAN-I in LSUN BEDROOM under Brown Gaussian noise (G). In this case, SI-NR-GAN-I fails to learn a clean image generator because the noise is beyond the assumption of SI-NR-GAN-I. This causes a difficulty in learning a denoiser, and the PSNR is almost the same as the noisy input. When comparing GN2GC with SI-NR-GAN-I and GN2GC with SI-NR-GAN-II, we can see that GN2GC obtains a better PSNR when using SI-NR-GAN-I, which achieves a better FID.

Table 13(b) shows the results for signal-dependent noises. We can observe that all models improve the PSNR compared to the original noisy input even when using the worst FID model (GN2GC with SD-NR-GAN-III in LSUN BEDROOM with Poisson noise (M)). Similar to the results for signal-independent noises, when comparing GN2GC with SD-NR-GAN-I, GN2GC with SD-NR-GAN-II, and GN2GC with SD-NR-GAN-III, we found that GN2GC tends to obtain a better PSNR when using SD-NR-GAN, which achieves the best FID. An exception to this is GN2GC with SD-NR-GAN-II in FFHQ under multiplicative Gaussian noise (I). In this case, SD-NR-GAN-II is worse than SD-NR-GAN-III in terms of the FID, whereas GN2GC with SD-NR-GAN-II outperforms GN2GC with SD-NR-GAN-III. This is because the better $G_n$ is obtained for SD-NR-GAN-III, while the better $G_n$ is learned with SD-NR-GAN-II.
B. Examples of generated and denoised images

In Figures 11–18, we show examples of images generated or denoised by the models described in Sections 8.1–8.3.

B.1. Examples of generated images for Section 8.1

- Figure 11: Examples of images generated for CIFAR-10 with signal-independent noises
- Figure 12: Examples of images generated for CIFAR-10 with signal-dependent noises

B.2. Examples of generated images for Section 8.2

- Figure 13: Examples of images generated for LSUN BEDROOM and FFHQ with signal-independent noises
- Figure 14: Examples of images and noises generated for LSUN BEDROOM and FFHQ with signal-independent noises
- Figure 15: Examples of images generated for LSUN BEDROOM and FFHQ with signal-dependent noises
- Figure 16: Examples of images and noises generated for LSUN BEDROOM and FFHQ with signal-dependent noises

B.3. Examples of denoised images for Section 8.3

- Figure 17: Examples of denoised images for LSUN BEDROOM and FFHQ under signal-independent noises
- Figure 18: Examples of denoised images for LSUN BEDROOM and FFHQ under signal-dependent noises
| Training images | (A) AGF | (B) AGV | (C) LGF | (D) LGV | (E) U | (F) MIX | (G) BG | (H) A+G |
|-----------------|---------|---------|---------|---------|------|---------|--------|--------|
| AmbientGAN      | FID: 26.7 | FID: 28.9 | FID: 21.8 | FID: 21.7 | FID: 28.3 | FID: 25.1 | FID: 30.3 | FID: 40.9 |
| P-AmbientGAN GAN | FID: 33.9 | FID: 122.2 | FID: 38.8 | FID: 38.0 | FID: 43.0 | FID: 32.5 | FID: 104.2 | FID: 209.7 |
| GAN SI-NR-GAN-0 | FID: 145.8 | FID: 136.0 | FID: 38.8 | FID: 38.8 | FID: 146.4 | FID: 125.6 | FID: 165.3 | FID: 265.9 |
| SI-NR-GAN-I     | FID: 40.7 | FID: 39.5 | FID: 23.1 | FID: 24.3 | FID: 38.6 | FID: 32.7 | FID: 71.6 | FID: 139.7 |
| SI-NR-GAN-II    | FID: 26.7 | FID: 27.8 | FID: 22.1 | FID: 22.4 | FID: 40.1 | FID: 24.8 | FID: 163.4 | FID: 233.2 |
| CBM3D+GAN       | FID: 36.7 | FID: 31.2 | FID: 22.4 | FID: 17.7 | FID: 31.6 | FID: 27.8 | FID: 32.2 | FID: 44.8 |
| N2V+GAN         | FID: 33.1 | FID: 38.4 | FID: 27.0 | FID: 35.9 | FID: 38.9 | FID: 30.2 | FID: 136.6 | FID: 169.1 |
| N2N+GAN         | FID: 34.6 | FID: 36.7 | FID: 22.7 | FID: 22.6 | FID: 36.4 | FID: 32.0 | FID: 163.8 | FID: 247.8 |

