Noise2Noise and Zero-Shot Noise2Noise

Image Restoration without clean data

Swarnava Chakraborty, Debanjan Saha,
Vipul Tejwani, Omar Muhammad

Indian Institute of Science, Bangalore

April 14, 2025

Goals of the Noise2Noise

- Challenge the belief that clean data is necessary for supervised denoising.

Goals of the Noise2Noise

- Challenge the belief that clean data is necessary for supervised denoising.
- Prove that training on noisy pairs can achieve the same result as using clean targets.

Goals of the Noise2Noise

- Challenge the belief that clean data is necessary for supervised denoising.
- Prove that training on noisy pairs can achieve the same result as using clean targets.
- Generalize the approach to various tasks: denoising, super-resolution.

Goals of the Noise2Noise

- Challenge the belief that clean data is necessary for supervised denoising.
- Prove that training on noisy pairs can achieve the same result as using clean targets.
- Generalize the approach to various tasks: denoising, super-resolution.
- Emphasize practical value avoid clean data collection.

Goals of the Noise2Noise

- Challenge the belief that clean data is necessary for supervised denoising.
- Prove that training on noisy pairs can achieve the same result as using clean targets.
- Generalize the approach to various tasks: denoising, super-resolution.
- Emphasize practical value avoid clean data collection.
- Applicable to any noise with zero mean.

Goals of the Noise2Noise

- Challenge the belief that clean data is necessary for supervised denoising.
- Prove that training on noisy pairs can achieve the same result as using clean targets.
- Generalize the approach to various tasks: denoising, super-resolution.
- Emphasize practical value avoid clean data collection.
- Applicable to any noise with zero mean.
- Demonstrates surprising effectiveness across domains.

Goals of Zero-Shot Noise2Noise (ZS-N2N)

- Extend Noise2Noise to a setting where only a single noisy image is available.

Goals of Zero-Shot Noise2Noise (ZS-N2N)

- Extend Noise2Noise to a setting where only a single noisy image is available.
- Create pseudo noisy pairs by downsampling simulate i.i.d. noise.

Goals of Zero-Shot Noise2Noise (ZS-N2N)

- Extend Noise2Noise to a setting where only a single noisy image is available.
- Create pseudo noisy pairs by downsampling simulate i.i.d. noise.
- Train a compact network from scratch per image.

Goals of Zero-Shot Noise2Noise (ZS-N2N)

- Extend Noise2Noise to a setting where only a single noisy image is available.
- Create pseudo noisy pairs by downsampling simulate i.i.d. noise.
- Train a compact network from scratch per image.
- Eliminate the need for large datasets make N2N zero-shot and image-specific.

Goals of Zero-Shot Noise2Noise (ZS-N2N)

- Extend Noise2Noise to a setting where only a single noisy image is available.
- Create pseudo noisy pairs by downsampling simulate i.i.d. noise.
- Train a compact network from scratch per image.
- Eliminate the need for large datasets make N2N zero-shot and image-specific.
- Focus on practical use in low-data or real-world scenarios.

Theory Behind Noise2Noise

- Given noisy pairs: $y_1 = x + n_1$, $y_2 = x + n_2$, where n_1, n_2 are i.i.d. zero-mean noise.

Theory Behind Noise2Noise

- Given noisy pairs: $y_1 = x + n_1$, $y_2 = x + n_2$, where n_1, n_2 are i.i.d. zero-mean noise.
- Define loss:

$$\mathcal{L}_{\text{noisy}} = \mathbb{E}_{x,n_1,n_2}[(f_\theta(y_1) - y_2)^2]$$

Theory Behind Noise2Noise

- Given noisy pairs: $y_1 = x + n_1$, $y_2 = x + n_2$, where n_1, n_2 are i.i.d. zero-mean noise.
- Define loss:

$$\mathcal{L}_{\text{noisy}} = \mathbb{E}_{x,n_1,n_2}[(f_\theta(y_1) - y_2)^2]$$

- Expand:

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

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

Theory Behind Noise2Noise

- Given noisy pairs: $y_1 = x + n_1$, $y_2 = x + n_2$, where n_1, n_2 are i.i.d. zero-mean noise.
- Define loss:

$$\mathcal{L}_{\text{noisy}} = \mathbb{E}_{x,n_1,n_2}[(f_\theta(y_1) - y_2)^2]$$

- Expand:

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

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

- Zero mean makes middle term zero:

$$= \mathbb{E}_{x,n_1}[(f_\theta(y_1) - x)^2] + \sigma^2$$

Theory Behind Noise2Noise

- Given noisy pairs: $y_1 = x + n_1$, $y_2 = x + n_2$, where n_1, n_2 are i.i.d. zero-mean noise.
- Define loss:

$$\mathcal{L}_{\text{noisy}} = \mathbb{E}_{x,n_1,n_2}[(f_\theta(y_1) - y_2)^2]$$

- Expand:

