Self-supervised methods for low-level vision : An Overview

Victor Letzelter
ENS Paris-Saclay
MVA, France
victor.letzelter@ens-paris-saclay.fr

Abstract

Since the emergence of discriminative Deep-learning-based denoising methods in the beginning of the XXIth *century, the powerful of those are now fully recognized. While the DNNs based methods were used at first in fully supervised settings (with the need of Clean - Noisy pairs:* NOISE2CLEAN*), the process has been simplified since. Indeed, the* NOISE2NOISE *algorithm which requires a pair of two images with independent Noises, and more recently, the* NOISE2VOID *method was proposed in self-supervised settings. The purpose of this overview is, at first, to describe the idea behind the recent methods. Then, a comparison of the performances of the supervised and self-supervised methods is proposed in several settings.*

1. Introduction

Deep Learning algorithms in supervised settings are reliable provided that enough Clean - Noisy pairs from the same distribution are available, which is not the case in some applications (biological or astronomical images for instance). In those cases, pairs of independant Noisy images are often more readily available, especially in static scenes [4]. With such pairs the NOISE2NOISE algorithm from [2] is available for use. The NOISE2VOID algorithm, designed by [4] is even more convenient, as it requires single Noisy images. The NOISE2VOID algorithm relies on blind-spot-networks in masking scheme setting (See section 5) in which only few pixels, regarding to the size of the image, contribute to the training of the loss function. To enhance the training process and the resulting image quality, Laine et al. have proposed more sophisticated version of this algorithm [5].

2. Related work

This article consists, at first on an brief overview of the principle of those algorithms, and a comparison of the performances of those denoising methods, using several image datasets (Kodak, BSD300, Set14) and several noise types. Especially the performances in hard conditions such as in the case of dependant noises (With the example of Brownian Noise) will be investigated further. An implementation of an extension of the Laine et al algorithm was proposed, for training and evaluating denoising models using dependant noises.

3. The Noise model

Several factors can degrade the quality of an image, including notably, the environmental conditions, the temperature of the sensor, insufficient light, dust on the screen during scanning the image. Noise can be defined as a random variable $\boldsymbol{n}$ which operates on a reference clean pixel value $\boldsymbol{x}$ of an image, providing a observed value $\boldsymbol{y}$. The three common noise types are the additive, the multiplicative and the impulse noise.

The aim of the denoising process is to estimate, for each pixel, the excepted value of the clean version of it.

The general metric which is used for measuring the efficiency of the denoising process is the Peak signal-to-noise ratio (PSNR):

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

Where $MSE = \frac{1}{3mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \|I(i,j) - K(i,j)\|^2$ with I, K respectively the clean and noisy images (in 8-bit format), and MAX_I is the maximum pixel value of the clean image.

3.1. The typical denoising framework with additive noise

The general framework for representing a noisy image is a decomposition in the form: $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$, where :

- $\boldsymbol{x}$ is the clean signal, whose *closed* components are **not** statistically independent: $p(\boldsymbol{x}_i \mid \boldsymbol{x}_j) \neq p(\boldsymbol{x}_i)$. This

dependancy between pixels is often represented using Markov Random Fields (MRFs) [4].

- $\boldsymbol{n}$ is the noise, conditionally pixel-wise independent given the signal $\boldsymbol{s}$: $p(\boldsymbol{n}|\boldsymbol{x}) = \prod_i p(\boldsymbol{n}_i|\boldsymbol{x}_i)$, with $\mathbb{E}[\boldsymbol{n}_i] = 0$.

3.2. Noise types typically used for performance evaluation

The enhanced version of NOISE2VOID proposed by [5] focuses especially in the case of i.i.d samples of noise, such that:

- **The white Gaussian Noise**: In this case, $\boldsymbol{n} \sim \mathcal{N}(0, \sigma^2)$, where σ can be known or not, and the $\boldsymbol{n}_i$ values for each pixel i are independant within each other.
- **Poisson Noise**: In this case: $\boldsymbol{y} \sim \frac{\mathcal{P}(\lambda \boldsymbol{x})}{\lambda}$
- **Uniform Impulse Noise**: In Impulse Noise with parameter $\alpha \in [0, 1]$, each pixel is with with probability α replaced with a value drawn from $(\mathcal{U}[0, 1])^3$ (provided that the pixel values are scaled to $[0, 1]$).

