CleanVision: Real-Time Denoising and Restoration of Noisy Images

Deep Learning Based Image Restoration Project

Team Members:

Abaha Mondal
Tapojit Bhattacharjee
Rajesh Misra

29.04.2025

Supervisor: Br. Bhaswarachaitanya (Tamal Maharaj)

Department of Computer Science
Ramakrishna Mission Vivekananda Educational and Research Institute, Belur, Howrah

Abstract

- Noise distorts critical features during image capture and transmission, reducing interpretation accuracy.
- **CleanVision** is a deep learning system for real-time denoising and restoration of high-resolution images.
- A **DnCNN** model was trained on synthetic degradations:
  - Gaussian noise,
  - Salt-and-Pepper noise,
  - Speckle noise,
  - Scribble marks.
- The **DIV2K dataset** enabled effective training across diverse scenes and textures.
- A real-time web app was deployed to deliver instant denoising for uploaded images.
- CleanVision bridges research and application, enabling impact in healthcare, security, and autonomous systems.

Introduction to Image Denoising

- Denoising is the first and critical step in image analysis workflows.
- Noise distorts visual content, affecting tasks like segmentation, detection, and recognition.
- Major noise sources:
  - Sensor limitations (pixel variations),
  - Environmental interference (fog, dust, low light),
  - Compression artifacts (JPEG).
- Without denoising, AI algorithms struggle, leading to misclassifications.
- Effective denoising improves both human and machine perception, enabling reliable computer vision applications.

Why Denoising Matters

- **Medical Imaging:** Noisy CT/MRI scans can obscure tiny tumors or fractures, affecting critical decisions.
- **Autonomous Vehicles:** Noisy sensors could lead to wrong object detection, risking accidents.
- **Surveillance and Security:** Noisy low-light footage hinders face recognition, license plate ID, and behavior analysis.
- **Scientific Research:** In astronomy, noise from atmospheric disturbances can obscure faint signals from distant stars.
- Denoising is crucial for health, safety, security, and scientific discovery — enabling precision in data-driven fields.

Applications of Image Denoising

- **Medical Imaging:** Low-dose imaging techniques introduce noise, which denoising algorithms mitigate, preserving patient safety.
- **Consumer Photography:** Smartphone cameras in low-light conditions produce noisy images; denoising enhances night-mode photography, improving clarity without hardware upgrades.
- **Video Surveillance:** Public security systems benefit from denoising, improving clarity for crime prevention and legal analysis.
- **Astronomy:** Denoising is crucial for detecting distant celestial bodies hidden in atmospheric noise.
- **Autonomous Driving:** Denoised images enable accurate lane detection, obstacle avoidance, and navigation, even in adverse conditions.

Project Objectives

- **Study Various Types of Noise:** Analyze real-world and synthetic noise patterns (Gaussian, Salt-and-Pepper, Speckle, Scribbles) and their effects on image structures.
- **Implement DnCNN Model:** Build a Denoising Convolutional Neural Network (DnCNN) with techniques like Batch Normalization and Residual Learning for stability and accuracy.
- **Quantitative Evaluation:** Measure denoising quality using PSNR and SSIM, benchmarking against ground-truth clean images.
- **Real-time Web Deployment:** Create a user-friendly web interface to allow instant denoising of uploaded images, showcasing real-world applicability.
- Each objective contributes to building a complete, functional denoising system ready for practical use.

Traditional Denoising Techniques

- **Mean Filter:** Replaces each pixel with the average of neighbors. Effective for minor noise but blurs edges.
- **Median Filter:** Replaces each pixel with the median of neighbors. Robust against Salt-and-Pepper noise, better edge preservation.
- **Gaussian Filter:** Applies a Gaussian-weighted average. Reduces noise without flattening edges, but slight blurring occurs with larger filters.
- **Common Issue:** All traditional filters rely on local averaging, reducing noise at the cost of losing important details.

Limitations of Traditional Filters

- **Blurring of Fine Structures:** Traditional filters smooth both noise and important details, leading to loss of critical visual information.
- **Fixed Behavior:** Filters operate with static rules and do not adapt based on context, noise intensity, or image type (e.g., medical scans vs. photos).
- **Failure under Mixed Noise:** Filters assume a single noise model and struggle with composite noise types (e.g., Gaussian + Salt-and-Pepper + Motion blur).
- Data-driven learning methods, which adapt and specialize, are essential for solving complex real-world denoising problems.