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

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

- Zero mean makes middle term zero:

$$= \mathbb{E}_{x,n_1}[(f_\theta(y_1) - x)^2] + \sigma^2$$

- So:

$$\mathcal{L}_{\text{noisy}} = \mathcal{L}_{\text{clean}} + \sigma^2$$

Theory Behind Noise2Noise

- Given noisy pairs: $y_1 = x + n_1$, $y_2 = x + n_2$, where n_1, n_2 are i.i.d. zero-mean noise.
- Define loss:

$$\mathcal{L}_{\text{noisy}} = \mathbb{E}_{x,n_1,n_2}[(f_\theta(y_1) - y_2)^2]$$

- Expand:

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

- Zero mean makes middle term zero:

$$= \mathbb{E}_{x,n_1}[(f_\theta(y_1) - x)^2] + \sigma^2$$

- So:

$$\mathcal{L}_{\text{noisy}} = \mathcal{L}_{\text{clean}} + \sigma^2$$

- Hence:

$$\arg\min_\theta \mathcal{L}_{\text{noisy}} = \arg\min_\theta \mathcal{L}_{\text{clean}}$$

Finite Corrupted Data in L_2 Minimization (Part 1)

Problem Setup:

- We want to compute the expected error in L_2 norm minimization.
- We have corrupted targets $\{y_i\}_{i=1}^N$ instead of clean targets.
- N is a finite number.
- y_i are arbitrary random variables with $E\{y_i\} = \bar{y}_i$.

Key Idea: The point of least deviation is found at the mean.

Expected Squared Difference: The expected squared difference between the means across realizations of the noise is:

$$E_y\left[\left(\frac{1}{N}\sum_{i=1}^{N} y_i - \bar{y}\right)^2\right] = \frac{1}{N^2} E_y\left[\left(\sum_{i=1}^{N} y_i\right)^2 - 2\bar{y}\left(\sum_{i=1}^{N} y_i\right) + \bar{y}^2\right]$$

$$= \frac{1}{N^2}\left[E_y\left(\sum_{i=1}^{N} y_i\right)^2 - 2\bar{y}E_y\left(\sum_{i=1}^{N} y_i\right) + N^2\bar{y}^2\right]$$

Finite Corrupted Data in L_2 Minimization (Part 2)

Following on from the previous slide:

$$= \frac{1}{N^2}\left[E_y\left(\sum_{i=1}^{N} y_i^2 + \sum_{i\neq j} y_i y_j\right) - 2\bar{y}N\bar{y} + N^2\bar{y}^2\right]$$

$$= \frac{1}{N^2}\left[\sum_{i=1}^{N} Var(y_i) + \sum_{i\neq j} Cov(y_i, y_j)\right]$$

Where we have used $E(\sum y_i) = \sum E(y_i) = N\bar{y}$ and basic properties of variance and covariance.

Simplification (Mutually Uncorrelated Case): If the corruptions are mutually uncorrelated, then $Cov(y_i, y_j) = 0$ for $i \neq j$. The equation simplifies to:

$$\frac{1}{N^2}\sum_{i=1}^{N} Var(y_i) = \frac{1}{N}\left[\frac{1}{N}\sum_{i=1}^{N} Var(y_i)\right] \quad (1)$$

ZS-N2N: At a Glance

- **What:** Efficient, dataset-free image denoising method.

ZS-N2N: At a Glance

- **What:** Efficient, dataset-free image denoising method.
- **How:** Operates on a *single* noisy image.

ZS-N2N: At a Glance

- **What:** Efficient, dataset-free image denoising method.
- **How:** Operates on a *single* noisy image.
- **Requirements:** No prior training data or noise information needed.

ZS-N2N: At a Glance

- **What:** Efficient, dataset-free image denoising method.
- **How:** Operates on a *single* noisy image.
- **Requirements:** No prior training data or noise information needed.
- **Key Advantage:** Uses a lightweight network (~20k params) $\rightarrow$ Fast computation (vs. DIP/Self2Self).

ZS-N2N: At a Glance

- **What:** Efficient, dataset-free image denoising method.
- **How:** Operates on a *single* noisy image.
- **Requirements:** No prior training data or noise information needed.
- **Key Advantage:** Uses a lightweight network (~20k params) $\rightarrow$ Fast computation (vs. DIP/Self2Self).
- **Foundation:** Leverages Noise2Noise (N2N) & Neighbour2Neighbour principles via novel downsampling.

ZS-N2N: N2N Pairs from a Single Image

- **Challenge:** Obtaining N2N pairs (y_1, y_2) with same noise distribution is difficult in practice.

ZS-N2N: N2N Pairs from a Single Image

- **Challenge:** Obtaining N2N pairs (y_1, y_2) with same noise distribution is difficult in practice.
- **ZS-N2N Approach:** Generate an *approximate* N2N pair from the *single* input noisy image y.

ZS-N2N: N2N Pairs from a Single Image

- **Challenge:** Obtaining N2N pairs (y_1, y_2) with same noise distribution is difficult in practice.
- **ZS-N2N Approach:** Generate an *approximate* N2N pair from the *single* input noisy image y.
- **Mechanism:** A specific downsampling technique creates $D_1(y)$ and $D_2(y)$.
  Such that:

ZS-N2N: N2N Pairs from a Single Image

- **Challenge:** Obtaining N2N pairs (y_1, y_2) with same noise distribution is difficult in practice.
- **ZS-N2N Approach:** Generate an *approximate* N2N pair from the *single* input noisy image y.
- **Mechanism:** A specific downsampling technique creates $D_1(y)$ and $D_2(y)$.
  Such that:
  - $D_1(y), D_2(y)$ have similar signal from x.