Laine et al. hightlight in [5] that their algorithm is on par with state of the art denoising algorithm in the case of i.i.d noise. Thus, an experiment with an example of dependant noise is a relevant experiment to be conducted. The case-study of Brownian Noise is investigated in the next section.

3.3. Brownian noise: an example of pixel-dependant noise

A Wiener process (or Brownian Motion) $W(t)$ can be obtained by integrating a white noise signal [1]:

$$W(t) = \int_0^t \frac{dW(\tau)}{d\tau} d\tau$$

To the difference of a White noise trajectory, whose power spectrum is flat

$$S_0 = \left| \mathcal{F}\left[\frac{dW(t)}{dt}\right](\omega) \right|^2 = \text{cst}$$

the power spectrum of Brownian Noise has a $\frac{1}{f^2}$ frequency spectrum

$$S(\omega) = |\mathcal{F}[W(t)](\omega)|^2 = \frac{S_0}{\omega^2}$$

The process for generating Brownian noise has thus been the following (adapted from [6] and [7])

1. Generating independant white Gaussian normal samples in a 2D grid.
2. Calculating the Fourier transform of the White noise signal.
3. Multiplying component-wise the result by the frequency matrix $M = \left(\frac{1}{i^2+j^2}\right)_{i,j}$. The choice of this frequency distribution is arbitrary, in the general form it the distribution of the $\frac{1}{f^2}$ values.
4. Applying the inverse fft 2D function and we consider the real part of the result.
5. Then, normalizing and resclaling the standard deviation of the data to obtain the willing standard deviation σ. The noise can then be added to the image, and the result will be clipped into the $[0, 1]$ segment.

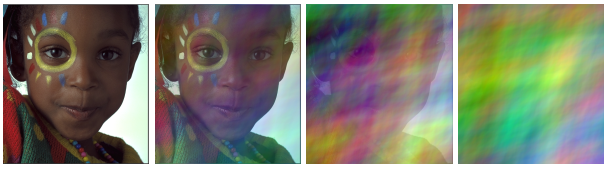


Figure 1. Brown noise generated with the homemade method, with (from left to right) the clean image, and $\sigma = 25,\ 255,\ 2550$ (in 8-bit format)

This noise is also called *red* noise, and belongs the general family colored noise or **power-law** noise, with a power spectral density per unit of bandwidth proportional to $\frac{1}{f^\beta}$.

This method has been implemented in the Tensorflow framework as an extension of the [5] code, adding this new possibility of Noise generation, for training and evaluation the Noise2clean and Noise2noise models.

Using the configuration described in Section 7.1, the models N2C (Baseline), N2N (See section 4) methods were trained on part of the ImageNet dataset ($\sim$ 7.5 hrs. of train for each model). The corresponding evaluation results are given in Figure 2. As the [5] model requires a close form of the noise expression, it was not evaluated here.

Means of the PSNR			
Method	KODAK	BSD300	SET14
N2C	24.856	24.868	24.866
N2N	25.516	25.574	24.966

Figure 2. Performances of the methods with Brownian Noise with $\sigma = 50$.

The models didn't handle well this type of noise. Visual results are given in Figure 3.

4. The Noise2noise model

The Noise2noise model was proposed by [2] in order overcome the lack of clean images, which requires long

Figure 3. From left to white, Clean image - Noisy image (27.91 dB) - Denoised with N2N (27.97 dB) - Denoised with N2C (27.95 dB)

exposures [2]. In Noise2noise settings, the data at disposal is in the form $(\boldsymbol{s}+\boldsymbol{n}, \boldsymbol{s}+\boldsymbol{n}')$ where $\boldsymbol{n}$ and $\boldsymbol{n}'$ are two independant realizations of a noise; the noisy target is used instead of the clean image $\boldsymbol{s}$.

Given those settings network then faces the issue of mapping one noisy instance to an other, which is an impossible task. In such settings, it can be shown that the network will try to perform this task outputting the *expected* image mapping, which is in fact, the clean version of it.

