PROJECT PROPOSAL/A BENCHMARK FOR IMAGE DENOISING UNDER GAUSSIAN/POISSON/SALT-AND-PEPPER NOISE

Yuxi Yang, Zihan Lyu, Yuting Yan

ABSTRACT

This project builds a comprehensive denoising benchmark to evaluate classical image denoising methods across multiple noise types and image formats. We systematically implement and compare six denoising algorithms to handle Gaussian, Poisson, and salt-and-pepper noise on both color and grayscale images. Each team member will focus on two specific methods, ensuring thorough implementation and evaluation. We assess denoising performance across multiple noise levels and structure-aware subsets (edges, textures, flat regions) using quantitative metrics including Peak Signal to Noise Ratio (PSNR), Structural Similarity (SSIM), and visual quality assessment.

***Index Terms*—** Image Denoising, Gaussian Noise, Poisson Noise, Impulse Noise, MAP estimation, Total Variation, Bilateral Filter, Median Filter, Non-local Means, Benchmark

1. INTRODUCTION

Denoising is foundational in computational imaging. It cleans up the image polluted by noise while retaining as many details and structures as possible. Noise may come from many practical factors, such as sensor limitation, insufficient light or imperfect transmission.

Over the years, many classical denoising algorithms have been proposed. Each performs well under certain conditions but also has its own limitations. For example, MAP methods with L2 or Tikhonov priors are mathematically elegant but often over-smooth fine image details. Median and bilateral filters are simple and effective, yet they struggle to preserve textures. Variational approaches such as ROF and TGV are good at maintaining edges but tend to produce blocky staircase-like artifacts. Meanwhile, Non-Local Means methods take advantage of image self-similarity to achieve high-quality results, although they require high computational cost.

In this study, we plan to systematically evaluate six classical denoising methods, including MAP with L2/Tikhonov prior, Median Filter, Bilateral Filter, Total Generalized Variation, Non Local Means, and ROF Denoising, under the types of noise they are designed to address.

We will test both color and grayscale versions of each image to better compare their strengths and limitations and to provide insights that may guide future improvements in image restoration.

2. METHODOLOGY

2.1. Pipeline

We created a set of 50 natural images from the open-source Common Object in Context (COCO) dataset. [1] and standardized them to 512x512. Due to evaluation purpose, we synthesize three families of corruption, which are Gaussian, Poisson, and salt-and-pepper with several severity levels. Then we set up two random seeds to average out sampling effects. Images are deterministically split with 10 form a validation set while another 40 form the test set.

2.2. Gaussian Noise

Gaussian noise is the most common type of noise in digital images, characterized by additive random values following a normal distribution. The noise model is $y = x + n$ where $n \sim \mathcal{N}(0, \sigma^2)$ and σ represents the noise standard deviation.

2.2.1. MAP with L2/Tikhonov Prior

[2] The Maximum A Posteriori (MAP) estimation with L2 prior is a classical approach for Gaussian noise removal:

$$\min_x \frac{1}{2}\|x - y\|_2^2 + \lambda\|x\|_2^2. \tag{1}$$

This method provides a closed-form solution and is computationally efficient, though it tends to over-smooth image details.

2.2.2. Gaussian-Adaptive Bilateral Filter (GABF)

The bilateral filter (BF) is a nonlinear, edge-preserving smoothing technique proficient in processing Gaussian noise [3]. It computes a weighted average of neighboring pixels, where the weights depend on both spatial proximity and intensity similarity. [4] [5] However, BF degrades considerably when the guidance g and noise filtering input I are identical. To address this noise sensitivity, GABF [6] ensures that g and I are nonidentical by first applying a Gaussian blur process to produce a low-pass guidance image $\bar{g}$:

$$\bar{g}(i) = \sum_j W_{i,j}^g(g) I_j, \quad W_{i,j}^g(g) = \frac{1}{K_i} \exp\left(-\frac{\|i-j\|^2}{\sigma_s^2}\right). \tag{2}$$

The Gaussian-adaptive bilateral kernel then exploits the Gaussian spatial kernel on the filtering input I while the Gaussian range kernel uses the low-pass guidance $\bar{g}$:

$$W_{i,j}^{\text{gabf}}(I, \bar{g}) = \frac{1}{K_i} \exp\left(-\frac{\|i-j\|^2}{\sigma_s^2}\right) \exp\left(-\frac{\|I_i - \bar{g}_j\|^2}{\sigma_r^2}\right), \tag{3}$$

and the filtering output is:

$$f(i) = \sum_j W_{i,j}^{\text{gabf}}(I, \bar{g}) I_j. \tag{4}$$

This approach enforces strict preservation of image edges and contours while smoothing surfaces, even when I is corrupted by different noise compositions.

Algorithm 2 showed how GABF is implemented.

Since the GABF model pipline is complex and take upto 6 hours to run, we will update the result for this denoiser in the final report.

2.2.3. Total Generalized Variation (TGV)

Total Generalized Variation (TGV) is an improvement over using Total Variation (TV) for image regularization. It penalizes the first-order gradient of the image, ∇u, and also introduces higher-order derivatives to better model the smoothness of the image.[7]

The second-order version of TGV is defined as

$$TGV_\alpha^2(x) = \min_w \ \alpha_1 \int |\nabla x - w| \, dx + \alpha_2 \int |Ew| \, dx, \quad (5)$$

where w is an auxiliary field approximating ∇u, $Ew = \frac{1}{2}(\nabla w + \nabla w^T)$ is the symmetrized gradient, and $\alpha_1, \alpha_2 > 0$ are weights balancing first and second order smoothness.