Figure 11. Examples of images generated for CIFAR-10 with signal-independent noises. We show $4 \times 4$ samples for each case. AmbientGAN$^\dagger$ is trained with the ground-truth noise models. N2N+GAN$^\dagger$ uses noisy image pairs when training N2N. The other models are trained using only noisy images (not including pairs) without complete noise information.
Figure 12. Examples of images generated for CIFAR-10 with signal-dependent noises. Here, $4 \times 4$ samples are shown for each case. AmbientGAN\(\dagger\) is trained with the ground-truth noise models. N2N+GAN\(\dagger\) uses noisy image pairs when training N2N. The other models are trained using only noisy images (not including pairs) without full knowledge of the noise.
Figure 13. Examples of images generated for LSUN Bedroom and FFHQ with signal-independent noises. We show 1 + 2 + 4 samples for each case. AmbientGAN† is trained with the ground-truth noise models. The other models are trained without complete noise information.
Figure 14. **Examples of images and noises generated for LSUN Bedroom and FFHQ with signal-independent noises.** Here, three samples are shown for each case. (a) During the training, only noisy images (third column) are given and clean images (first column) and ground-truth noises (second column) are not observable. (b)(c) Our proposed SI-NR-GANs learn to generate clean images (fourth and seventh columns) and noises (fifth and eighth columns) simultaneously. Their combinations (sixth and ninth columns) are given to $D_y$ and are compared with real noisy images (third column).
Figure 15. **Examples of images generated for LSUN BEDROOM and FFHQ with signal-dependent noises.** We show $1 + 2 + 4$ samples for each case. AmbientGAN† is trained with the ground-truth noise models. The other models are trained without full knowledge of the noise.
Figure 16. Examples of images and noises generated for LSUN BEDROOM and FFHQ with signal-dependent noises. We show three samples for each case. (a) During the training, only noisy images (third column) are given and clean images (first column) and ground-truth noises (second column) are not observable. (b)(c)(d) Our proposed SD-NR-GANs learn to generate clean images (fourth, seventh, and tenth columns) and noises (fifth, eighth, and eleventh columns) at the same time. Their combinations (sixth, ninth, and twelfth columns) are provided to $D_\alpha$ and are compared with real noisy images (third column).
Figure 17. Examples of denoised images for LSUN BEDROOM and FFHQ under signal-independent noises. Numbers below images indicate the PSNR for each image.
Figure 18. Examples of denoised images for LSUN BEDROOM and FFHQ under signal-dependent noises. Numbers below images indicate the PSNR for each image.
C. Implementation details

Notation. In the description of the network architectures, we apply the following notations.

- Linear: Linear layer
- Conv: Convolutional layer
- ReLU: Rectified unit \[61\]
- LReLU: Leaky rectified unit \[51, 76\]
- ResBlock: Residual block \[27\]

In the description of the training settings, we use the following notations. We used an Adam optimizer \[38\] during all experiments.

- \(\alpha\): Learning rate of Adam
- \(\beta_1\): The first-order momentum parameter of Adam
- \(\beta_2\): The second-order momentum parameter of Adam

C.1. Details on Section 8.1

C.1.1 Noise details

As described in Section 8.1, we tested 16 noises, shown in Figure 2(A)–(P), the details of which are provided in the caption.

C.1.2 Details of GANs

In this section, we provide the details of GANs (AmbientGAN, P-AmbientGAN, GAN, SI-NR-GANs, and SD-NR-GANs) described in Section 8.1.