Deep Learning for Denoising

- **Shift to Learning-Based Approaches:** Deep learning models learn from real data, rather than relying on manual rules.
- **Data-Driven Intelligence:** CNNs learn hierarchical features (edges to textures) to differentiate noise from real details.
- **CNNs Handle Variability:** The model adapts to complex noise, lighting, and textures through diverse training data.
- **State-of-the-Art Performance:** Deep learning, especially DnCNN, outperforms traditional filters in PSNR, SSIM, and human perception.
- **DnCNN:** A pioneering architecture using residual learning to predict noise, highly effective for various types of noise.
- Deep learning shifts the paradigm from static filtering to intelligent, context-aware denoising.

Dataset Overview — DIV2K

- **High-Resolution Images:** Most images are 2040×1080 pixels or higher, capturing fine textures and details.
- **Diverse Content:** Includes urban, rural, architectural, natural scenes, people, animals, and more.
- **Variety of Conditions:** Daylight, night scenes, different weather, and viewpoints ensure wide generalization.
- **Realistic Challenges:** Complex backgrounds, shadows, and reflective surfaces prepare the model for real-world tasks.
- **Dataset Split:**
  - 800 images for training,
  - 100 images for validation,
  - 100 images for testing.
- DIV2K is ideal for training denoising models like DnCNN, preserving fine details while handling diverse noise types.

Sample Images from DIV2K

- **Urban Landscapes:** Skyscrapers, streets, traffic scenes — rich in sharp edges and small patterns.
- **Natural Landscapes:** Forests, oceans, mountains — preserving textures like leaves, water ripples, and snowflakes.
- **Portraits and People:** Faces and clothing textures — critical to retain details like eyes, wrinkles, and expressions.
- **Textures and Fine Details:** Objects like brick walls, woven fabrics, and grass fields — presenting frequency-domain challenges.
- **Importance:** Exposure to diverse content trains DnCNN to generalize, preventing overfitting to specific noise types, scenes, or textures.

Types of Noise Simulated

- **Gaussian Noise:** Random pixel variations (mean = 0, variance ^{2}). Common in low-light sensors and transmission interference.
- **Salt and Pepper Noise:** Random black/white pixels simulating dead pixels or data loss. Destructive to visual continuity.
- **Speckle Noise:** Multiplicative noise, common in medical ultrasound and radar imaging, where pixel intensities fluctuate.
- **Scribble/Cross Marks:** Manually drawn lines simulating scratches or document damage, adding high-frequency noise.
- **Why Simulate Multiple Noise Types?** Real-world images face mixed noise. A good model must handle overlapping noise patterns while preserving details.
- CleanVision was trained on complex, mixed-noise conditions to ensure robustness.

Noise Application Strategy

- **Multiple Noises:** For each clean image, randomly apply 1–3 types of noise (e.g., Gaussian + salt-and-pepper) to simulate real-world conditions.
- **Random Parameters:** Vary Gaussian noise severity (values) and randomize speckle and salt-pepper densities per image.
- **Scribble Injection:** Randomly draw 3–6 black lines of varying length, thickness, and orientation, sometimes crossing critical areas like faces or text.
- **Resulting Dataset:** No two noisy images are identical, forcing the model to generalize and avoid memorization.
- **Importance:** This setup ensures CleanVision can handle diverse, unpredictable real-world noise, not just ideal cases.

Sample Noisy Images

- **Visual Characteristics:**
  - Salt and pepper: Random white/black spots.
  - Gaussian noise: Fine-grain disturbances across surfaces.
  - Speckle noise: Ripple-like distortions in smooth areas.
  - Scribble marks: Harsh black scratches disrupting continuity.
- **Severity:**
  - Mildly corrupted images (PSNR 30–35 dB).
  - Severely degraded images (PSNR $<$20 dB), almost unrecognizable without denoising.
- **Importance:** Training on severely degraded images enables CleanVision to learn advanced restoration techniques and handle even heavily corrupted images while preserving structure and texture.

Introduction to DnCNN

- **Background:** Traditional CNNs predicted the clean image, but this is complex due to diverse structures (textures, edges).
- **DnCNN Innovation (2017):** Proposed residual learning — predicting the noise component instead of the clean image. The clean image is derived by subtracting predicted noise from the noisy input.
- **Advantages:**
  - Simplifies learning — noise has simpler patterns.
  - Faster convergence — focuses on noise features.
  - Better generalization — adapts to varying noise and image types.
- **Impact:** DnCNN set a benchmark, advancing denoising performance and inspiring future models. In CleanVision, DnCNN enabled efficient, intelligent denoising.

