From Smoothing to Transformers: How Modern Models Denoise Images?

$Anna\ Marquez^{*}, Karamine\ Ellis^{*}, Kaiser\ Hamid^{*}$
(* denotes equal contributions)

What is Image De-noising?

The process with which we reconstruct a signal from a noisy one.

<u>Goal of Image De-noising:</u>

- ❑ Remove noise
- ❑ Preserve useful information

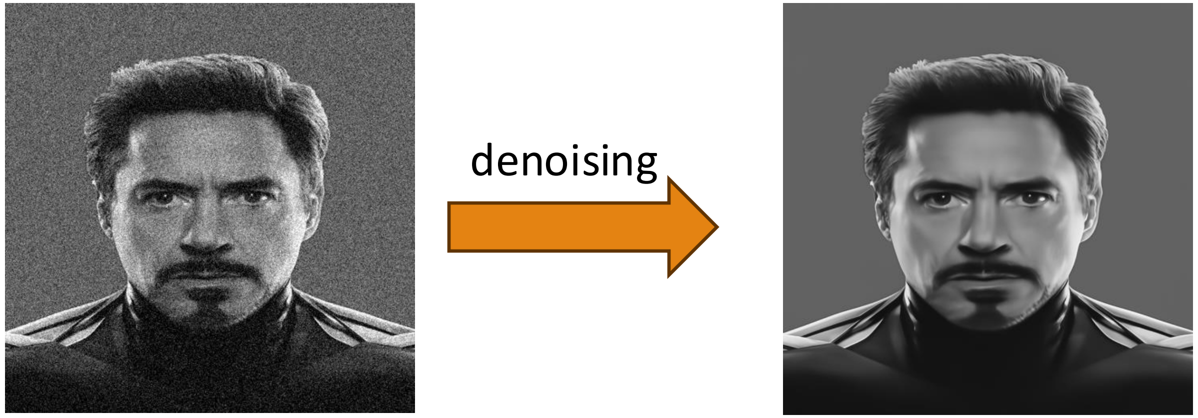


Fig. 1: Noisy image restored by Restormer [1]