5. The Noise2void model

The NOISE2VOID method [4] was created as an alternative of the NOISE2NOISE method to overcome the cases when independant noisy realization of the same scene are not available.
This method consists of predicting, the clean version of each pixel $\hat{\boldsymbol{s}}_i$ given, as input, its close neighborhood deprived from the point $\hat{\boldsymbol{x}}_i$: the $(\tilde{\boldsymbol{x}}^j_{\mathrm{RF}(i)})$ points. The network consists then of solving the corresponding optimization problem:

$$\arg\min_{\boldsymbol{\theta}} \sum_{i,j} L\left(f\left(\tilde{\boldsymbol{x}}^j_{\mathrm{RF}(i)}; \boldsymbol{\theta}\right), \boldsymbol{x}^j_i\right)$$

6. Improvement of the method by [laine]

On his paper, [5] highlights that the implementation of the NOISE2VOID method proposed by [4] uses a masking scheme instead of a blind spot design, which reduces the training efficiency. An an enhancement, [5] proposed a innovative computation using a denoiser with four branches, each of them rescriting the receptive field to a given space direction.
The author furthermore proposed a Bayesian reasoning for improving the performance of the model in the cases when the noise model $p(\boldsymbol{x}|\boldsymbol{s})$ is known. The based is based on inferring, after NN training, the evaluation of the mean μ_x and the variance Σ_x of a Gaussian approximation of the $p\left(\boldsymbol{s}|\Omega_x\right)$, where Ω_x denotes the context of $\boldsymbol{x}$, and deducing at test time

$$\mathbb{E}_{\boldsymbol{x}}\left[p\left(\boldsymbol{x} \mid \boldsymbol{y}, \Omega_y\right)\right] \propto \int \underbrace{p(\boldsymbol{y} \mid \boldsymbol{x})}_{\text{Noise model}} \underbrace{p\left(\boldsymbol{x} \mid \Omega_y\right)}_{\text{Unobserved}} \mathrm{d}\mu(\boldsymbol{x})$$

7. Comparison of the methods

7.1. Technical configuration and settings

The Tensorflow implementation and pretrained models used were extracted from [3], were executed on Colab Pro with GPU T4. As training datasets for each of the N2C, N2N and [5] models, 44328 images from ImageNet were used, with sizes 256x256 and 512x512 pixels. The validation datasets KODAK, BSD300 and SET14 were used. The noise types were used either with constant and uniformly distributed parameters: Gaussian ($\sigma = 25$ or $\sigma \in [5, 50]$), Poisson ($\lambda = 30$ or $\lambda \in [5, 50]$), Impulse ($\alpha = 0.5$ or $\alpha \in [0, 1]$) and Brownian Noise ($\sigma = 50$).

7.2. Experiments

The Figures 5 to 10 reproduce the results from the TABLE 2 and TABLE 3 or the [5] paper, with **3-digits** instead of 2 digits of precision. Otherwise, the performances of the methods with regards to the size of the training set was also tackled (See Figure 7.5).

7.3. Remarks and interpretation

PSNR(N2N)−PSNR([5])					
Gauss25	G5-50	Poisson30	P5-50	Imp50	I0-100
0.009	0.296	0.145	0.151	-0.045	0.092

Figure 4. Mean difference between PSNRs with [5] and N2N methods

The mean of the differences between N2N and [5] performances for different noise types is given in Figure 4. We notice that, except for the Gaussian and impulse noises, [5] performs better in the case of constant parameter (σ or α). The second experiment shows, otherwise, that the [5] is slightly more efficient that the N2C Baseline model when dealing with smaller training sets when dealing Gaussian constant Gaussian Noise ($\sigma = 25$). When dealing with the complete training set, the N2C Baseline method still provides, on average, the best results.

7.4. Conclusions

In addition of being convenient when missing clean data or independant noisy data, the model of [5] provides competitive results, with regards to the N2C Baseline. The next possible path of investigation could be to propose a model for the noise type in specific real settings, and to study the performance of the algorithms in such conditions.