DnCNN Architecture

- **First Layer:** Convolutional layer + ReLU activation. Extracts low-level features (edges, textures) from the noisy image.
- **Intermediate Layers (15–20):**
  - Convolutional layers (3×3 kernels) capture spatial features.
  - BatchNorm stabilizes training and allows higher learning rates.
  - ReLU introduces non-linearity for learning complex mappings.
- **Final Layer:** Convolutional layer predicting residual noise (difference between noisy and clean images).
- **Key Architectural Choices:**
  - Zero Padding: Ensures output size matches input size.
  - No Pooling: Retains fine-grained spatial information.
- **Importance:** This simple design enables DnCNN to extract hierarchical noise patterns while preserving image structure and resolution.

Benefits of Residual Learning

- **Simpler Target Distribution:** Noise has simpler patterns (e.g., random high-frequency signals) than complex natural images.
- **Reduced Training Difficulty:** Predicting noise focuses the model on removing artifacts, rather than reconstructing entire clean images.
- **Faster Convergence:** Residual learning enables stable gradients, reducing training time and achieving higher performance with fewer epochs.
- **Better Generalization:** The model learns fundamental noise properties, making it more robust to unseen noise types.
- **Real-World Example:** In CleanVision, residual learning helped handle highly corrupted images with multiple noise types, producing sharp, clean results.

Tools Used

- **Google Colab:**
  - Free GPU access (Tesla T4, P100) for large-scale training.
  - Easy integration of libraries, version control, and cloud storage (Google Drive).
- **OpenCV (cv2):**
  - Essential for reading high-resolution images, preprocessing (resize, normalize), and augmenting images with noise.
- **PyTorch:**
  - Chosen for its dynamic computation graph and ease of debugging.
  - Used `nn.Module` for model construction, `Autograd` for automatic differentiation, and GPU acceleration for faster training.

Data Preparation

- **Image Loading:**
  - Loaded DIV2K images using OpenCV, ensuring proper RGB color channel order.
- **Normalization:**
  - Rescaled pixel values from [0, 255] to [0, 1] to prevent exploding/vanishing gradients.
- **Format Conversion:**
  - Converted images from H×W×C to C×H×W format for PyTorch compatibility.
- **Patch Extraction (Optional):**
  - Divided large 2K images into smaller patches (128×128 or 256×256) for efficient mini-batch training and reduced GPU memory usage.

Noise Functions

- **add_gaussian_noise(img, mean, sigma):** Adds Gaussian noise to simulate thermal fluctuations.
  - **Parameters:** Mean () = 0, Standard Deviation () controls intensity.
- **add_salt_pepper_noise(img, prob):** Replaces pixels with black (0) or white (255), mimicking transmission errors or dust.
  - **Parameter:** prob controls noise density (e.g., prob=0.02 for 2
- **add_speckle_noise(img):** Introduces multiplicative noise, common in radar, sonar, and ultrasound.
- **add_cross_marks(img, num_lines):** Draws random black lines to simulate scratches or scanning artifacts.
  - Randomizes number, position, angle, and thickness.

Importance: Realistic and varied augmentation ensures the DnCNN model is robust to real-world, unpredictable noise.

Directory Setup

Organizing Datasets for Efficient Training and Evaluation

- **DIV2K_HR/** - High-resolution clean images from the DIV2K dataset (used for supervised learning and evaluation metrics).
- **DIV2K_Noisy/** - Noisy versions generated by applying noise functions, with filenames like `0001_noisy.png`.
- **Automation Scripts:**
  - Load and apply random noise to clean images.
  - Save noisy images with consistent naming conventions.

Importance:

- **Data Pairing:** Ensures easy noisy-clean image matching for training.
- **Error Minimization:** Avoids label mismatches.
- **Batch Loading:** Compatible with PyTorch's DataLoader for efficient training.

Model Training Details

Designing an Effective Training Process

- **Batch Size = 8:**
  - Small enough to fit GPU memory.
  - Ensures stable gradients and generalization.
- **Epochs = 20:**
  - Provides a balance between convergence and avoiding overfitting.
  - Early stopping based on validation loss.
- **Learning Rate = 0.001:**
  - Standard for Adam optimizer — allows steady learning.
- **Optimizer = Adam:**
  - Combines momentum and RMSprop for faster convergence.
- **Loss Function = MSE:**
  - Measures the squared difference between predicted and actual noise.
  - Directly correlates with PSNR for quality evaluation.

Training Loop

Training Flow of CleanVision

1. Forward Pass: Input a noisy image, output predicted noise.
2. Loss Computation: Compute MSE between predicted noise and true noise (noisy - clean).
3. Backward Pass: Use autograd for gradient calculation, backpropagate error.
4. Optimizer Step: Adam optimizer updates model weights.
5. Progress Monitoring: Log training and validation loss to detect overfitting.
6. Epoch Completion: Repeat steps 1–5 for each epoch until convergence.

Key Points: Small batch size + Adam optimizer = smoother convergence. MSE loss minimizes restoration errors.