ZS-N2N: N2N Pairs from a Single Image

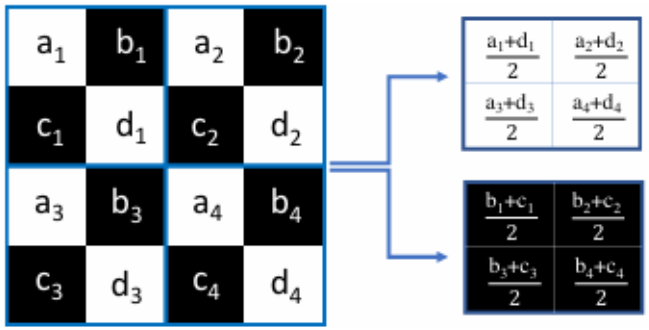
- **Challenge:** Obtaining N2N pairs (y_1, y_2) with same noise distribution is difficult in practice.
- **ZS-N2N Approach:** Generate an *approximate* N2N pair from the *single* input noisy image y.
- **Mechanism:** A specific downsampling technique creates $D_1(y)$ and $D_2(y)$.
  Such that:
  - $D_1(y), D_2(y)$ have similar signal from x.
  - $D_1(y), D_2(y)$ have (approximately) independent noise components from e.

ZS-N2N: N2N Pairs from a Single Image

- **Challenge:** Obtaining N2N pairs (y_1, y_2) with same noise distribution is difficult in practice.
- **ZS-N2N Approach:** Generate an *approximate* N2N pair from the *single* input noisy image y.
- **Mechanism:** A specific downsampling technique creates $D_1(y)$ and $D_2(y)$.
  Such that:
  - $D_1(y), D_2(y)$ have similar signal from x.
  - $D_1(y), D_2(y)$ have (approximately) independent noise components from e.
- **Training:** A small, lightweight network is trained *specifically* on this generated pair $(D_1(y), D_2(y))$ for each input image y.

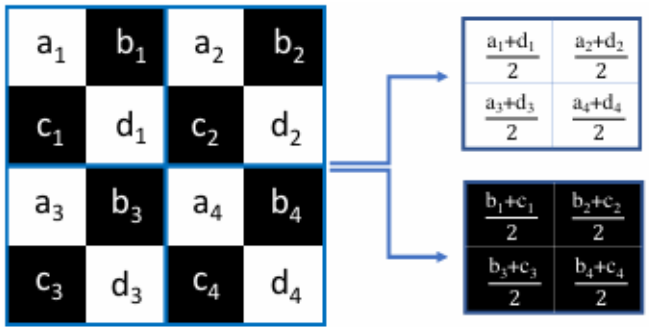
ZS-N2N Method: Image Pair Downsampler

- **Input:** Noisy image y (Size: $H \times W \times C$)



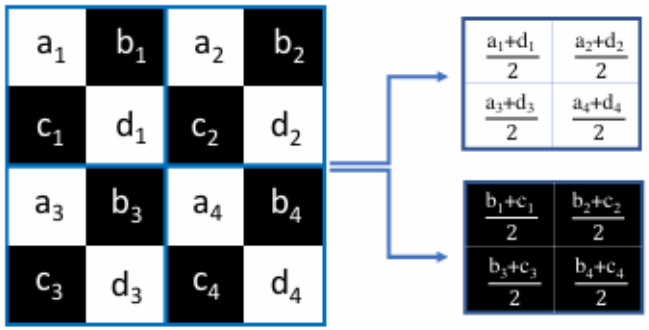
ZS-N2N Method: Image Pair Downsampler

- **Input:** Noisy image y (Size: $H \times W \times C$)
- **Output:** Two downsampled images $D_1(y), D_2(y)$ (Size: $H/2 \times W/2 \times C$)



ZS-N2N Method: Image Pair Downsampler

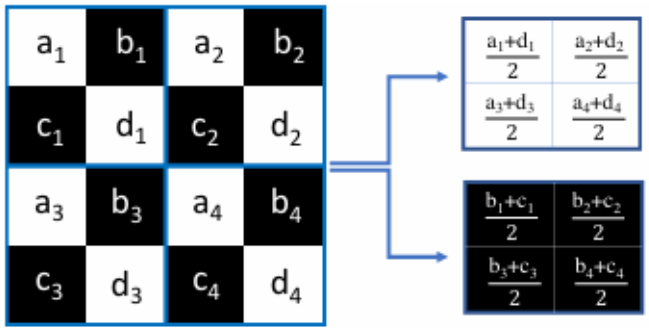
- **Input:** Noisy image y (Size: $H \times W \times C$)
- **Output:** Two downsampled images $D_1(y), D_2(y)$ (Size: $H/2 \times W/2 \times C$)
- **Process:** Channel-wise 2D convolution (stride 2) with fixed kernels:



ZS-N2N Method: Image Pair Downsampler

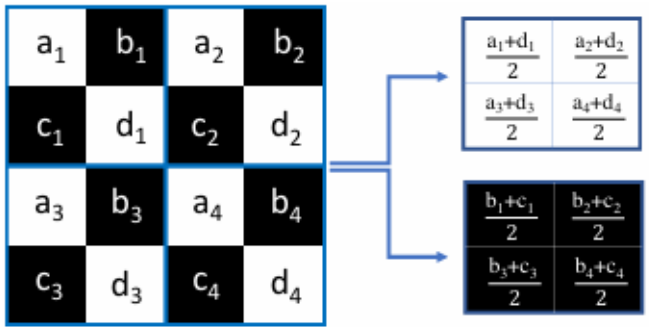
- **Input:** Noisy image y (Size: $H \times W \times C$)
- **Output:** Two downsampled images $D_1(y), D_2(y)$ (Size: $H/2 \times W/2 \times C$)
- **Process:** Channel-wise 2D convolution (stride 2) with fixed kernels:
  - Kernel $k_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ (Averages diagonal pixels)

    $\rightarrow D_1(y) = y \otimes k_1$



ZS-N2N Method: Image Pair Downsampler

- **Input:** Noisy image y (Size: $H \times W \times C$)
- **Output:** Two downsampled images $D_1(y), D_2(y)$ (Size: $H/2 \times W/2 \times C$)
- **Process:** Channel-wise 2D convolution (stride 2) with fixed kernels:
  - Kernel $k_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ (Averages diagonal pixels)
    $\rightarrow D_1(y) = y \otimes k_1$
  - Kernel $k_2 = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix}$ (Averages anti-diagonal pixels)
    $\rightarrow D_2(y) = y \otimes k_2$



ZS-N2N Method: Why is Noise Independent in D_1, D_2?

- **Mechanism:** Each pixel in $D_1(y)$ averages a *different pair* of original pixels than the corresponding pixel in $D_2(y)$ within each 2×2 patch.

ZS-N2N Method: Why is Noise Independent in D_1, D_2?

- **Mechanism:** Each pixel in $D_1(y)$ averages a *different pair* of original pixels than the corresponding pixel in $D_2(y)$ within each 2×2 patch.
- **Assumption:** Requires original noise in y to be pixel-wise independent and zero-mean.

ZS-N2N Method: Why is Noise Independent in D_1, D_2?

- **Mechanism:** Each pixel in $D_1(y)$ averages a *different pair* of original pixels than the corresponding pixel in $D_2(y)$ within each 2×2 patch.
- **Assumption:** Requires original noise in y to be pixel-wise independent and zero-mean.
- **Result:** Averaging distinct, independent noise samples preserves independence between the noise components in $D_1(y)$ and $D_2(y)$.

ZS-N2N Method: Training Loss

- **Goal:** Train lightweight network f_θ using the pair $D_1(y), D_2(y)$.

Trained via gradient descent (1k-2k iterations).

ZS-N2N Method: Training Loss

- **Goal:** Train lightweight network f_θ using the pair $D_1(y), D_2(y)$.
- **Symmetric Residual Loss ($\mathcal{L}_{\text{res}}$):** Predicts the noise, not the clean image.

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

Trained via gradient descent (1k-2k iterations).

ZS-N2N Method: Training Loss

- **Goal:** Train lightweight network f_θ using the pair $D_1(y), D_2(y)$.
- **Symmetric Residual Loss ($\mathcal{L}_{\text{res}}$):** Predicts the noise, not the clean image.

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

- **Symmetric Consistency Loss ($\mathcal{L}_{\text{cons}}$):** Regularizer; enforces scale consistency.

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

Trained via gradient descent (1k-2k iterations).

ZS-N2N Method: Training Loss

- **Goal:** Train lightweight network f_θ using the pair $D_1(y), D_2(y)$.
- **Symmetric Residual Loss ($\mathcal{L}_{\text{res}}$):** Predicts the noise, not the clean image.

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

- **Symmetric Consistency Loss ($\mathcal{L}_{\text{cons}}$):** Regularizer; enforces scale consistency.

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

- **Final Loss:** $\mathcal{L}(\theta) = \mathcal{L}_{\text{res}}(\theta) + \mathcal{L}_{\text{cons}}(\theta)$

Trained via gradient descent (1k-2k iterations).

ZS-N2N Method: Lightweight Network

- **Key Idea:** Use a *very simple* network to prevent overfitting on the downsized image pair.

ZS-N2N Method: Lightweight Network

- **Key Idea:** Use a *very simple* network to prevent overfitting on the downsized image pair.
- **Structure:**

ZS-N2N Method: Lightweight Network

- **Key Idea:** Use a *very simple* network to prevent overfitting on the downsized image pair.
- **Structure:**
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU

ZS-N2N Method: Lightweight Network

- **Key Idea:** Use a *very simple* network to prevent overfitting on the downsized image pair.
- **Structure:**
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU

ZS-N2N Method: Lightweight Network

- **Key Idea:** Use a *very simple* network to prevent overfitting on the downsized image pair.
- **Structure:**
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU
  - Conv (1x1 kernel)

ZS-N2N Method: Lightweight Network

- **Key Idea:** Use a *very simple* network to prevent overfitting on the downsized image pair.
- **Structure:**
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU
  - Conv (1x1 kernel)
- **Parameters:** $\sim$20,000 (significantly fewer than U-Nets etc.)