7.5. Annex: Details of the results

Means of the PSNR				
Method	σ known ?	KODAK	BSD300	SET14
N2C	Yes	32.466	31.084	31.259
N2N	Yes	32.448	31.074	31.231
[5]	Yes	32.451	31.027	31.247

Figure 5. With $\boldsymbol{n} \sim \mathcal{N}(0, \sigma^2 = 25^2)$

Means of the PSNR				
Method	σ known ?	KODAK	BSD300	SET14
N2C	Yes	32.581	31.250	31.251
N2N	Yes	32.572	31.246	31.241
[5]	Yes	32.474	31.156	30.541

Figure 6. With $\boldsymbol{n} \sim \mathcal{N}(0, \sigma^2)$, $\sigma \in [5, 50]$

Means of the PSNR				
Method	σ known ?	KODAK	BSD300	SET14
N2C	Yes	31.806	30.401	30.451
N2N	Yes	31.795	30.392	30.442
[5]	Yes	31.653	30.249	30.290

Figure 7. With $\boldsymbol{x} \sim \frac{\mathcal{P}(\lambda \boldsymbol{s})}{\lambda}$, $\lambda = 30$

Means of the PSNR				
Method	σ known ?	KODAK	BSD300	SET14
N2C	Yes	31.322	29.897	29.966
N2N	Yes	31.312	29.891	29.966
[5]	Yes	31.152	29.742	29.824

Figure 8. With $\boldsymbol{x} \sim \frac{\mathcal{P}(\lambda \boldsymbol{s})}{\lambda}$, $\lambda \in [5, 50]$

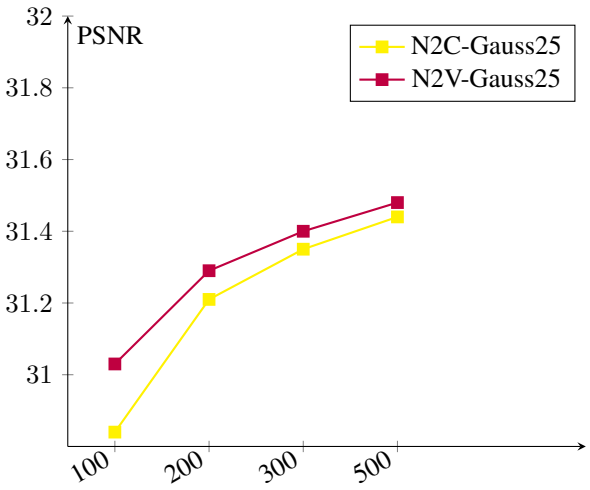
Means of the PSNR				
Method	σ known ?	KODAK	BSD300	SET14
N2C	Yes	33.318	31.194	31.447
N2N	Yes	32.877	30.847	30.961
[5]	Yes	32.978	30.772	31.070

Figure 9. With impulse noise ($\alpha = 0.5$)

Means of the PSNR				
Method	σ known ?	KODAK	BSD300	SET14
N2C	Yes	31.725	30.388	29.645
N2N	Yes	31.566	30.262	29.368
[5]	Yes	31.431	30.156	29.335

Figure 10. With impulse noise ($\alpha \in [0, 1]$)

Avg. PSNR against # of training images (from [5])



References

[1] Wikipedia contributors. *Wikipedia, The Free Encyclopedia*. [Online; accessed 22-July-2004]. 2004. URL: https://de.wikipedia.org/wiki/1/f%C2%B2-Rauschen.

[2] Jaakko Lehtinen et al. "Noise2noise: Learning image restoration without clean data". In: *arXiv preprint arXiv:1803.04189* (2018).

[3] Joshua Batson and Loic Royer. "Noise2self: Blind denoising by self-supervision". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 524–533.

[4] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. "Noise2void-learning denoising from single noisy images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2129–2137.

[5] Samuli Laine et al. "High-quality self-supervised deep image denoising". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 6970–6980.

[6] Matlab contributors. *1D colored Noise generation with Matlab*. URL: https://www.mathworks.com/matlabcentral/fileexchange/42919-pink-red-blue-and-violet-noise-generation-with-matlab.

[7] Matlab contributors. *Generate spatial data*. URL: https://www.mathworks.com/matlabcentral/fileexchange/5091-

generate-spatial-data?w.mathworks.com.