Implementing Noise2Noise and Zero-Shot Noise2Noise: Image Restoration without clean data

Debanjan Saha **Omar Muhammad** **Swarnava Chakraborty** **Vipul Tejwani**

Abstract

Given the challenge of acquiring clean data for supervised denoising of images, the Noise2Noise (N2N) method offers a valuable alternative. This work investigates image denoising using the Noise2Noise framework and its extension, Zero-Shot Noise2Noise (ZS-N2N), which operates on a single noisy image. We demonstrate its performance through comparative experiments against classical approaches and further exploring the sensitivity of the ZS-N2N model to key parameters like network depth and loss function weighting.

1 Introduction

The fundamental premise of N2N (Lehtinen et al. [2018]) is that, given a set of pairs of noisy images y_1 = **x** + n_1 and y_2 = **x** + n_2, where **x** is the clean image and n_1 and n_2 are independent noise realizations *of zero mean*, we have,

$$\begin{aligned} L_{noisy} &= E_{x,n_1,n_2}[(f_\theta(y_1) - y_2)^2] \\ &= E_{x,n_1,n_2}[(f_\theta(y_1) - (x + n_2))^2] \\ &= E_{x,n_1}[(f_\theta(y_1) - x)^2] - 2E_{x,n_1,n_2}[(f_\theta(y_1) - x)n_2] + E_{x,n_1,n_2}[n_2^2] \end{aligned}$$

Since n_2 has zero mean and is independent of x and n_1 (and thus y_1), the middle term is zero:

$$E_{x,n_1,n_2}[(f_\theta(y_1) - x)n_2] = E_{x,n_1}[f_\theta(y_1) - x]E_{n_2}[n_2] = 0$$

Let $\sigma^2 = E[n_2^2]$ (variance of the noise).

$$L_{noisy} = E_{x,n_1}[(f_\theta(y_1) - x)^2] + E[n_2^2]$$

$$L_{noisy} = L_{clean} + \sigma^2$$

As σ^2 is independent of parameters θ,

$$\arg\min_\theta(L_{noisy}) = \arg\min_\theta(L_{clean})$$

Hence in theory, N2N reaches the same performance as N2C if the dataset is infinitely large and noise is zero mean.

In ZS-N2N, we use a single noisy image **y** = **x** + **e** to generate two downsampled versions that approximate an N2N pair — ie they retain similar underlying signal of the original image and have independent noise components. A small network is then trained on this single image.

2 Methodology

2.1 Noise2Noise

This method does not have a specific methodology and can be trained with any form of neural networks, as long as we know what noise to remove.That being said, we do have different methodologies for different types of noises. We performed our experiments with two different neural networks. The testing metric was chosen to be Peak Signal-to-Noise Ratio (see Appendix A.2).

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2.2 Zero-Shot Noise2Noise

2.2.1 Downsampling

Given a noisy input image y ($H \times W \times C$), two downsampled images, $D_1(y)$ and $D_2(y)$ (each $H/2 \times W/2 \times C$), are generated via 2D convolutions (stride 2) with fixed kernels:

- Kernel $k_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \rightarrow D_1(y) = y \otimes k_1$
- Kernel $k_2 = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} \rightarrow D_2(y) = y \otimes k_2$

This retains all pixel information while aiding in separating noise components (see Appendix A.3 for details on noise independence).

2.2.2 Training Objective (Loss Function)

A lightweight network f_θ is trained per image y using the following objective:

Residual Learning: The network f_θ is trained to estimate the noise.