Problem Formulation

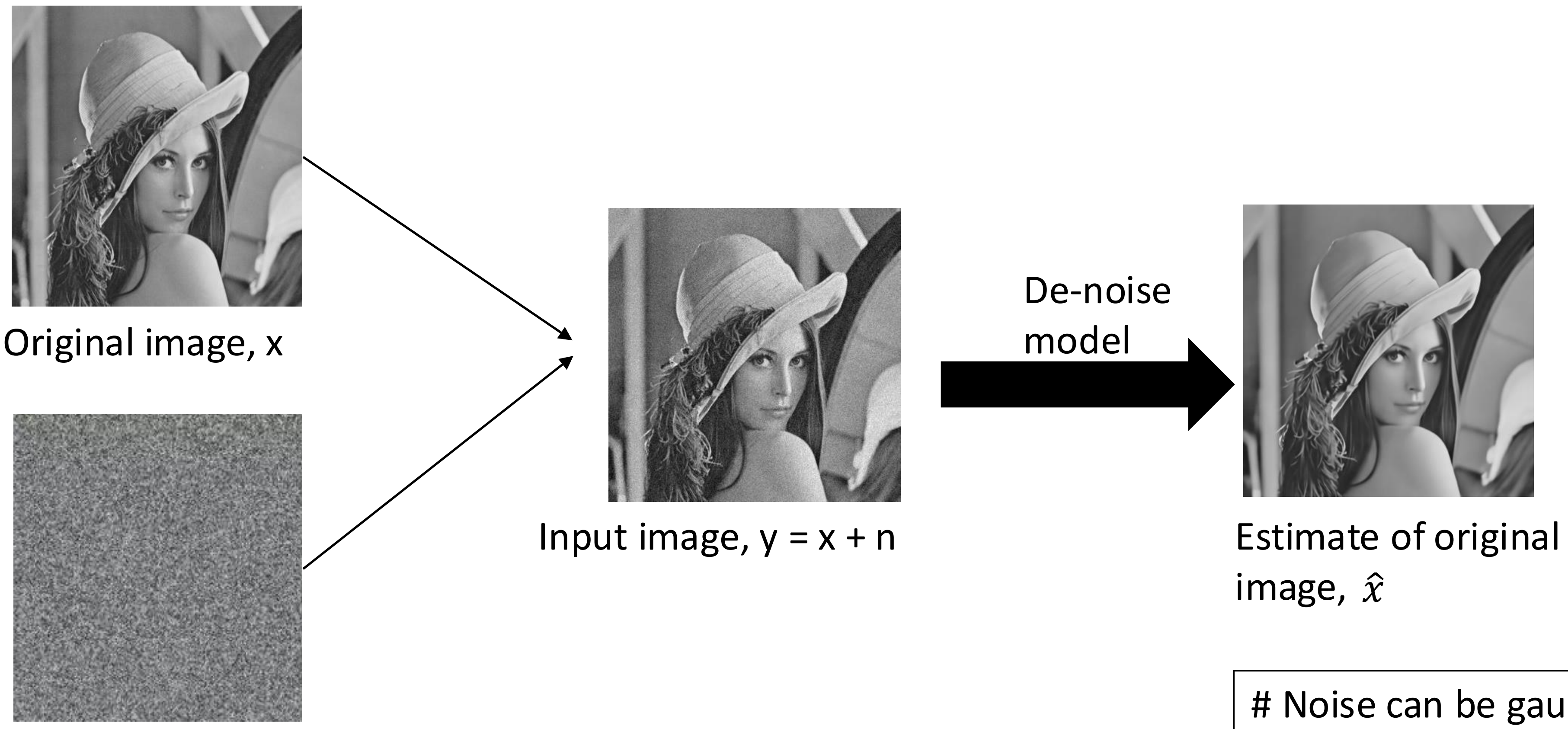


Fig. 2: Problem definition flow-chart

Noise can be gaussian and salt & pepper

How can we denoise image?

Smoothing

- Mean filter – Takes average of all surrounding pixels which replaces that pixel.
- Blurs everything slightly, so good for gaussian noise but removes fine details and edge clarity

Median

- Replaces pixel with median of surrounding pixels
- Good for salt and pepper, preserves edges

Adaptive (Lee)

- Uses variance to decide how much smoothing to give
  - Low variance = strong smoothing
  - High variance = slight smoothing
- Reduces noise in noisy areas but keeps structure and details of image

How can we denoise image? (Cont.)

DnCNN (CVPR’17)

- Converts the image into feature space using many 3×3 convolution layers (64 channels).
- Applies Batch Normalization (BN) to stabilize feature distributions.
- Uses ReLU to suppress weak/noisy activations and keep meaningful edges.
- Learns the residual noise instead of the clean image:
  Clean = Noisy – PredictedNoise

Restormer (CVPR’22)

- Converts the image into deep feature space (48 → 96 → 192 → 384 channels).
- Uses self-attention so each pixel compares itself with all other pixels (captures long-range dependencies).
- Learns global structure:
  “Pixels that behave consistently are structure; random ones are noise.”
- Uses GDFN (Gated DConv Feed-Forward Network) to enhance important features and suppress noisy ones.

Training recipe

Methods	Dataset Requirements	Training time	Hardware Requirements
Smoothing	None	None	CPU
Median	None	None	CPU
Adaptive	None	None	CPU
DnCNN (CVPR’17)	BSD400 dataset. Trained per noise σ = 15, 25, 50	~1 day on GPU (NVIDIA Titan X)	GPU
Restormer (CVPR’22)	Large-scale datasets depending on task: **SIDD, DND, GoPro, HIDE, RealBlur, Rain13K** etc.	Multi-A100 training for very long iterations	Multi-GPU

Benchmarking dataset

BDS-68 dataset:

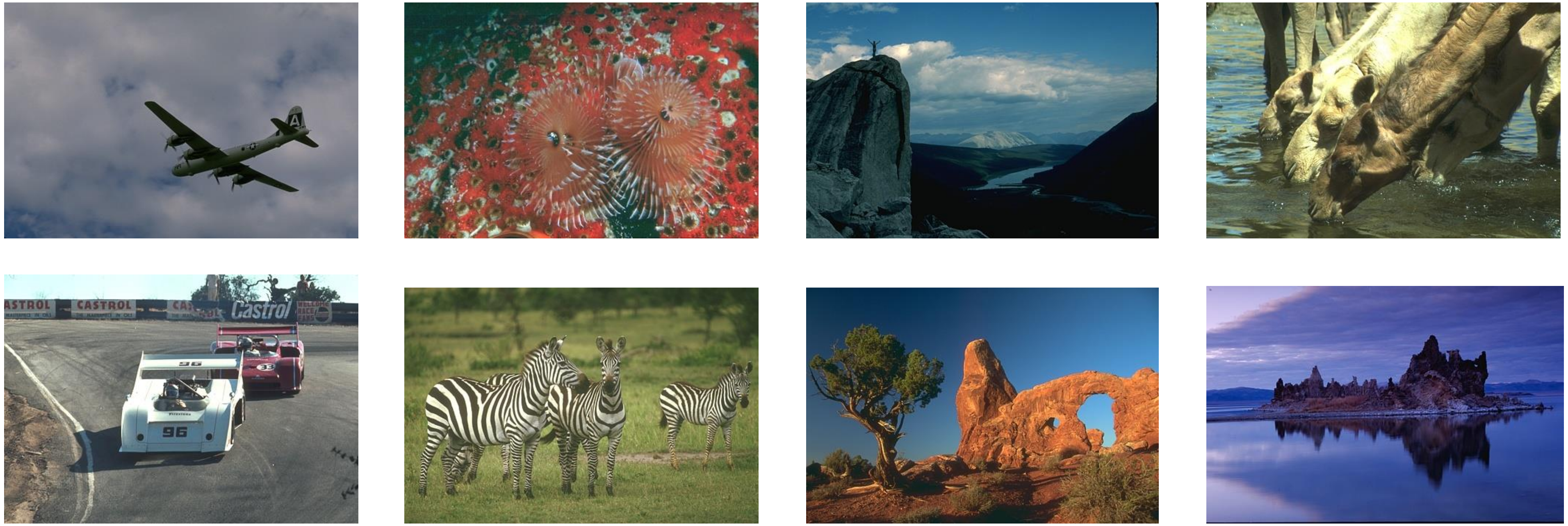


Fig. 3: Few examples from BDS-68 dataset

Benchmarking dataset (cont.)

Our own supplied images for analysis:

Fig. 4: Our supplied image set

Evaluation Criteria

PSNR (Peak Signal-to-Noise Ratio)
Measures pixel-wise reconstruction quality.

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

$$\mathrm{MSE} = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(x_{ij} - \hat{x}_{ij})^2$$

Where:

- x = original clean image
- $\hat{x}$ = denoised image
- MAX_I = maximum pixel value (255 for 8-bit images)
- Higher PSNR = better pixel accuracy

SSIM (Structural Similarity Index)
Measures structural, contrast, and luminance similarity.

$$\mathrm{SSIM}(x, \hat{x}) = \frac{(2\mu_x\mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)}$$

Where:

- $\mu_x,\ \mu_{\hat{x}}$ = mean intensity
- $\sigma_x^2,\ \sigma_{\hat{x}}^2$ = variance
- $\sigma_{x\hat{x}}$ = covariance
- $C_1,\ C_2$ = constants for numerical stability
- Higher SSIM = better structural preservation

Quantitative Analysis

Method	PSNR↑		SSIM↑		Runtime (ms)↓
	Gauss	SP	Gauss	SP	Avg.
Smoothing	23.99	24.29	0.925	0.929	371.0
Median	24.23	**29.06**	0.929	**0.973**	**10.2**
Adaptive	24.38	23.43	0.923	0.927	1635.7
DnCNN [2]	**28.33**	21.67	**0.969**	0.901	181.5
Restormer [1]	26.13	20.20	0.948	0.867	3827.7

Table 1: Benchmarking on BDS-68 dataset

Quantitative Analysis (cont.)