$$\min_x \ \tfrac{1}{2}\|x - y\|_2^2 + \lambda TGV_\alpha^2(x). \quad (6)$$

Assuming the noise is Gaussian,

$$\min_x \ \tfrac{1}{2}\|x - y\|_2^2 + \lambda TGV_\alpha^2(x). \quad (7)$$

We adopt the second-order TGV prior to suppress staircasing while preserving smooth transitions. With auxiliary field w, it approximates ∇x, the model becomes

$$\min_{x,w} \ \tfrac{1}{2}\|x - y\|_2^2 + \lambda\big(\alpha_1 \|\nabla x - w\|_1 + \alpha_2 \|Ew\|_1\big).$$

Set $z = (x, w)$ and define $Kz = (\nabla x - w, \ Ew)$. Then the problem is $\min_z g(z) + f(Kz)$. We solve it using the Chambolle Pock primal dual method.[8]

The dual step projects each pixel onto ℓ_2 balls with radii α_1 and α_2 (over RGB channels). The primal x-update is

$$x^k = \frac{x^{k-1} + \tau \, \mathrm{div}(p^k) + (\tau/\lambda)\, y}{1 + \tau/\lambda}.$$

To note that, λ controls data–prior tradeoff, larger λ, stronger denoising and smoother image; smaller λ stays closer to y.
α_1 controls first-order term $\|\nabla x - w\|_1$, larger α_1 makes $w \approx \nabla x$ and keeps edges sharper.
α_2 controls second-order term $\|Ew\|_1$, larger α_2, smoother ramps, less staircasing.
τ, σ control step sizes and need to meet the requirement of $\tau\sigma\|K\|^2 < 1$. Larger values, faster speed, more instability.

2.2.4. Non Local Means (NL-Means)

Non-Local Means exploits image self-similarity to achieve high-quality denoising results:

$$NL[u](x) = \frac{1}{C(x)} \int_\Omega w(x, y)\, u(y)\, dy, \quad (8)$$

where $x \in \Omega$,

$$w(x, y) = \exp\left(-\frac{\|G_a * (u(x + \cdot) - u(y + \cdot))\|^2}{h^2}\right), \quad (9)$$

$$C(x) = \int_\Omega \exp\left(-\frac{\|G_a * (u(x + \cdot) - u(z + \cdot))\|_{(0)}^2}{h^2}\right) dz \quad (10)$$

C(x) is a normalizing constant. G_a is a Gaussian kernel and h acts as a filtering parameter.

For each pixel x, NL-means searches all other pixels y in the image. It assigns larger weights to pixels whose surrounding patches are similar to the patch around x. The denoised value at x is the weighted average of these pixels.[9]

NL-means can also be formulated as a variational problem.

$$\min_x \ \tfrac{1}{2}\|x - y\|_2^2 + \tfrac{\lambda}{2} \sum_{i,j} w_{ij}(x_i - x_j)^2, \quad (11)$$

We use the NL-means estimator. Discretizing on pixels i, j, with reference patch P_i of radius r_p and Gaussian patch kernel G_a, we write

$$\begin{aligned} d_{ij}^2 &= \big\| G_a(\sigma_a) \circ (P_i - P_j) \big\|_2^2 \\ w_{ij} &= \exp\big(-\, d_{ij}^2/h^2\big), \\ x(i) &= \frac{\sum_{j \in \mathcal{N}_{r_s}(i)} w_{ij}\, y(j)}{\sum_{j \in \mathcal{N}_{r_s}(i)} w_{ij}} \ . \end{aligned} \quad (12)$$

Here $\mathcal{N}_{r_s}(i)$ is a square search window of radius r_s centered at i. For RGB, compute d_{ij}^2 channel-wise and sum over channels. Use reflective padding to support patches near borders. This matches Algorithm 6: lines 3–6 extract P_i, P_j; line 7 computes d_{ij}^2; line 8 sets w_{ij}; lines 9 and 11 accumulate and normalize to obtain $x(i)$.

To note that,
r_p controls patch size, larger r_p, more context captured, more risks of over smoothing).
r_s controls search window, larger r_s, better matches, higher cost).
σ_a controls Gaussian patch kernel G_a inside the distance; larger σ_a, flatter spatial weighting.
h controls filtering strength, larger h, stronger denoising.

2.2.5. ROF Denoising

The Rudin-Osher-Fatemi (ROF) model is a variational approach that minimizes total variation to preserve edges while removing noise:

$$\min_x \frac{1}{2}\|x - y\|_2^2 + \lambda \|\nabla x\|_{2,1}, \quad \|\nabla x\|_{2,1} = \sum_i \sqrt{(D_x x)_i^2 + (D_y x)_i^2}. \quad (13)$$

The ROF model can be solved using the Chambolle–Pock primal–dual algorithm or ADMM splitting. The Chambolle–Pock updates are:

$$p^{k+1} = \Pi_{\|\cdot\|_2 \le \lambda}\big(p^k + \sigma_p \, \nabla \bar{x}^k\big), \quad (14)$$

$$x^{k+1} = \frac{x^k + \tau\left(\mathrm{div}\, p^{k+1} + \frac{1}{\sigma^2} y\right)}{1 + \frac{\tau}{\sigma^2}}, \quad (15)$$

$$\bar{x}^{k+1} = x^{k+1} + \theta\,(x^{k+1} - x^k), \qquad \theta \in [0, 1]. \quad (16)$$

For RGB images, use vectorial TV:

$$\big\|\nabla x\big\|_{2,1} = \sum_i \sqrt{\sum_{c \in \{R,G,B\}} \big\|\nabla x_i^{(c)}\big\|_2^2}. \quad (17)$$

2.3. Poisson Noise

Poisson noise, also known as shot noise, is signal-dependent and follows a Poisson distribution. It commonly occurs in low-light imaging and medical imaging. The noise model is $y_i \sim \mathrm{Poisson}(x_i)$ where $x_i > 0$ represents the true intensity value.

2.3.1. MAP Extensions for Poisson Noise

[10]
Variance-stabilizing transform (VST) + Gaussian MAP: This approach transforms the Poisson data to approximate a Gaussian distribution, then applies standard Gaussian denoising methods.
Direct Poisson MAP with L2 prior: Use the exact Poisson negative log-likelihood and an L2 prior. A simple and very effective solver is proximal gradient with a closed-form pixelwise prox for the Poisson data term. Let $g(x) = \sum_i (x_i - y_i \log x_i)$ and $h(x) = \frac{\lambda}{2}\|x\|_2^2$.