Symmetric Residual Loss ($\mathcal{L}_{\text{res}}$):

$$\mathcal{L}_{\text{res}}(\theta) = \frac{1}{2}\left(\|D_1(y) - f_\theta(D_1(y)) - D_2(y)\|_2^2 + \|D_2(y) - f_\theta(D_2(y)) - D_1(y)\|_2^2\right) \tag{1}$$

Symmetric Consistency Loss ($\mathcal{L}_{\text{cons}}$): This term acts as a regularizer, enforcing scale consistency and preventing overfitting (see Appendix A.4 for rationale).

$$\mathcal{L}_{\text{cons}}(\theta) = \frac{1}{2}\left(\|D_1(y) - f_\theta(D_1(y)) - D_1(y - f_\theta(y))\|_2^2 + \|D_2(y) - f_\theta(D_2(y)) - D_2(y - f_\theta(y))\|_2^2\right) \tag{2}$$

Final Loss: The network is trained by minimizing $\mathcal{L}(\theta) = \mathcal{L}_{\text{res}}(\theta) + \mathcal{L}_{\text{cons}}(\theta)$ via gradient descent (typically 1k–2k iterations).

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{res}}(\theta) + \mathcal{L}_{\text{cons}}(\theta) \tag{3}$$

2.2.3 Network Architecture

ZS-N2N uses a **simple and lightweight network** to prevent overfitting on the single image pair. The network consists of Two convolutional layers with 3×3 kernels, followed by a final 1×1 convolutional layer. It contains **no normalization** or **pooling layers**. This results in approx. **20,000 parameters**, enabling fast training/inference (CPU feasible).

2.2.4 Denoising Step

Once parameters $\hat{\theta}$ are optimized for image y, the final denoised image $\hat{x}$ is obtained:

$$\hat{x} = y - f_{\hat{\theta}}(y) \tag{4}$$

3 Experiments

In our study, we explored and compared different denoising approaches using both classical methods and deep learning-based models. We implemented and trained two neural network architectures — a simple CNN and a U-Net — on images corrupted with Gaussian and Poisson noise. Additionally, we evaluated the Zero-Shot Noise2Noise (ZS-N2N) method by varying key hyperparameters like network depth and loss balancing.

3.1 Noise2Noise

We have used two neural network architectures in our experiments, one of which is a simple CNN and the other a residual network called U -Net (Ronneberger et al. [2015]). For more details on the network architectures, see the Appendix A.5.

We used a dataset of 500 training images randomly selected from ImageNet 256×256 and 50 test images comprising 24 Kodak images and 26 from BSD300. Gaussian noise (levels 25, 50, 75) and Poisson noise (λ = 15, 20, 25) are added to simulate corruption. Models are trained with both noisy and clean targets, resulting in 12 model variants.

We have used two well known denoising techniques for comparison, namely BM3D for Gaussian (Dabov et al. [2007]) and ANSC for Poisson (Makitalo and Foi [2013]). More about these techniques can be found in the Appendix A.6 and A.7. The implementation code is available in Appendix A.8.

Fig 3.1 shows how the convergence speed is affected in noise-to-noise compared to noise-to-clean with different levels of noise (gaussian with standard deviation 25 and 50)
Table 1 gives a comparison of the PSNR values for different methods used. We have taken the values as $mean \pm 2\sigma$ where the mean and σ is calculated for all 50 test images.

Figure 1: Convergence of Noise2Noise

Figure 2: Sample Image Denoising

Table 1: Average PSNR (dB) of Models Under Different Noise Conditions

Model	Gaussian Noise			Poisson Noise		
	$\sigma = 25$	$\sigma = 50$	$\sigma = 75$	$\lambda = 15$	$\lambda = 20$	$\lambda = 25$
Noise Level (image)	20.45 ± 0.48	14.88 ± 0.64	12.01 ± 0.56	16.08 ± 1.58	17.25 ± 1.62	18.16 ± 1.66
Simple CNN (clean)	**28.16 ± 1.98**	24.45 ± 2.22	22.24 ± 2.12	**25.41 ± 2.50**	26.01 ± 2.62	**26.50 ± 2.38**
N2C (UNet)	27.37 ± 2.40	**25.05 ± 3.16**	**24.19 ± 3.38**	25.22 ± 2.86	**26.36 ± 2.98**	26.36 ± 3.00
Simple CNN (noisy)	27.30 ± 1.54	23.60 ± 2.22	21.34 ± 2.44	24.66 ± 2.62	25.52 ± 2.62	25.98 ± 2.42
N2N (UNet)	27.55 ± 2.34	24.20 ± 2.88	**22.54 ± 2.94**	24.73 ± 3.22	25.68 ± 2.94	26.41 ± 2.94
Zero-shot N2N	**28.40 ± 2.78**	**24.21 ± 2.68**	21.50 ± 2.48	**25.20 ± 3.16**	**26.13 ± 3.02**	**26.78 ± 3.02**
BM3D	28.09 ± 4.32	24.87 ± 4.45	22.88 ± 4.52	–	–	–
ANSC	–	–	–	22.13 ± 5.00	22.23 ± 5.06	22.29 ± 5.10