ZS-N2N Method: Lightweight Network

- **Key Idea:** Use a *very simple* network to prevent overfitting on the downsized image pair.
- **Structure:**
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU
  - Conv (1x1 kernel)
- **Parameters:** $\sim$20,000 (significantly fewer than U-Nets etc.)
- No normalization layers, no pooling layers.

ZS-N2N Method: Lightweight Network

- **Key Idea:** Use a *very simple* network to prevent overfitting on the downsized image pair.
- **Structure:**
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU
  - Conv (3x3 kernel) $\rightarrow$ LeakyReLU
  - Conv (1x1 kernel)
- **Parameters:** $\sim$20,000 (significantly fewer than U-Nets etc.)
- No normalization layers, no pooling layers.
- **Benefits:** Fast training/inference, CPU feasible, avoids overfitting.

ZS-N2N Method: Final Denoising Step

- **Process:** Apply the network $f_{\hat{\theta}}$ (trained specifically for image y) to the *original* noisy image y.

ZS-N2N Method: Final Denoising Step

- **Process:** Apply the network $f_{\hat{\theta}}$ (trained specifically for image y) to the *original* noisy image y.
- **Mechanism:** The network predicts the noise component. Subtract this predicted noise from the original noisy image.

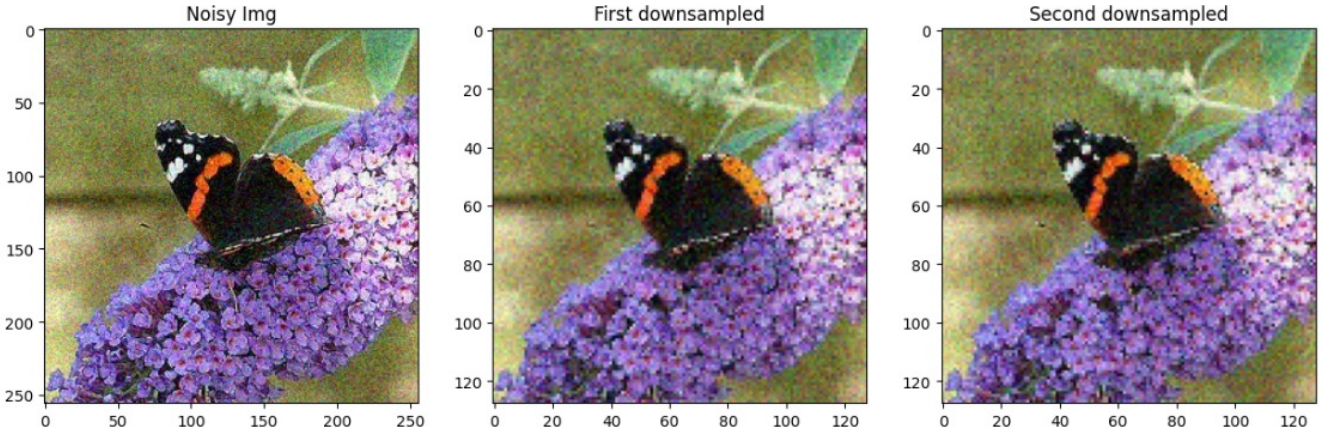
ZS-N2N Method: Final Denoising Step

- **Process:** Apply the network $f_{\hat{\theta}}$ (trained specifically for image y) to the *original* noisy image y.
- **Mechanism:** The network predicts the noise component. Subtract this predicted noise from the original noisy image.
- **Equation:**

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

where $\hat{x}$ is the clean image estimate.

Sample Image



(Downsampled Images)

(Denoised Image)

Overview of Architectures and Dataset

- Two architectures: Simple CNN and U-Net (residual network).

Overview of Architectures and Dataset

- Two architectures: Simple CNN and U-Net (residual network).
- Dataset: 500 training images from ImageNet (256×256), 50 test images (Kodak + BSD300).

Overview of Architectures and Dataset

- Two architectures: Simple CNN and U-Net (residual network).
- Dataset: 500 training images from ImageNet (256×256), 50 test images (Kodak + BSD300).
- Corruptions: Gaussian noise (25, 50, 75), Poisson noise ($\lambda = 15, 20, 25$).

Training Setup

- Models trained with noisy and clean targets (12 model variants).

Training Setup

- Models trained with noisy and clean targets (12 model variants).
- CNN: Batch 32, learning rate 1×10^{-3}, 50 epochs.

Training Setup

- Models trained with noisy and clean targets (12 model variants).
- CNN: Batch 32, learning rate 1×10^{-3}, 50 epochs.
- U-Net: Batch 32, learning rate 1×10^{-4}, 100 epochs, early stopping, learning rate scheduler.

Training Setup

- Models trained with noisy and clean targets (12 model variants).
- CNN: Batch 32, learning rate 1×10^{-3}, 50 epochs.
- U-Net: Batch 32, learning rate 1×10^{-4}, 100 epochs, early stopping, learning rate scheduler.
- images resized and normalized.