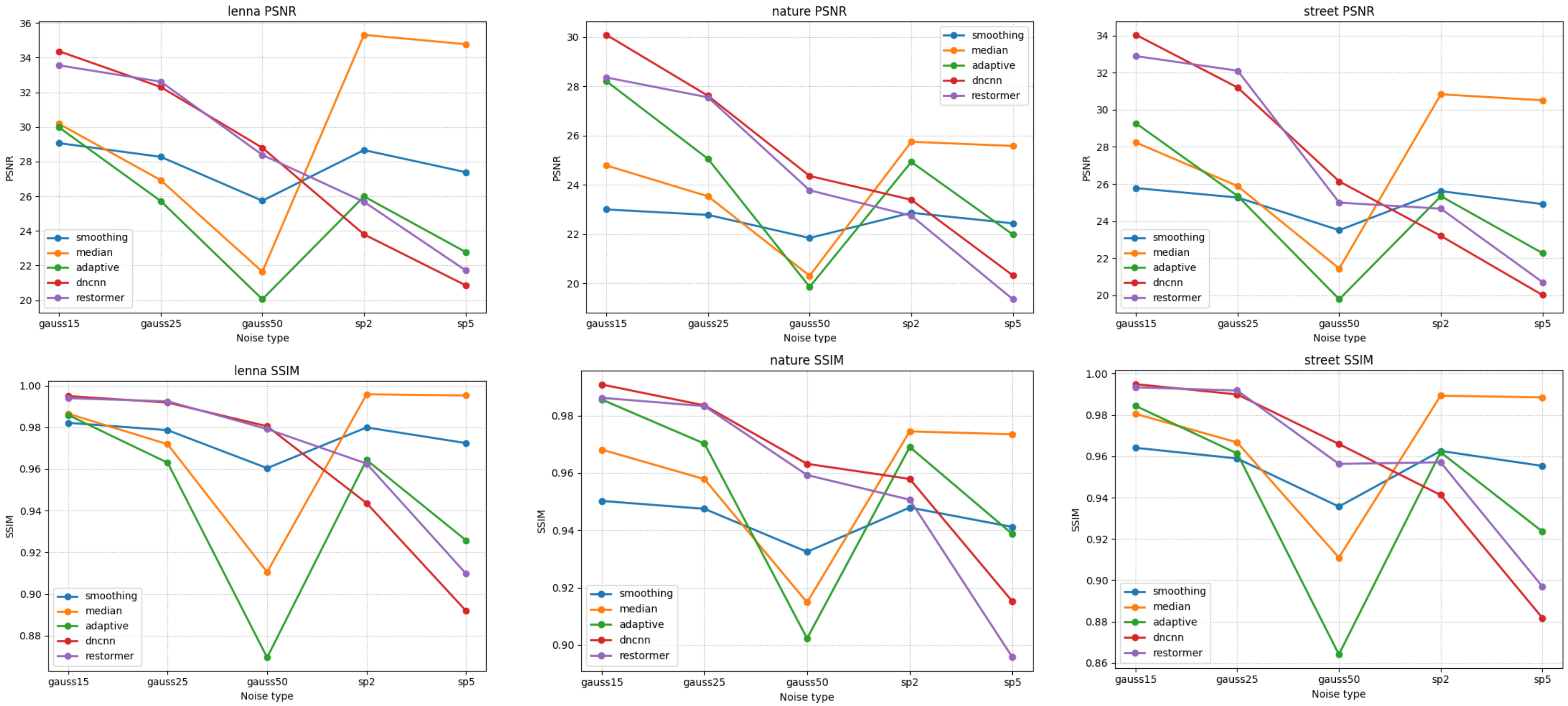