3.1.1 U-Net

Training is done using a batch size of 32, learning rate of 1×10^{-4} with the Adam optimizer, and up to 100 epochs with early stopping after 10 epochs of no improvement. A scheduler reduces the learning rate by a factor of 0.2 after 5 stagnant epochs. 15% of training data is used for validation. Images are resized to 256×256, normalized, and loaded with error handling. We have used techniques like normalization and skip connections in our network.

3.1.2 CNN

Training is conducted with a batch size of 32, a learning rate of 1×10^{-3} using the Adam optimizer, and for a total of 50 epochs.

3.2 Zero-Shot Noise2Noise

The training was done for 3000 epochs using Adam optimizer with an initial learning rate of 0.001 which decayed by a factor of 0.5 every 1000 epochs. The network consists of two 3×3 convolutional layers followed by a 1×1 convolutional layer. See Appendix A.11 for a sample denoised image.

3.3 Exploring Hyperparameter Sensitivity in Zero-Shot Noise2Noise

3.3.1 Objective

This experiment aimed to evaluate the impact of two key hyperparameters on the performance of the Zero-Shot Noise2Noise (ZS-N2N) image denoising algorithm:

a) Network depth (number of convolutional layers: $L = 2$, $L = 3$, $L = 4$).

b) The relative weighting between the residual loss and the consistency loss.

3.3.2 Method

The ZS-N2N algorithm was tested with varying network depths ($L = 2, 3, 4$ layers) and different relative weights between the residual and consistency loss terms (Res/Cons = 0.5, 1.0, 2.0, 10.0, and ∞).

Experiments were conducted on a set of 10 PNG images corrupted with Gaussian noise (level 25) and Poisson noise (level 25). For each image and configuration,the model was trained for 2000 epochs. Performance was evaluated using the average Peak Signal-to-Noise Ratio (PSNR) across all 10 images for each configuration. The implementation code is available in Appendix A.10.

3.3.3 Key Results & Observations

The graph related to the results is in Appendix A.9.

- $L = 2$ **Network:** Performed the worst overall. Its highest PSNR was achieved when the consistency loss was entirely removed (Res/Cons $= \infty$). Performance generally decreased as the consistency loss weight increased (ratio decreased).
- $L = 3$ **Network:** Achieved the best overall and most robust performance, particularly with Res/Cons ratios of 1.0 or higher, confirming the effectiveness of the model chosen in the original paper.
- $L = 4$ **Network:** Achieved high PSNR only when the consistency loss was strongly weighted (low Res/Cons ratios like 0.5). Its performance degraded sharply as the consistency loss weight decreased, falling below both $L = 3$ and $L = 2$). This indicates higher capacity networks are prone to overfitting the downsampled noise patterns and relies heavily on regularization to generalize to the full resolution image.

4 Conclusion

This work successfully implemented and evaluated the Noise2Noise (N2N) framework and its zero-shot extension, ZS-N2N, for image denoising tasks where clean target data is unavailable. Our experiments confirmed the N2N principle, showing that models trained on noisy pairs can achieve competitive performance compared to classical methods like BM3D and ANSC, particularly when appropriate network architectures like U-Nets are employed.

Furthermore, we detailed and implemented the ZS-N2N method, demonstrating its capability to denoise images effectively using only a single noisy instance. The investigation into ZS-N2N hyperparameters revealed that the 3-layer network configuration (as described in the original ZS-N2N paper ([Mansour and Heckel, 2023]) offers the best overall performance, providing a robust balance between model capacity and regularization.