Training Setup

- Models trained with noisy and clean targets (12 model variants).
- CNN: Batch 32, learning rate 1×10^{-3}, 50 epochs.
- U-Net: Batch 32, learning rate 1×10^{-4}, 100 epochs, early stopping, learning rate scheduler.
- images resized and normalized.
- Validation: 15%

Convergence Graph

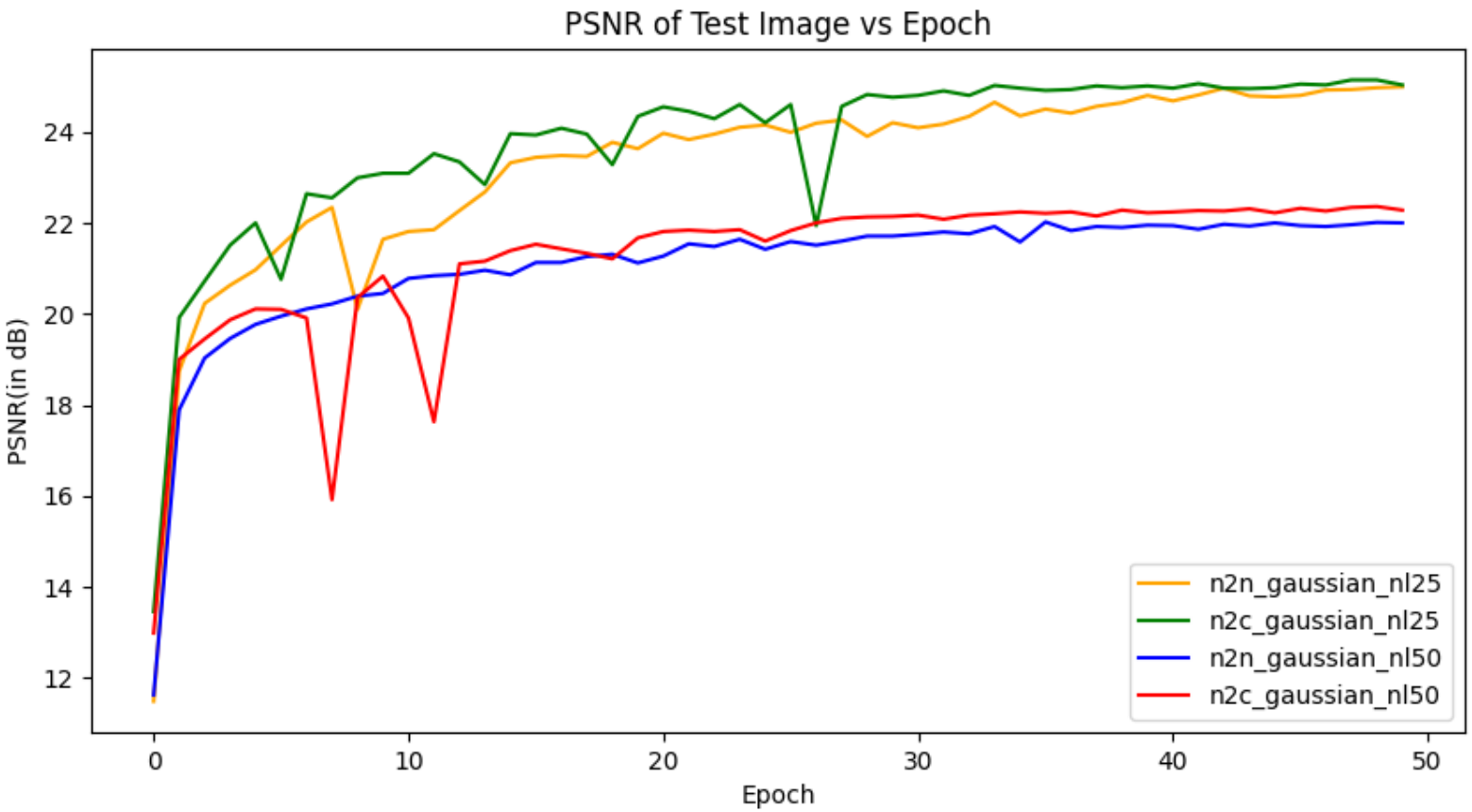


Figure 1: PSNR vs Epoch for different models

Classical Comparison Methods

- BM3D used for Gaussian denoising.

Classical Comparison Methods

- BM3D used for Gaussian denoising.
- ANSC used for Poisson denoising.

Table of Data for comparison

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

Sample Images

Zero-Shot N2N Setup and Evaluation

- ZS-N2N evaluated with L = 2, 3, 4 layer networks.

Zero-Shot N2N Setup and Evaluation

- ZS-N2N evaluated with L $= 2, 3, 4$ layer networks.
- Res/Cons loss ratios: $0.5, 1.0, 2.0, 10.0, \infty$.