2.4. Salt-and-Pepper Noise

Salt-and-pepper noise, also known as impulse noise, is characterized by random pixels being set to either the minimum (0) or maximum (255) intensity values. This type of noise typically occurs due to transmission errors, faulty memory locations, or camera sensor malfunctions.

2.4.1. MAP for Salt-and-Pepper Noise

A simple MAP model uses a Laplace (L1) data term with an L2 prior:

$$\widehat{x} \;=\; \arg\min_x \mu\,\|x - y\|_1 \;+\; \frac{\lambda}{2}\|x\|_2^2 \tag{18}$$

For color images, apply channel-wise or on the luminance Y channel.

2.4.1.1. MAP for Salt-and-Pepper (L1 data, L2 prior).

Solve

$$\min_x \; F(x) = g(x)+h(x), \quad g(x) = \mu\|x-y\|_1, \qquad h(x) = \frac{\lambda}{2}\|x\|_2^2.$$

Soft-threshold (elementwise): $\mathrm{soft}(z,\theta) = \mathrm{sign}(z)\,\max(|z|-\theta,\,0)$.
Notes. - Choose $\tau \le 1/\lambda$ (Lipschitz constant of ∇h is λ). - For RGB, apply channel-wise or run on luminance $Y = 0.299R + 0.587G + 0.114B$. - Initialization $x^{(0)} = y$ is fine for impulse noise; robust to outliers due to the L1 term. - FISTA acceleration can be added; the prox form stays the same.

2.4.2. Different Applied Median Filter (DAMF)

The Progressive Switching Median Filter (PSMF) is an easy-to-implement iterative filter that progressively removes noise over several passes, first introduced by Zhou Wang and David Zhang in 1999 [11].
However, Erkan et al. 2018 indicated PSMF has limited effectiveness at high noise densities evaluated by Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM), and stated DAMF outperformed the other four models they tested. [12]. DAMF first checks if a pixel is corrupted by salt-and-pepper noise. If it is noisy, it is replaced by the median of its noise-free neighbors found within an adaptively sized window; otherwise, left unchanged.

$$\hat{Y}(i,j) = \begin{cases} Y(i,j) & \text{if } Y(i,j) \notin \{0,255\} \\ \mathrm{med}\{Y(x,y) \in W_k \setminus \{0,255\}\} & \text{if } Y(i,j) \in \{0,255\} \end{cases} \tag{19}$$

Algorithm 5 in Appendix showed how DAMF is implemented.

2.5. Evaluation Methodology

The Evaluation pipeline consists of data preparation, noise addition, tuning parameters, quantitative and qualitative evaluation.

We selected 50 images from different object categories and lighting conditions from the open-source Common Object in Context (COCO) dataset [13] for noise inducing. We standardized the input image by resizing all images to 512×512 resolution. Then split the dataset by using a deterministic seed. The images were randomly partitioned into 10 images for parameter tuning and 40 images for quantitative evaluation.

Three noise models were applied to the images: Gaussian noise at $\sigma \in \{5, 15, 25, 50\}$, Poisson noise at scaling factors $\alpha \in \{5, 20\}$ and Salt–and–pepper noise at corruption probabilities $p \in \{0.05, 0.10\}$. Each noisy image was generated under two independent random seeds 0, 1 to improve robustness.

All denoising methods were tuned on a validation set. For each noise type and level, every parameter was evaluated across 10 validation images and two noise seeds. The configuration achieving the highest average PSNR was selected.

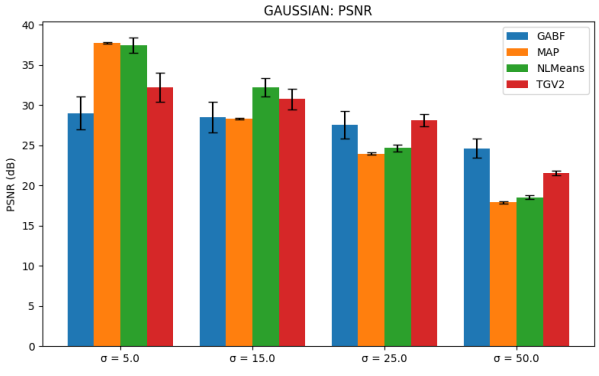
The tuned parameters are TV-L2 regularization weights $\lambda \in \{0.02, 0.05, 0.1, 0.2, 0.4\}$; GABF parameters (σ_s, σ_r) for each noise type; Non-local Means settings involving different choices of h, patch radius, search radius, and kernel width; TGV^2 parameters $(\lambda, \alpha_1, \alpha_2)$; and DAMF window sizes $\{5, 7, 9\}$ for salt–and–pepper noise.

Using the tuned parameters, we evaluated methods for their suitable noise type on 40-image test set: DAMF for salt–and–pepper noise; GABF, TGV^2, and NLMeans for Gaussian noise; MAP-L2 for both Gaussian and Poisson noise; and TV-L2 for Poisson noise. PSNR and SSIM were computed for each image, and results including noise level, and seed were recorded for analysis.

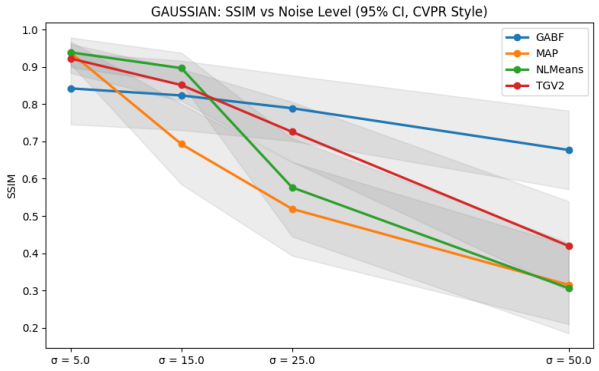
We computed a 10% trimmed mean and trimmed standard deviation for PSNR and SSIM. For each noise type we used bar charts to compare different methods at fixed noise levels, and line plots to show performance trends as noise intensity increases. PSNR figures include trimmed-mean values with error bars, while SSIM curves additionally show shaded confidence bands of approximately 95% for uncertainty.