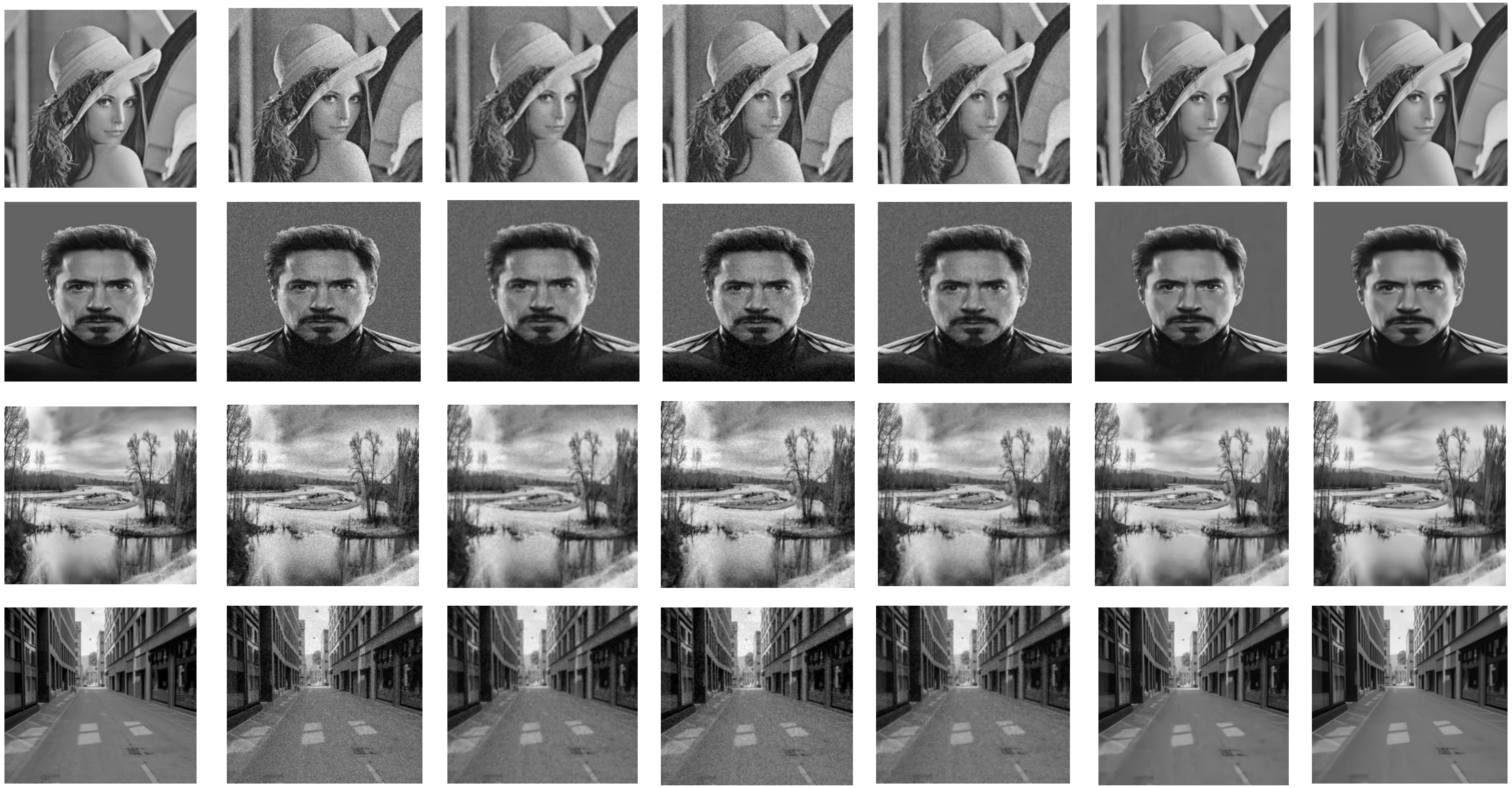