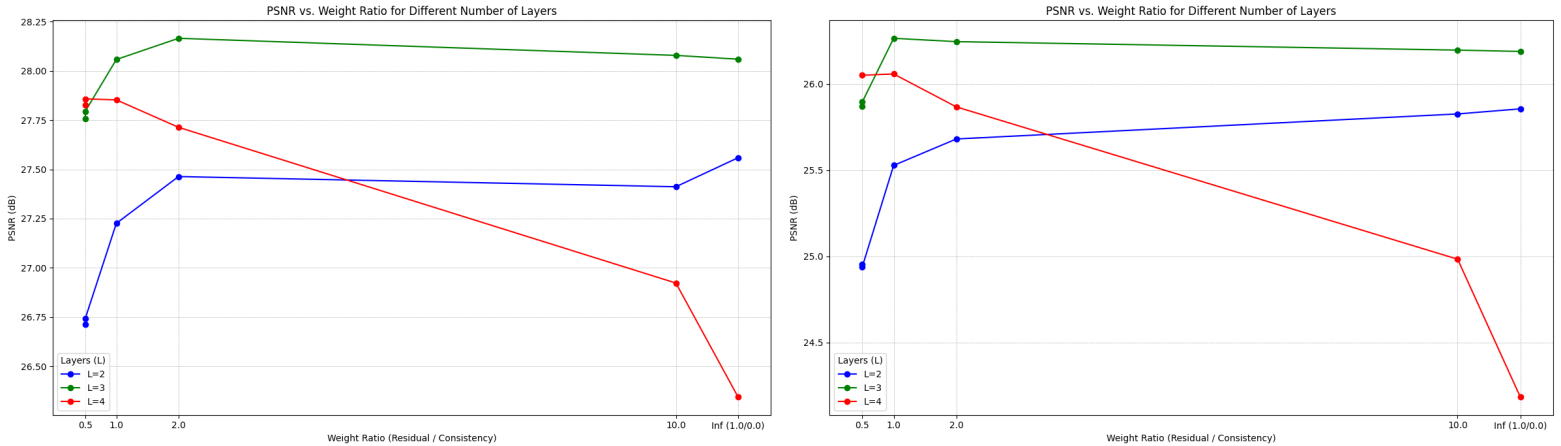
Zero-Shot N2N Setup and Evaluation

- ZS-N2N evaluated with L $= 2, 3, 4$ layer networks.
- Res/Cons loss ratios: 0.5, 1.0, 2.0, 10.0, ∞.
- Training: 2000 epochs per configuration, on 10 PNG images.

Zero-Shot N2N Setup and Evaluation

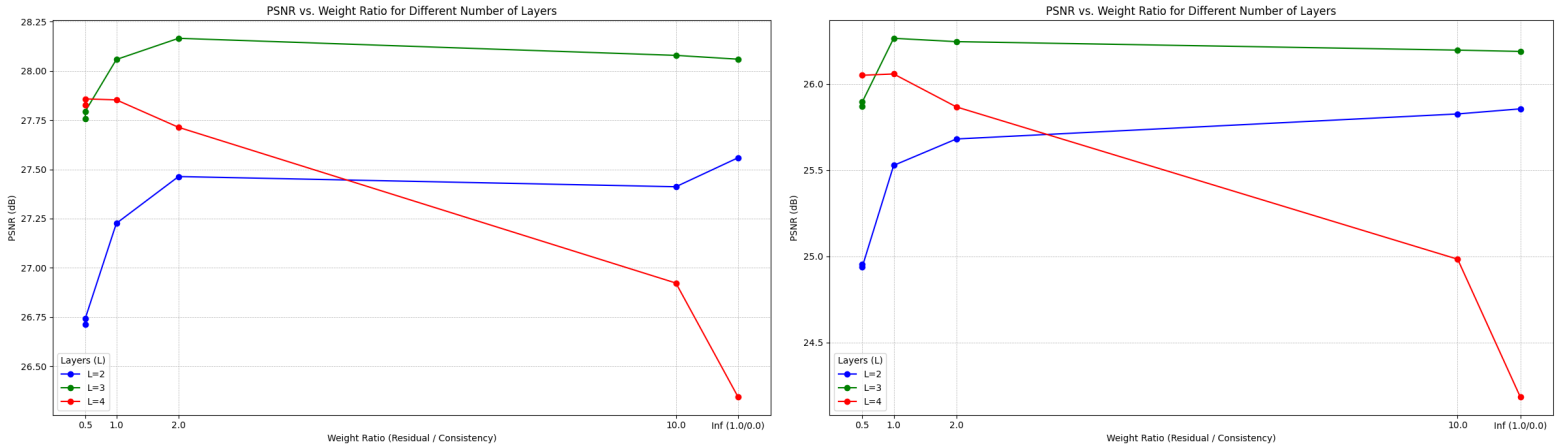
- ZS-N2N evaluated with L = 2, 3, 4 layer networks.
- Res/Cons loss ratios: 0.5, 1.0, 2.0, 10.0, ∞.
- Training: 2000 epochs per configuration, on 10 PNG images.
- Evaluation: PSNR averaged over 10 images.