References

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020. URL https://arxiv.org/pdf/2011.10566.

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16 (8):2080–2095, 2007. doi: 10.1109/TIP.2007.901238.

Tao Huang, Songjiang Li, Xu Jia, Huchuan Lu, and Jianzhuang Liu. Neighbor2neighbor: Self-supervised denoising from single noisy images. In *Conference paper at CVPR 2021*, 2021. URL https://arxiv.org/pdf/2101.02824.

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 2965–2974, 2018. URL http://proceedings.mlr.press/v80/lehtinen18a.html.

Markku Makitalo and Alessandro Foi. Optimal inversion of the generalized anscombe transformation for poisson-gaussian noise. *IEEE Transactions on Image Processing*, 22(1):91–103, 2013. doi: 10.1109/TIP.2012.2210725.

Youssef Mansour and Reinhard Heckel. Zero-Shot Noise2Noise: Efficient Image Denoising without any Data. In *Conference paper at CVPR 2023*, 2023. URL https://arxiv.org/pdf/2303.11253.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28.

A APPENDIX

A.1 Contributions

A.1.1 Debanjan Saha

- Implemented a smaller version of Noise2Noise paper working together with Vipul.
- Vipul and I implemented a simple CNN to compare with the N2N UNet models. We also trained N2C (clean targets) counterparts of the Simple CNN and N2N implementations.
- Created standard dataset for all works along with Vipul. Also wrote a brief report about experimental findings.
- I trained the models for gaussian noise and also performed BM3D.
- Did some literature review (not so rigorously) of similar denoising techniques like using stable diffusion's posterior part for gradual denoising.

A.1.2 Omar Muhammad

- Literature review of ZS-N2N.
- Implemented the experiment on hyperparameter sensitivity to see how network depth and loss-weight ratios influence the denoising.
- Created the presentation with Swarnava
- Created the report with Swarnava
- Presenting with Swarnava

A.1.3 Swarnava Chakraborty

- Literature review of the pioneer paper, Noise2Noise Lehtinen et al. [2018].
- Modified some parts of Zero-Shot code to make it compatible and work fine in our systems
- Created slides and report along with Omar, and added our works of Noise2Noise and zero-shot Noise2Noise.
- Will be presenting our findings alongside Omar

A.1.4 Vipul Tejwani

- Implemented a smaller version of Noise2Noise paper working together with Debanjan.
- Debanjan and I implemented a simple CNN to compare results with N2N. Additionally, trained a simple CNN, which takes clean images as targets and implemented N2C to check the convergence of N2N along with Debanjan.
- Created standard dataset for all works along with Debanjan.
- I trained the models for Poisson noise and also compared the results of N2N and Zero-shot N2N with Anscombe Image denoiser (which uses BM3D), which is used for Poisson Noise.

A.2 Peak Signal-to-Noise Ratio (PSNR)

This is a very commonly used metric for determining how good our CNN model has performed the image analysis task.

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

where $MAX =$ maximum possible pixel value in image (generally 255 for 8-bit RGB images) and MSE is the mean squared error between the predicted image and the actual image.

More PSNR implies a better model and vice versa.

A.3 Noise Independence in Downsampled Images

The noise in the two downsampled images, $D_1(y)$ and $D_2(y)$, becomes independent due to the way the downsampling kernels operate. Each output pixel in $D_1(y)$ is the average of a distinct pair of pixels from each 2×2 patch in the original image, while each output pixel in $D_2(y)$ is the average of the other two pixels from the same patches.

Assuming the noise in the original image is pixel-wise independent and zero-mean, the averaging operation does not introduce correlation. Therefore, the noise components in the two downsampled images are distinct and uncorrelated, making them independent.

A.4 Rationale for Consistency Loss

The Symmetric Consistency Loss ($\mathcal{L}_{\text{cons}}$, see Eq. 2) serves as a crucial regularizer in the ZS-N2N training process. While the Residual Loss ($\mathcal{L}_{\text{res}}$, Eq. 1) trains the network f_θ using only the downsampled images ($D_1(y), D_2(y)$), the ultimate goal is to apply the trained network to the full-resolution image y.