For quantitative evaluation, we show a representative test image processed under three noise conditions, Gaussian $\sigma = 5$, salt–and–pepper $p = 0.05$, and Poisson $\alpha = 20$. For each noise type, we display noisy input with corresponding denoised outputs from the best parameters.

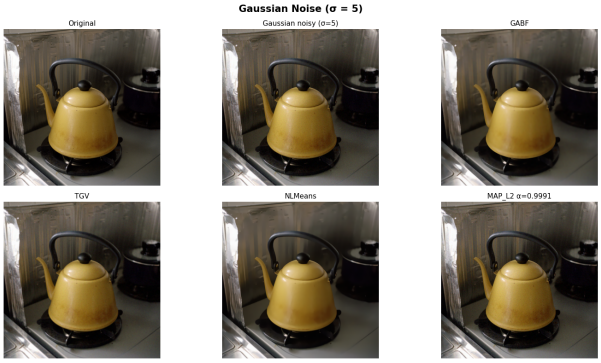
3. RESULTS



(a) PSNR Comparison for 4 Denoisers under Gaussian Noise



(b) SSIM Comparison for 4 Denoisers under Gaussian Noise



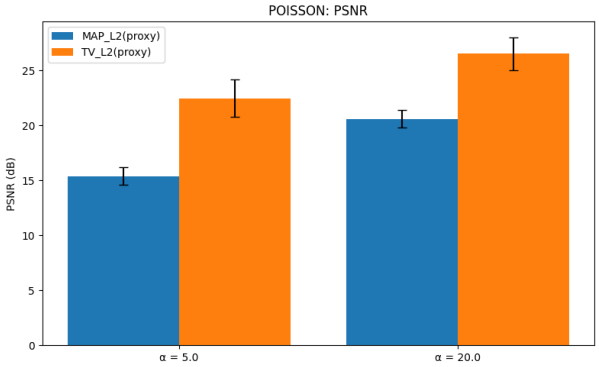
(c) Denoiser Performance under Gaussian Noise $\sigma = 5.0$

Fig. 1: Denoising Results Comparison between Ground Truth, Noisy, and Reconstructed Versions

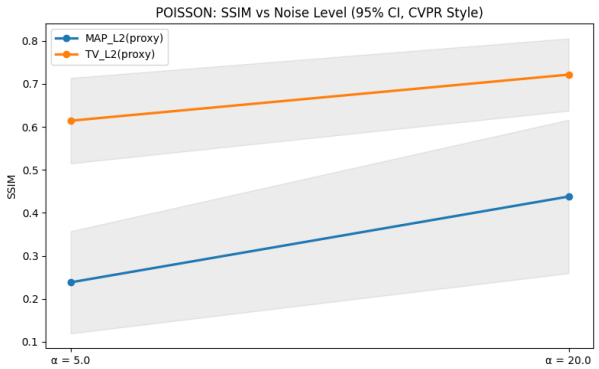
Fig. 1 shows the evaluation results of the performance of four denoising models (GABF, TGV2, NLMeans, and MAP_L2) on Gaussian noise removal for color images from the COCO 2014 test set. **Fig.1a** At the lowest noise level of $\sigma = 5.0$, all four methods perform best. The MAP model achieves the highest mean PSNR of 37.74 dB, demonstrating strong noise suppression capability. The runner-up model, NLMeans, achieves a mean PSNR of 37.46 dB, followed by TGV2 with 32.19 dB. The worst-performing model is GABF with a mean PSNR of 28.98 dB. As the noise intensity increased 10-fold ($\sigma = 50.0$), both MAP and NLMeans's performance drastically decreased, with GABF becoming the best-performing model with a mean PSNR of 24.63 dB. **Fig.1b** At noise level $\sigma = 5.0$, all four models show similar performance, with NLMeans achieving the highest mean SSIM of 0.9385 (with 95% confidence interval (CI) between 0.8988 and 0.9782), demonstrating strong ability to preserve image quality. The worst-performing model, GABF, has a mean SSIM of 0.8420 (with 95% CI between 0.7456 and 0.9384). Similarly, when noise intensity increased 10-fold ($\sigma = 50.0$), the best-performing model became GABF with a mean SSIM of 0.6769 (95% CI: 0.5716 to 0.7822), leaving NLMeans and MAP as the two worst-performing models. **Fig.1c** shows the visualized comparison of the 4 models with their best performing parameters under Gaussian noise level ($\alpha = 5.0$, seed = 1).

In general, the MAP and NLMeans models show strong ability to suppress noise while preserving image quality when noise intensity is low. On the other hand, the GABF model demonstrates greater tolerance under high noise intensity, showing graceful degradation under increased corruption. The error bars in the PSNR plot indicate varying consistency across methods and noise levels, with standard deviations ranging from approximately 0.06 dB (MAP at $\sigma = 5.0$) to 2.04 dB (GABF at $\sigma = 5.0$). The SSIM decline across noise levels shows that while MAP and NLMeans maintain high perceptual quality at low noise ($\sigma = 5.0$), with SSIM above 0.93, their performance degrades significantly at higher noise levels ($\sigma = 50.0$), dropping to 0.32 and 0.31, respectively. In contrast, GABF shows more robust performance across all noise levels, confirming its effectiveness in maintaining structural information under severe Gaussian noise corruption.