Network architectures. Table 14 shows the generator and discriminator network architectures. We used the ResNet architectures \[27\]. Following the study on a real gradient penalty regularization (\(R_1\) regularization) \[59\], which we used as a GAN regularization, we multiplied the output of the ResNet blocks with 0.1 and did not use batch normalization \[29\]. In NR-GANs, we used similar architectures for \(G_x\) and \(G_n\). The number of dimensions of the latent vector \(d_z\) in the generator was set to 128, except for \(G_n\) in SD-NR-GAN-II and -III, where \(d_z\) was set to 256 because the noise generator \(G_n(z_n, z_x)\) in these models receives an image latent vector \(z_x\) along with a noise latent vector \(z_n\). The images were normalized within the range of \([-1, 1]\).

Training settings. As a GAN objective function, we used a non-saturating GAN loss \[21\] with \(R_1\) regularization \[59\]. We set the trade-off parameter for \(R_1\) regularization to 10. With NR-GANs, we additionally applied a diversity-sensitive regularization (DS regularization) \[78\] with a trade-off parameter of 0.02. We discuss the effect of this parameter in Appendix A.1. We trained the networks for 200k iterations using the Adam optimizer \[38\] with \(\alpha = 0.0003, \beta_1 = 0, \beta_2 = 0.99\), and a batch size of 64. We updated the generator and discriminator alternatively. Similar to prior studies \[36, 59\], we used an exponential moving average with a decay of 0.999 over the weights to produce the final generator.

C.1.3 Details of denoisers

In this section, we describe the details of the denoisers (N2N and N2V) used in Section 8.1.

Network architectures. Table 15 shows the denoiser network architecture. We used the U-net architecture \[68\]. This is the same as that used in the study on N2N \[46\]. The input images were normalized within the range of \([-0.5, 0.5]\).

Training settings. We trained the network for 200k iterations using the Adam optimizer \[38\] with \(\alpha = 0.0003, \beta_1 = 0.9, \beta_2 = 0.99\). The learning rate was kept constant during the training except for the last 30% iterations, where the learning rate was smoothly ramped down to zero. Following the study on N2N \[46\], we used a batch size of 4 for N2N. Just like the observation in the study of N2V \[44\], we found that N2V works better with a larger batch size. Hence, we used a batch size of 64 for N2V. Following the study on N2V \[44\], we manipulated 64 pixels per input image and used a uniform pixel selection (UPS) with a kernel size of \(5 \times 5\) as a masking method.
C.1.4 Evaluation details

As discussed in the main text, we used the Fréchet inception distance (FID) [28] as an evaluation metric because its validity has been demonstrated in recent large-scale studies on GANs [48, 45], and because its sensitivity to the noise has also been shown [28]. This metric measures the 2-Wasserstein distance between a real distribution $p^r$ and a generative distribution $p^g$ using the following equation:

$$d^2(p^r, p^g) = \|m^r - m^g\|^2_2 + \text{Tr}(C^r + C^g - 2(C^r C^g)^{1/2}), \quad (7)$$

where $\{m^r, C^r\}$ and $\{m^g, C^g\}$ denote the mean and covariance of the final feature vectors of the inception model calculated over real and generated samples, respectively.