The consistency loss bridges this gap by enforcing that the network's behavior is coherent across scales. It minimizes the difference between:

1. Denoising the downsampled image directly: $D_i(y) - f_\theta(D_i(y))$
2. Denoising the full image first, then downsampling: $D_i(y - f_\theta(y))$

By penalizing discrepancies between these two pathways, the consistency loss discourages the network from merely memorizing noise patterns specific to the downsampled views ($D_1(y), D_2(y)$). It encourages f_θ to learn a more general noise representation applicable at both the downsampled and full resolutions. This constraint helps prevent overfitting to the limited downsampled training data and improves the quality and generalization of the final denoising result applied to the original image y.

A.5 Network architecture details

A.5.1 U-Net

The model is implemented using a U-Net architecture that features a contracting (encoder) path and an expanding (decoder) path with symmetric skip connections. The key components are:

- **DoubleConv Modules:** Two sequential convolutional layers, each followed by batch normalization and a ReLU activation.
- **Downsampling (Down) Blocks:** Max pooling operations followed by DoubleConv to reduce spatial dimensions and increase feature depth.
- **Upsampling (Up) Blocks:** Bilinear interpolation (or transposed convolutions) for upscaling, where the upsampled features are concatenated with corresponding encoder features via skip connections.
- **Output Layer (OutConv):** A final 1×1 convolution to map the features back to the desired number of channels.

A.5.2 Simple CNN

The model is a straightforward CNN designed for image denoising, consisting of three convolutional layers interleaved with a LeakyReLU activation. The key components are:

- **First Convolution:** A 3×3 convolution that maps the 3-channel noisy input to an embedding space with 48 channels, followed by a LeakyReLU activation.
- **Second Convolution:** A 3×3 convolution operating on the 48-channel feature space, again followed by a LeakyReLU activation.
- **Output Convolution:** A 1×1 convolution that directly predicts the clean image, preserving the 3-channel structure.

A.6 Block Matching and 3D Filtering (BM3D)

This is a traditional, non-learning way which exploits Self-similarity of natural images. Some features are given as follows:

- **Key Idea:** Exploit non-local self-similarity in natural images by grouping similar patches and filtering them jointly.
- **Two-Stage Process:**
  1. **Basic Estimate:**
     - Find similar patches using block-matching.
     - Stack them into a 3D array (group).
     - Apply 3D linear transform (e.g., DCT), then threshold coefficients.
     - Inverse transform and aggregate overlapping patches.
  2. **Final Estimate:**
     - Repeat the process using the basic estimate to improve grouping.
     - Use Wiener filtering in the transform domain for better accuracy.
- **Why It Works:**
  - Grouping similar patches increases redundancy.
  - Transform domain sparsifies the signal, making thresholding effective.
- **Output:** High-quality denoised image with minimal loss of detail and texture.

A.7 Adaptive Non-Local Sparse Coding (ANSC)

We used this, in our case, for denoising Poisson noise and comparing with Noise2Noise. For converting from Poisson to Gaussian Noise, this method uses something called Anscombe Transform.

A.7.1 Anscombe transform

$$A(Y) = 2\sqrt{Y + 3/8}$$

Using Taylor expansion of first order:

$$A(Y) \approx A(\lambda) + A'(\lambda)(Y - \lambda)$$

Taking variances (assuming $Y \sim \text{Poisson}(\lambda)$ so $E[Y] = \lambda$, $Var(Y) = \lambda$):

$$\begin{aligned}\text{Var}(A(Y)) &\approx \text{Var}(A(\lambda) + A'(\lambda)(Y - \lambda)) \\ &\approx (A'(\lambda))^2 \text{Var}(Y - \lambda) \\ &\approx (A'(\lambda))^2 \text{Var}(Y)\end{aligned}$$