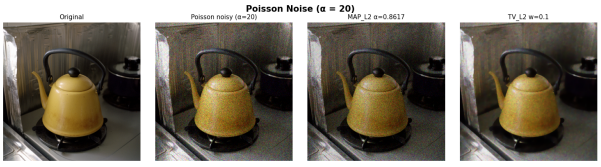
Fig. 2 presents the quantitative and qualitative evaluation of two denoising models (MAP_L2 and TV_L2) under Poisson noise. **Fig. 2a** At low noise level ($\alpha = 5.0$), the TV_L2 model outperforms MAP_L2 by a substantial margin, achieving a mean PSNR of 22.46 dB compared to MAP_L2's 15.40 dB. This indicates that TV regularization is more effective than MAP_L2 at handling the signal-dependent variance characteristic of Poisson corruption. When the noise intensity increases fourfold ($\alpha = 20.0$), both models experience a performance boost, but TV_L2 remains superior with a mean PSNR of 26.52 dB, while MAP_L2 reaches 20.59 dB. **Fig. 2b** A similar trend is observed in the SSIM comparison. At $\alpha = 5.0$, MAP_L2 achieves a low mean SSIM of 0.238 (95% CI: 0.120–0.357), whereas TV_L2 produces significantly better structural fidelity with a mean SSIM of 0.614 (95% CI: 0.515–0.714). When noise intensity increases ($\alpha = 20.0$), both models improve, with TV_L2 again outperforming MAP_L2 (0.721 vs. 0.438 mean SSIM). This suggests that TV regularization exhibits a more stable and robust behavior across Poisson noise levels. **Fig. ??** shows the visual comparison of the reconstruction results for both models under $\alpha = 20.0$. TV_L2 produces noticeably cleaner edges and fewer intensity fluctuations, while MAP_L2 exhibits blotchy artifacts typical of Poisson noise amplification. Overall, TV_L2 demonstrates consistently superior denoising performance under all Poisson noise levels, with higher PSNR, better perceptual quality (SSIM), and reduced visual artifacts. MAP_L2 shows limited robustness to signal-dependent noise, performing significantly worse, especially at low count levels ($\alpha = 5.0$). The standard deviations in PSNR range from approximately 0.80 dB (MAP_L2 at $\alpha = 20.0$) to 1.70 dB (TV_L2 at $\alpha = 5.0$), indicating higher variability for TV_L2 at lower counts but overall stronger absolute performance across the board.



(a) PSNR Comparison for 4 Denoisers under Poisson Noise



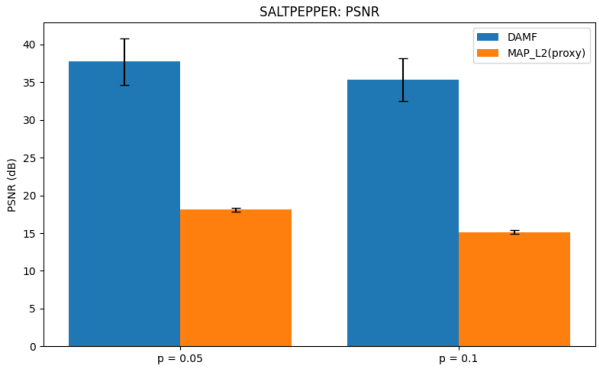
(b) SSIM Comparison for 4 Denoisers under Poisson Noise



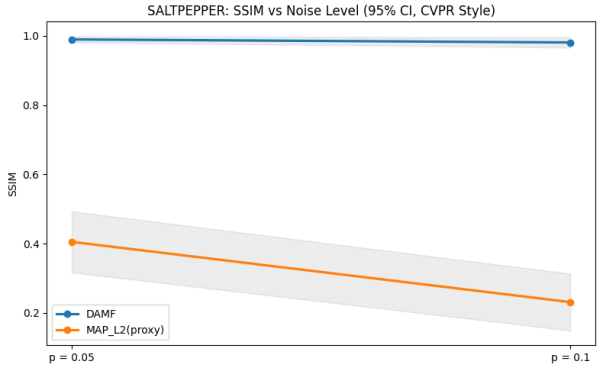
(c)

Fig. 2: Denoising Results Comparison between Ground Truth, Noisy, and Reconstructed Versions

Fig. 3 summarizes the evaluation of two models—DAMF and MAP_ L2—under salt-and-pepper impulse noise. **Fig. 3a** At a corruption probability of $p = 0.05$, DAMF significantly outperforms MAP_ L2, achieving a mean PSNR of 37.76 dB compared to MAP_ L2's 18.11 dB. This large performance gap reflects DAMF's explicit design for handling impulse noise through median-like adaptive filtering. When corruption increases to $p = 0.1$, both methods experience a drop in PSNR, but DAMF remains superior (35.34 dB vs. 15.11 dB). DAMF therefore provides strong resilience to sparse extreme-valued disruptions, while MAP_ L2 struggles due to the incompatibility between L2 priors and impulsive corruption patterns. **Fig. 3b** The SSIM comparison further highlights this difference. At $p = 0.05$, DAMF achieves an exceptionally high mean SSIM of 0.990 (95% CI: 0.981–0.998), indicating near-perfect structural preservation. MAP_ L2, in contrast, achieves 0.405 mean SSIM (95% CI: 0.317–0.493), reflecting substantial structural distortion. At $p = 0.1$, DAMF again maintains high structural fidelity (SSIM = 0.981), while MAP_ L2 collapses to 0.231 (with CI between 0.149 and 0.313). **Fig. 3c** visually demonstrates these patterns. DAMF successfully eliminates isolated noise spikes without blurring fine details, whereas MAP_ L2 produces strong artifacts and fails to remove many corrupted pixels due to the non-Gaussian nature of the noise. Across all corruption levels, DAMF is consistently the best-performing method for handling salt-and-pepper noise. MAP_ L2, by contrast, demonstrates limited denoising ability, low PSNR, and poor perceptual quality under this type of noise. The standard deviations in PSNR span from 0.25 dB (MAP_ L2) to 3.09 dB (DAMF), highlighting the higher variability but also the significantly higher absolute quality that DAMF achieves.