| Name         | Layer                                      |
|--------------|--------------------------------------------|
| INPUT        | $y \in \mathbb{R}^{H \times W \times 3}$  |
| ENC_CONV0    | $3 \times 3$ Conv 48, LReLU               |
| ENC_CONV1    | $3 \times 3$ Conv 48, LReLU               |
| POOL1        | $2 \times 2$ Maxpool                      |
| ENC_CONV2    | $3 \times 3$ Conv 48, LReLU               |
| POOL2        | $2 \times 2$ Maxpool                      |
| ENC_CONV3    | $3 \times 3$ Conv 48, LReLU               |
| POOL3        | $2 \times 2$ Maxpool                      |
| ENC_CONV4    | $3 \times 3$ Conv 48, LReLU               |
| POOL4        | $2 \times 2$ Maxpool                      |
| ENC_CONV5    | $3 \times 3$ Conv 48, LReLU               |
| POOL5        | $2 \times 2$ Maxpool                      |
| ENC_CONV6    | $3 \times 3$ Conv 48, LReLU               |
| UPSAMPLE5    | $2 \times 2$ Upsample                     |
| CONCAT5      | Concatenate output of POOL4               |
| DEC_CONV5A   | $3 \times 3$ Conv 96, LReLU               |
| DEC_CONV5B   | $3 \times 3$ Conv 96, LReLU               |
| UPSAMPLE4    | $2 \times 2$ Upsample                     |
| CONCAT4      | Concatenate output of POOL3               |
| DEC_CONV4A   | $3 \times 3$ Conv 96, LReLU               |
| DEC_CONV4B   | $3 \times 3$ Conv 96, LReLU               |
| UPSAMPLE3    | $2 \times 2$ Upsample                     |
| CONCAT3      | Concatenate output of POOL2               |
| DEC_CONV3A   | $3 \times 3$ Conv 96, LReLU               |
| DEC_CONV3B   | $3 \times 3$ Conv 96, LReLU               |
| UPSAMPLE2    | $2 \times 2$ Upsample                     |
| CONCAT2      | Concatenate output of POOL1               |
| DEC_CONV2A   | $3 \times 3$ Conv 96, LReLU               |
| DEC_CONV2B   | $3 \times 3$ Conv 96, LReLU               |
| UPSAMPLE1    | $2 \times 2$ Upsample                     |
| CONCAT1      | Concatenate output of INPUT               |
| DEC_CONV1A   | $3 \times 3$ Conv 64, LReLU               |
| DEC_CONV1B   | $3 \times 3$ Conv 32, LReLU               |
| DEC_CONV1C   | $3 \times 3$ Conv 3                       |

Table 15. Denoiser architecture. This network is fully convolutional; therefore, it can take an arbitrary-size image as an input.

C.2 Details on Section 8.2

C.2.1 Noise details

As described in Section 8.2, we selected six noises for LSUN BEDROOM and two noises for FFHQ, such that they include variations. The noise parameters are the same as those described in Section 8.1 except for Brown Gaussian noise (G) where a $31 \times 31$ Gaussian filter is used instead of a $5 \times 5$ Gaussian filter.

C.2.2 Details of GANs

In this section, we provide the details regarding GANs (AmbientGAN, GAN, SI-NR-GANs, and SD-NR-GANs) described in Section 8.2.

Network architectures. Table 16 shows the generator and discriminator architectures for LSUN BEDROOM and FFHQ.

When calculating the FID, we used $10k$ real test samples and $10k$ generated samples, following the suggestion from previous large-scale studies on GANs [48, 45].

Table 16. Generator and discriminator architectures for LSUN BEDROOM and FFHQ.

| Generator $G(z)$                                      |
|-------------------------------------------------------|
| $z \in \mathbb{R}^{d \times 4}$                                     |
| Linear $\rightarrow 4 \times 4 \times 1024$               |
| ResBlock up 1024                                      |
| ResBlock up 512                                       |
| ResBlock up 256                                       |
| ResBlock up 128                                       |
| ResBlock up 64                                        |
| ReLU, $3 \times 3$ Conv 3                               |
| Tanh                                                  |

| Discriminator $D(y)$                                   |
|-------------------------------------------------------|
| $y \in \mathbb{R}^{128 \times 128 \times 3}$           |
| ResBlock down 64                                       |
| ResBlock down 128                                      |
| ResBlock down 256                                      |
| ResBlock down 512                                      |
| ResBlock down 1024                                     |
| ResBlock 1024                                          |
| ReLU, Global average pooling                           |
| Linear $\rightarrow 1$                                |

When calculating the FID, we used $10k$ real test samples and $10k$ generated samples, following the suggestion from previous large-scale studies on GANs [48, 45].