Key Results and Performance



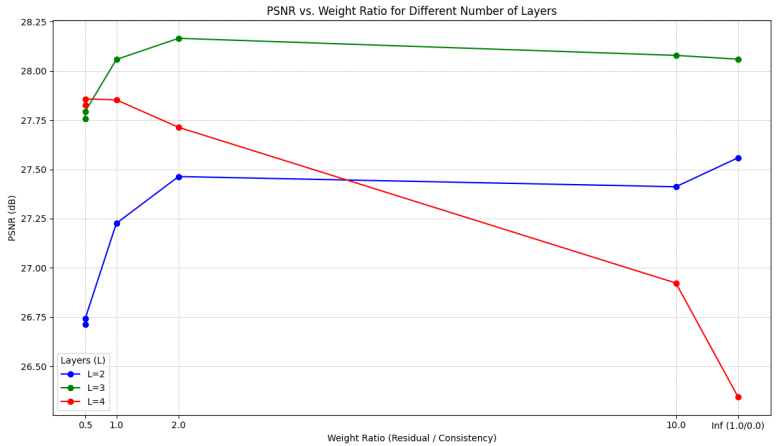
- L=3: Best performance for Res/Cons ≥ 1.0.

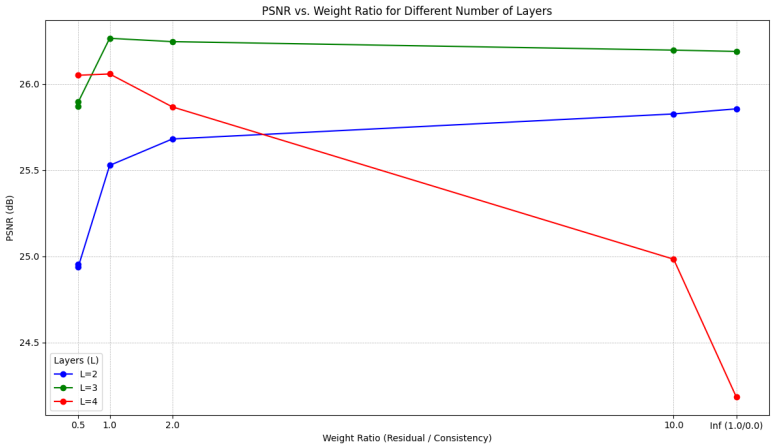
Key Results and Performance



- L=3: Best performance for Res/Cons ≥ 1.0.
- L=2: Worst, better only when no consistency loss.

Key Results and Performance





- L=3: Best performance for Res/Cons ≥ 1.0.
- L=2: Worst, better only when no consistency loss.
- L=4: Good only with low Res/Cons (e.g., 0.5), degrades at high ratios.

Insights

The results highlight a trade-off between model capacity and the need for regularization via the consistency loss:

- The $L = 3$ network strikes an effective balance, confirming the effectiveness of the model chosen in the original paper.

Insights

The results highlight a trade-off between model capacity and the need for regularization via the consistency loss:

- The $L = 3$ network strikes an effective balance, confirming the effectiveness of the model chosen in the original paper.
- The higher-capacity $L = 4$ network is prone to overfitting the downsampled noise patterns when the consistency loss (regularization) is weak. It relies heavily on this regularization to generalize effectively to the full-resolution image.

Insights

The results highlight a trade-off between model capacity and the need for regularization via the consistency loss:

- The $L = 3$ network strikes an effective balance, confirming the effectiveness of the model chosen in the original paper.
- The higher-capacity $L = 4$ network is prone to overfitting the downsampled noise patterns when the consistency loss (regularization) is weak. It relies heavily on this regularization to generalize effectively to the full-resolution image.
- The $L = 2$ network appears under-capacitated for the task, and the consistency loss constraint seems to hinder its performance rather than help it.

Progress Summary

- 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.

Progress Summary

- 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.
- Implemented the ZS-N2N method, demonstrating its capability to denoise images effectively using only a single noisy instance.

Progress Summary

- 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.
- 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) offers the best overall performance.

Progress Summary

- 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.
- 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) offers the best overall performance.
- Established an interesting trend between model performance and loss weights for different network depths.

References I

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

Dabov, Kostadin et al. (2007). "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Transactions on Image Processing* 16.8, pp. 2080–2095. DOI: 10.1109/TIP.2007.901238.

Huang, Tao et al. (2021). "Neighbor2neighbor: Self-supervised denoising from single noisy images". In: *Conference paper at CVPR 2021*. URL: https://arxiv.org/pdf/2101.02824.

References II

Lehtinen, Jaakko et al. (2018). "Noise2Noise: Learning Image Restoration without Clean Data". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 2965–2974. URL: http://proceedings.mlr.press/v80/lehtinen18a.html.

Makitalo, Markku and Alessandro Foi (2013). "Optimal inversion of the generalized Anscombe transformation for Poisson-Gaussian noise". In: *IEEE Transactions on Image Processing* 22.1, pp. 91–103. DOI: 10.1109/TIP.2012.2210725.

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

References III

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

Contributions

- **Swarnava:** Literature review of N2N, created reports and slides. Contributed to N2N implementation.
- **Debanjan:** N2N, N2C and CNN implementation with Vipul, dataset and experiment report creation. Gaussian noise model training and BM3D.
- **Omar Muhammad:** Literature review of ZS-N2N, Implemented the experiment on hyperparameter sensitivity, created the report and presentation. Presenting with Swarnava.
- **Vipul:** N2N, N2C and CNN implementation with Debanjan. Poisson noise model training and comparison with Anscombe Image Denoiser.

Thank You / Questions?

Thank You!

Questions?