(a) PSNR Comparison for 4 Denoisers under Salt and Pepper Noise



(b) SSIM Comparison for 4 Denoisers under Salt and Pepper

(c)

Fig. 3: Denoising Results Comparison between Ground Truth, Noisy, and Reconstructed Versions

4. CONCLUSION

We constructed a small but rigorous denoising benchmark across three different kinds of noise on 50 COCO images with a fixed train, validation, and test split. The parameter tuning on validation only, and reporting with trimmed mean PSNR/SSIM. The results reveal clear winners by noise type and severity. Under Gaussian noise, MAP–L2 and NL-Means perform best at low corruption ($\sigma \in \{5, 15\}$), whereas GABF is the most robust at high corruption and achieves the top PSNR/SSIM at $\sigma = 50$. For Poisson noise ($\alpha \in \{5, 20\}$), TV–L2 consistently outperforms MAP–L2, delivering the highest PSNR and SSIM at both photon levels. For Salt&Pepper noise ($p \in \{0.05, 0.1\}$), the adaptive median filter (DAMF) is decisively superior, producing the largest PSNR margins and near-perfect SSIM at lower corruption. In general, this standardized protocol provides fair comparisons and actionable

guidance on method selection by noise regime.

5. REFERENCES

[1] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "COCO dataset," https://cocodataset.org, 2014, Accessed: 2025-11-03.

[2] A. N. Tikhonov and V. Y. Arsenin, *Solutions of ill-posed problems*, Winston, 1977.

[3] Carlo Tomasi and Roberto Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the IEEE International Conference on Computer Vision*, Bombay, India, 1998, IEEE, pp. 839–846.

[4] Mauro Bertero and Patrizia Boccacci, *Introduction to inverse problems in imaging*, IOP Publishing, 1998.

[5] Leonid I. Rudin, Stanley Osher, and Emad Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1–4, pp. 259–268, 1992.

[6] Bo-Hao Chen, Yi-Syuan Tseng, and Jia-Li Yin, "Gaussian-adaptive bilateral filter," *IEEE Signal Processing Letters*, vol. 27, pp. 1670–1674, 2020.

[7] Kristian Bredies, Karl Kunisch, and Thomas Pock, "Total generalized variation," *SIAM Journal on Imaging Sciences*, vol. 3, no. 3, pp. 492–526, 2010.

[8] Antonin Chambolle and Thomas Pock, "A first-order primal–dual algorithm for convex problems with applications to imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, no. 1, pp. 120–145, May 2011, Published online: 21 December 2010.

[9] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel, "A non-local algorithm for image denoising," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005, vol. 2, pp. 60–65, IEEE.

[10] Norbert Wiener, *Extrapolation, interpolation, and smoothing of stationary time series*, MIT Press, Cambridge, MA, 1964, Reprint of the 1949 edition.

[11] Zhou Wang and David Zhang, "Progressive switching median filter for the removal of impulse noise from highly corrupted images," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 1, pp. 78–80, January 1999.

[12] Uğur Erkan, Levent Gökrem, and Serdar Enginoğlu, "Different applied median filter in salt and pepper noise," *Computers & Electrical Engineering*, vol. 70, pp. 789–798, August 2018.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

6. APPENDIX

Algorithm 1 Noise–centric denoising benchmark (MAP vs. TV)

Input: CLEAN512 (50 PNGs, 512×512), Gaussian levels $\mathcal{L}_g = \{5, 15, 25, 50\}$, Poisson counts $\mathcal{L}_p = \{5, 20\}$, Salt&Pepper flips $\mathcal{L}_s = \{0.05, 0.10\}$, TV grid $\mathcal{W} = \{0.02, 0.05, 0.1, 0.2, 0.4\}$, seeds $\mathcal{S} = \{0, 1\}$.

Output: CSV with per-image metrics, aggregated tables, and plots (PSNR bars + SSIM lines).

Helpers: Normalize input images.
$x \in [0, 255] \to x_N \in [0, 1]$; $Y(x) = 0.299R + 0.587G + 0.114B$.

Noise Models on x_N:

Gaussian: $y = x + n,\ n \sim \mathcal{N}(0, (\sigma/255)^2)$

Poisson: $y \sim \mathrm{Pois}(\alpha x)/\alpha$

Salt&Pepper: $y = \begin{cases} 1 & \text{w.p. } p/2, \\ 0 & \text{w.p. } p/2, \\ x & \text{otherwise.} \end{cases}$

MAP_L2 (SURE-tuned shrink): $\hat{x} \leftarrow \alpha y,\ \alpha = \max\left(0, 1 - \frac{N(\sigma/255)^2}{\langle y, y\rangle + \varepsilon}\right)$

TV_L2: $\hat{x} \leftarrow \mathrm{TV\_denoise}(y; w)$ using Chambolle–Pock solver.

(S1) Deterministic split: Shuffle with seed 2025; 10 images → validation, remaining 40 → test. Save manifest CSV (image id, path, split).

(S2) Tune TV on validation:

for each noise type ∈ {Gaussian, Poisson, Salt&Pepper} **do**
  for each level ℓ in its level set **do**
    $w^\star \leftarrow \arg\max_{w \in \mathcal{W}} \frac{1}{|\mathrm{val}|\,|\mathcal{S}|} \sum_{x,s} \mathrm{PSNR}\left(Y(\mathrm{TV\_L2}(y(x, \ell, s); w)), Y(x)\right)$
    Cache $w^\star$ for (noise, ℓ)
  end for
end for

(S3) Evaluate on test set: Initialize results table $\mathcal{R}$.