C.2 Details on Section 8.2

C.2.1 Noise details

As described in Section 8.2, we selected six noises for LSUN BEDROOM and two noises for FFHQ, such that they include variations. The noise parameters are the same as those described in Section 8.1 except for Brown Gaussian noise (G) where a $31 \times 31$ Gaussian filter is used instead of a $5 \times 5$ Gaussian filter.

C.2.2 Details of GANs

In this section, we provide the details regarding GANs (AmbientGAN, GAN, SI-NR-GANs, and SD-NR-GANs) described in Section 8.2.

Network architectures. Table 16 shows the generator and discriminator architectures. We used the ResNet architectures [27]. Following a previous study on $R_1$ regularization [59], which we used as a GAN regularization, we multiplied the output of the ResNet blocks with 0.1 and did not use batch normalization [29]. In NR-GANs, we used
similar architectures for $G_x$ and $G_n$. The number of dimensions of the latent vector $d_z$ in the generator was set to 256, except for $G_n$ in SD-NR-GAN-II and -III, where $d_z$ was set to 512 because the noise generator $G_n(z_n, z_x)$ in these models incorporates an image latent vector $z_x$ along with an noise latent vector $z_n$. The images were normalized within the range of $[-1, 1]$.

**Training settings.** As a GAN objective function, we used a non-saturating GAN loss [21] with $R_1$ regularization [59]. We set the trade-off parameter for $R_1$ regularization to 10. In NR-GANs, we additionally used DS regularization [78] with a trade-off parameter of 1.

During the experiments on LSUN BEDROOM, we trained the networks for 500k iterations using the Adam optimizer [38] with $\alpha = 0.0003$, $\beta_1 = 0$, $\beta_2 = 0.99$, and a batch size of 64. We updated the generator and discriminator alternatively. Similar to prior studies [36, 59], we used an exponential moving average with a decay of 0.999 over the weights to produce the final generator.

During the experiments on LSUN BEDROOM, we trained the networks for 500k iterations, and the other settings were the same as those described in LSUN BEDROOM.

C.2.3 Evaluation details

For the same reason as that described in Appendix C.1.4, we used the FID as an evaluation metric. When calculating the FID, we used 10k real test samples and 10k generated samples, following the suggestion from previous large-scale studies on GANs [48, 45].

C.3. Details on Section 8.3

C.3.1 Details of denoisers

In this section, we describe the details of the denoisers (N2C, N2N, N2V, N2S, and GN2GC) described in Section 8.3.

**Network architecture.** We used the same network architecture as that described in Appendix C.1.3 (Table 15). This is also the same as that used in the study on N2N [46]. Note that this network is fully convolutional; therefore, it can take an arbitrary-size image as an input. The input images were normalized within the range of $[-0.5, 0.5]$.

**Training settings.** During the experiments on LSUN BEDROOM, we trained the network for 500k iterations using the Adam optimizer [38] with $\alpha = 0.0003$, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. The learning rate was kept constant during the training except for the last 30% iterations, where the learning rate was smoothly ramped down to zero. Following the study on N2N [46], we used a batch size of 4 for N2N. For a fair comparison, we also used a batch size of 4 for N2C and GN2GC. This means that N2C, N2N, and GN2GC were trained under the same setting except for different input and output images. Similar to the observations discussed in Appendix C.1.3, we found that N2V and N2S operate better with a larger batch size. Hence, we used a batch size of 64 for N2V and N2S. Following the study on N2V [44], for N2V, we manipulated 64 pixels per input image and used a uniform pixel selection (UPS) with a kernel size of $5 \times 5$ as a masking method. With N2S, instead of UPS, we used random overwriting as a masking method [3], i.e., the pixel is overwritten with a random color within the range of $[-0.5, 0.5]$.

During the experiments on FFHQ, we trained the network for 300k iterations, and the other settings were the same as those in LSUN BEDROOM.

C.3.2 Evaluation details

As an evaluation metric, we used the PSNR, which is commonly used in image denoising. Herein, the score averaged over all test sets is provided.