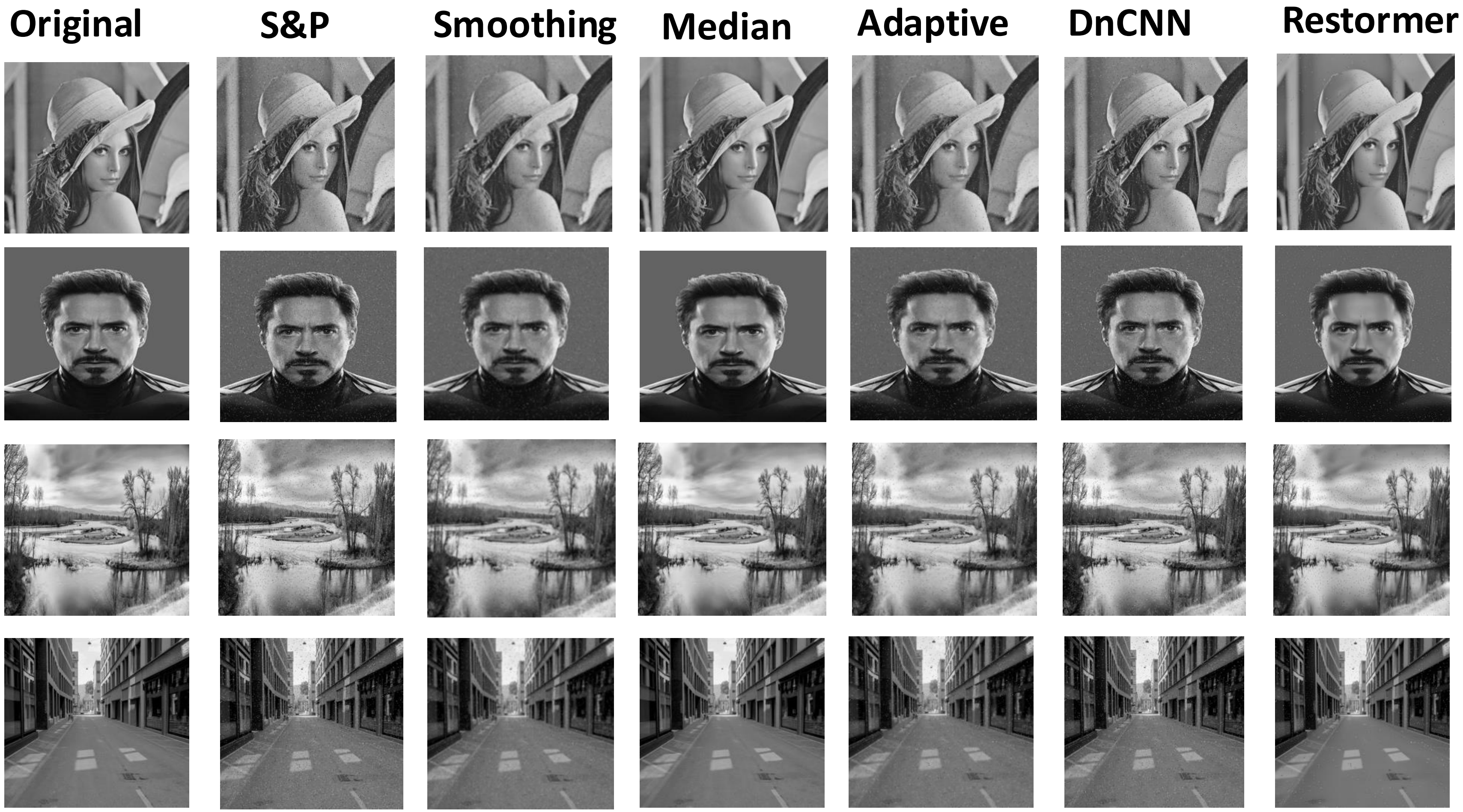
Fig. 5: Trend of PSNR and SSIM on our supplied images

Qualitative Analysis

Original **Gaussian** **Smoothing** **Median** **Adaptive** **DnCNN** **Restormer**



Qualitative Analysis (cont.)



Conclusions

• Performance drops for all methods as Gaussian noise increases.
• Deep models excel on Gaussian noise; classical median filter dominates SP noise.
• Nature image is hardest because textured regions amplify noise effects.
• DnCNN outperforms Restormer here because we used a single blind Restormer checkpoint.
• Restormer's reported SOTA results rely on noise-specific training; our setup is intentionally blind.
• Best practical strategy:
– If noise level known → Restormer (per-noise).
– If noise level unknown → DnCNN.
– For SP noise → Median filter.

Thank you! Any Questions?

Scan here for references and GitHub link:

Requirements

- ✓ Python 3.9.0
- ✓ PyTorch 2.8.0+cpu
- ✓ NumPy 2.0.2
- ✓ Pillow (PIL): 1.13.1
- ✓ Matplotlib: 3.9.4

Noise example

Original Gaussian 15 Gaussian 25 Gaussian 50 S&P 2% S&P 5%