for $x \in$ test **do**
  for $s \in \mathcal{S}$ **do**
    for Gaussian $\sigma \in \mathcal{L}_g$ **do**
      $y \leftarrow \mathrm{Gaussian}(x, \sigma, s)$
      MAP_L2: $(\hat{x}, \alpha) \leftarrow \mathrm{SURE\text{-}shrink}(y, \sigma)$
      TV_L2: $\hat{x} \leftarrow \mathrm{TV\_L2}(y; w^\star_{(\mathrm{Gaussian},\sigma)})$
      Append rows to $\mathcal{R}$
    end for
    for Poisson $\alpha \in \mathcal{L}_p$ **do**
      $y \leftarrow \mathrm{Poisson}(x, \alpha, s)$
      MAP_L2 proxy: $\hat{x} \leftarrow \mathrm{SURE\text{-}shrink}(y, \sigma{=}25)$
      TV_L2 tuned: $\hat{x} \leftarrow \mathrm{TV\_L2}(y; w^\star_{(\mathrm{Poisson},\alpha)})$
      Append rows to $\mathcal{R}$
    end for
    for Salt&Pepper $p \in \mathcal{L}_s$ **do**
      $y \leftarrow \mathrm{SaltPepper}(x, p, s)$
      MAP_L2 proxy: $\hat{x} \leftarrow \mathrm{SURE\text{-}shrink}(y, \sigma{=}25)$
      TV_L2 tuned: $\hat{x} \leftarrow \mathrm{TV\_L2}(y; w^\star_{(\mathrm{S\&P},p)})$
      Append rows to $\mathcal{R}$
    end for
  end for
end for

(S4) Aggregation & Plots: For each (noise, level, method), compute 10% trimmed mean PSNR/SSIM. Generate bar plots (PSNR + error bars) and line plots (SSIM vs. level).

Algorithm 2 GABF (Gaussian-Adaptive Bilateral Filter)

Input: Noisy image I; spatial parameter σ_s; range parameter σ_r; neighborhood $\mathcal{N}(i)$

Output: Filtered image f

▷ Compute low-pass guidance $\bar{g}$ using Gaussian blur

for all pixels i **do**
  $\bar{g}(i) \leftarrow 0$; $K_i^{(g)} \leftarrow 0$
  for all $j \in \mathcal{N}(i)$ **do**
    $w_s \leftarrow \exp\left(-\frac{\|i-j\|^2}{2\sigma_s^2}\right)$
    $\bar{g}(i) \leftarrow \bar{g}(i) + w_s \cdot I_j$
    $K_i^{(g)} \leftarrow K_i^{(g)} + w_s$
  end for
  $\bar{g}(i) \leftarrow \bar{g}(i)/K_i^{(g)}$
end for

▷ Apply bilateral filter using guidance image

for all pixels i **do**
  $K_i \leftarrow 0$; $f(i) \leftarrow 0$
  for all $j \in \mathcal{N}(i)$ **do**
    $w_s \leftarrow \exp\left(-\frac{\|i-j\|^2}{2\sigma_s^2}\right)$
    $w_r \leftarrow \exp\left(-\frac{\|I_i - \bar{g}_j\|^2}{2\sigma_r^2}\right)$
    $w \leftarrow w_s \cdot w_r$
    $f(i) \leftarrow f(i) + w \cdot I_j$

$K_i \leftarrow K_i + w$
end for
$f(i) \leftarrow f(i)/K_i$
end for

Algorithm 3 NL-Means RGB/Gray

Input: Noisy image $y \in \mathbb{R}^{H\times W\times C}$ with color $C \in \{1, 3\}$; patch radius r_p; search radius r_s; Gaussian patch kernel G_a; filtering parameter $h > 0$
Output: Denoised image $\hat{x} \in \mathbb{R}^{H\times W\times C}$

Pad y reflectively to support $r_s + r_p$
for all pixels i **do**
  Extract reference patch P_i of size $(2r_p{+}1)\times(2r_p{+}1)$
  $Z \leftarrow 0; x(i) \leftarrow 0$
  for all j in the $(2r_s{+}1) \times (2r_s{+}1)$ search window around i **do**
    Extract patch P_j
    $d^2 \leftarrow \| G_a(\sigma_a) \circ (P_i - P_j) \|_2^2$
    $w_{ij} \leftarrow \exp\big(-d^2/h^2\big)$
    $x(i) \leftarrow x(i) + w_{ij}\, y(j); \quad Z \leftarrow Z + w_{ij}$
  end for
  $x(i) \leftarrow x(i)/Z$
end for
$\hat{x} \leftarrow x$

Algorithm 4 Direct Poisson MAP (L2 prior) via Proximal Gradient

Input: Poisson data y (counts or scaled), prior weight $\lambda > 0$, stepsize $\tau \in (0,\ 1/\lambda]$, max iters K, positivity ε
Output: Denoised estimate $\hat{x}$

Init: $x^{(0)} \leftarrow \max(y, \varepsilon)$ ▷ elementwise, keeps positivity
for $k = 0, 1, \ldots, K-1$ **do**
  Gradient step on h: $v \leftarrow x^{(0)}$ **(placeholder)**
  $v \leftarrow x^{(k)} - \tau\, \nabla h\big(x^{(k)}\big) = x^{(k)} - \tau\,\lambda\, x^{(k)} = (1 - \tau\lambda)\, x^{(k)}$
  Prox step on g (elementwise):
  for each pixel i **do**
    $x_i^{(k+1)} \leftarrow \dfrac{v_i - \tau + \sqrt{(v_i - \tau)^2 + 4\tau\, y_i}}{2}$
    $x_i^{(k+1)} \leftarrow \max\big(x_i^{(k+1)},\ \varepsilon\big)$
  end for
  Stopping (optional): if $\dfrac{\|x^{(k+1)} - x^{(k)}\|_2}{\|x^{(k)}\|_2} < 10^{-4}$ then **break**
end for
Postproc: clip to display range, e.g. $x^{(K)} \leftarrow \mathrm{clip}(x^{(K)}, [0, 1])$
return $\hat{x} \leftarrow x^{(K)}$