Since $A'(\lambda) = \frac{d}{d\lambda}(2\sqrt{\lambda + 3/8}) = 2 \cdot \frac{1}{2\sqrt{\lambda+3/8}} = \frac{1}{\sqrt{\lambda+3/8}}$, we have $(A'(\lambda))^2 = \frac{1}{\lambda+3/8}$.

$$\text{Var}(A(Y)) \approx \frac{1}{\lambda + 3/8}\text{Var}(Y) = \frac{1}{\lambda + 3/8} \cdot \lambda = \frac{\lambda}{\lambda + 3/8}$$

For $\lambda \gg 3/8$:

$$\text{Var}(A(Y)) \approx 1$$

Thus, we get a Gaussian distribution with approximately constant variance.

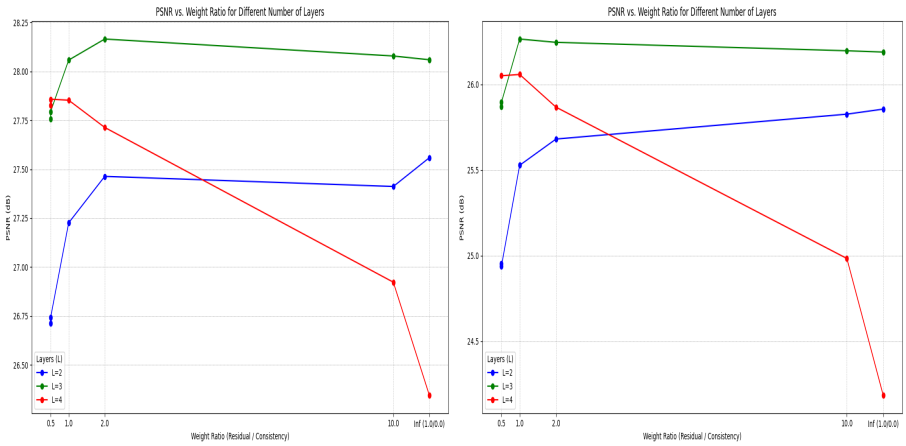
A.7.2 Process

- **Pipeline for Poisson Denoising:**
  1. Apply Anscombe Transform: $A(Y)$
  2. Denoise assuming Gaussian noise (e.g., BM3D)
  3. Apply **inverse** Anscombe Transform to get denoised estimate of X
- **Optimal Inversion (Mäkitalo and Foi, 2011):**
  - A non-linear, unbiased inverse is used to correct bias in naive inversion.
  - Avoids artifacts at low intensities.
- **Why It Works:**
  - Gaussian denoisers can now be used effectively.
  - Works well especially at moderate to high count levels.

A.8 Noise2Noise Code

`https://github.com/deban9017/Noise2Noise-Project`

A.9 Graph showing comparison of different layers for Zero-Shot



A.10 Hyperparameter Sensitivity Code

`https://www.kaggle.com/code/omarm1/notebook0fdaa2ee9d`

A.11 Sample Image for ZS-N2N

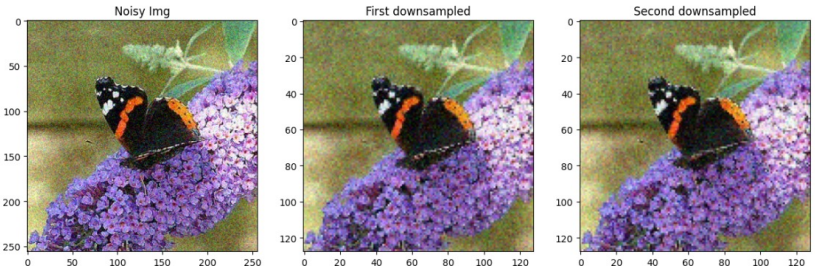


Figure 3: Downsampled Images

Figure 4: Denoised Image

A.12 Sample Images for all models

Figure 5: Original vs Noisy vs Denoised Images (Gaussian noise, $\sigma = 25$

Figure 6: Original vs Noisy vs Denoised Images (Gaussian noise, $\sigma = 25$

A.13 Github Link

`https://github.com/deban9017/Noise2Noise-Project`