Algorithm 5 Poisson MAP via VST + Gaussian Denoising

Input: Noisy image y with $y \sim \mathrm{Pois}(\alpha x)$, photon scale α
Output: Denoised estimate $\hat{x} \in [0, 1]$

(Normalize) Scale intensities to $[0, 1]$ if needed.
(Forward VST) ▷ Anscombe transform
$z \leftarrow 2\sqrt{\alpha\, y + \frac{3}{8}}$ ▷ If $y \in [0, 1]$, equivalently $z = 2\sqrt{y + \frac{3}{8}/\alpha}$
(Gaussian denoising on z; $\sigma \approx 1$)
if method = MAP_L2 (SURE) **then**
  $\hat{z} \leftarrow \mathrm{SURE\_shrink}(z,\ \sigma{=}1)$
else if method = TV_L2 **then**
  $w^\star \leftarrow \arg\max_{w\in\mathcal{W}} \mathrm{PSNR}\big(\mathrm{TV\_denoise}(z; w),\ z_{\text{clean}}\big)$ on validation
  $\hat{z} \leftarrow \mathrm{TV\_denoise}(z; w^\star)$ ▷ Vectorial TV for RGB
end if
(Inverse VST, bias-corrected)
$\hat{x} \leftarrow \dfrac{1}{\alpha}\left(\dfrac{\hat{z}}{2}\right)^2 - \dfrac{3}{8\alpha}$ **then** clip $\hat{x}$ to $[0, 1]$
return $\hat{x}$

Algorithm 6 TGV^2 RGB/Gray

Input: Noisy image $y \in \mathbb{R}^{H\times W\times C}$ with color $C \in \{1, 3\}$; data weight $\lambda > 0$; TGV weights $\alpha_1, \alpha_2 > 0$; step sizes $\tau, \sigma > 0$; iterations N
Output: Denoised image $\hat{x} \in \mathbb{R}^{H\times W\times C}$

$x^0 \leftarrow y; \quad w^0 \leftarrow 0 \in \mathbb{R}^{H\times W\times 2\times C}$
$p^0 \leftarrow 0 \in \mathbb{R}^{H\times W\times 2\times C}; \quad q^0 \leftarrow 0 \in \mathbb{R}^{H\times W\times 3\times C}$
$\bar{x}^0 \leftarrow x^0; \quad \bar{w}^0 \leftarrow w^0$
for $k = 1$ **to** N **do**
  $p^k \leftarrow p^{k-1} + \sigma\big(\nabla \bar{x}^{k-1} - \bar{w}^{k-1}\big)$
  $p^k \leftarrow \dfrac{p^k}{\max\big(1,\ \|p^k\|_{2,\text{color}}/\alpha_1\big)}$
  $q^k \leftarrow q^{k-1} + \sigma\, E\bar{w}^{k-1}$
  $q^k \leftarrow \dfrac{q^k}{\max\big(1,\ \|q^k\|_{2,\text{color}}/\alpha_2\big)}$
  $x_{\text{old}} \leftarrow x^{k-1}$
  $x^k \leftarrow \dfrac{x^{k-1} + \tau\, \mathrm{div}(p^k) + (\tau/\lambda)\, y}{1 + \tau/\lambda}$
  $w_{\text{old}} \leftarrow w^{k-1}$
  $w^k \leftarrow w^{k-1} + \tau\big(\mathrm{div}_E(q^k) + p^k\big)$
  $\bar{x}^k \leftarrow 2x^k - x_{\text{old}}$
  $\bar{w}^k \leftarrow 2w^k - w_{\text{old}}$
end for
$\hat{x} \leftarrow x^N$

Note: Set $C = 1$ for grayscale; set $C = 3$ to denoise the three RGB channels jointly.

Algorithm 7 Salt&Pepper MAP

Input: Noisy image y, weights $\mu > 0$, $\lambda > 0$, step $\tau \in (0,\ 1/\lambda]$, max iters K, clamp range $[a, b]$ (e.g. $[0, 1]$), tolerance ε
Output: Denoised estimate $\hat{x}$

Init: $x^{(0)} \leftarrow y$
for $k = 0, 1, \ldots, K-1$ **do**
  Gradient step on h: $v \leftarrow x^{(k)} - \tau\,\nabla h(x^{(k)}) = x^{(k)} - \tau\,\lambda\,x^{(k)} = (1 - \tau\lambda)\,x^{(k)}$
  Prox step on g (shifted soft-threshold):

$$x^{(k+1)} \ \leftarrow\ y \ +\ \mathrm{soft}\big(v - y,\ \tau\mu\big)$$

  Clamp (optional): $x^{(k+1)} \leftarrow \mathrm{clip}\big(x^{(k+1)}, [a, b]\big)$
  Stopping (optional): if $\dfrac{\|x^{(k+1)} - x^{(k)}\|_2}{\|x^{(k)}\|_2} < \varepsilon$ then break
end for
return $\hat{x} \leftarrow x^{(k+1)}$

Algorithm 8 DAMF (Different Applied Median Filter)

Input: Noisy image Y (uint8, values in $[0, 255]$); maximum window size $k_{\max}$ (default: 7)
Output: Denoised image $\hat{Y}$

$\hat{Y} \leftarrow Y$
for $k \in \{3, 5, 7, \ldots, k_{\max}\}$ **do** ▷ Iterate through window sizes
  for all pixels (i, j) **do**
    Convert $\hat{Y}$ to uint8 representation
    if $\hat{Y}_{\text{uint8}}(i, j) \in \{0, 255\}$ **then** ▷ Pixel is noisy
      $W_k \leftarrow$ window of size $k \times k$ centered at (i, j)
      $S \leftarrow \{\, \hat{Y}(x, y) \mid (x, y) \in W_k,\ \hat{Y}_{\text{uint8}}(x, y) \notin \{0, 255\} \,\}$
      if $|S| > 0$ **then**
        $\hat{Y}(i, j) \leftarrow \mathrm{Median}(S)$
      end if
    end if
  end for
end for
return $\hat{Y}$