Evaluation Metrics

Quantifying Restoration Quality

Peak Signal-to-Noise Ratio (PSNR):

- Measures the ratio between the max pixel value and the root mean squared error (RMSE) between restored and ground-truth images.
- Formula:
$$PSNR = 20 \log_{10}\left(\frac{MAX_{pixel}}{\sqrt{MSE}}\right)$$
- Higher PSNR = better restoration. PSNR $>$30 dB = good, $>$40 dB = excellent.

Evaluation Metrics (cont.)

Structural Similarity Index Measure (SSIM):

- Measures perceptual similarity based on luminance, contrast, and structure.
- Ranges from 0 (completely different) to 1 (identical).
- SSIM $>0.85 =$ high perceptual similarity.

Why Use Both?

- PSNR captures pixel-level fidelity.
- SSIM evaluates human perception.
- Together, they provide a balanced evaluation of restoration quality.

Sample Results

Noisy Inputs: Gaussian, salt-and-pepper, scribbles, speckles.

CleanVision Output:

- Sharp edges restored.
- Textures and color fidelity maintained.
- Noise removal (graininess, speckles).

Evaluation:

- PSNR: 18–25 dB (noisy) to 30–40 dB (restored).
- SSIM: $>$0.90.

Visual Comparison:

- Left: Noisy, Right: Restored, (Center: Clean).

Real-Time Web Deployment

Frontend: Simple, responsive design. Users can upload noisy images, denoise, and download restored images.

Backend: DnCNN model loaded into memory. Preprocessing, denoising, and post-processing on user requests.

Deployment Frameworks: Streamlit for rapid deployment. Flask/FastAPI + Docker for production.

Performance: Inference time for 512×512 image: < 1 second.

Impact: CleanVision transformed from a research project to a usable, shareable tool for real-world applications.

Challenges Faced

Noise Diversity Simulation:
Balancing realistic noise combinations to avoid overfitting or instability.

GPU Memory Limitations:
Training on high-res images stressed GPU memory. Solutions: reduced batch sizes, patch-based training, gradient checkpointing.

Balancing Denoising Detail:
Aggressive denoising could harm textures. Fine-tuned model depth, filters, and learning rates to preserve details.

Real-Time Inference Optimization:
Inference times reduced user experience. Optimized with model quantization, parallel batching, and lightweight front-end.

Lesson:
Overcoming challenges enhanced technical and problem-solving skills, improving project management and resource optimization.

Key Learnings

Residual Learning:
Predicting residuals simplifies tasks like image restoration and extends to deblurring, super-resolution, and compression artifact removal.

Noise Modeling:
Realistic noise simulations are critical to prevent overfitting and ensure robust real-world performance.

Hyperparameter Sensitivity:
Minor adjustments in batch size, learning rate, and normalization have significant effects on model quality.

Web Deployment:
Gained experience with Streamlit, Flask, and server setups for preparing models for end-user interaction (loading, APIs, UX).

Collaboration Project Management:
Effective task division, planning, and communication were key to successful project execution.

Future Scope

Real-World Noisy Images:
Gather noisy data (e.g., surveillance, mobile) to handle complex distortions.

Transformer Architectures:
Experiment with ViT, Swin Transformer for better denoising.

Mobile/Edge Deployment:
Optimize for mobile (INT8, TensorFlow Lite, PyTorch Mobile).

Self-Supervised Learning:
Train models without clean data (Noise2Noise, Noise2Void).

Multi-Task Models:
Combine denoising with super-resolution, color correction.

Vision:
Expand CleanVision into a multi-purpose image restoration system for diverse industries.

Conclusion

Problem:
Image noise from acquisition, transmission, or environment affects critical sectors like healthcare and surveillance.

Solution:
CleanVision, based on DnCNN, effectively removes noise while preserving key image details.

Achievements:
- Removes multiple noise types.
- Preserves edges, textures, and color.
- High performance (PSNR, SSIM).

Deployment:
Real-time web app for practical use.

Key Strengths:
- Robust against various noise types.
- Fast inference and easy deployment.

References

1. JEBUR, R.S.: Image denoising using mean filter. *Al-Salam Journal for Engineering and Technology*. 2(2), 527–532 (2023)
2. Hong, N.M., Thanh, N.C.: Distance-based mean filter for image denoising. *Association for Computing Machinery*. 78, 74–80 (2020) `https://doi.org/10.1145/3380688.3380704`
3. Shah, A., Bangash, J.I., Khan, A.W., Ahmed, I., Khan, A., Khan, A., Khan, A.: Comparative analysis of median filter and its variants for removal of impulse noise from gray scale images. *Journal of King Saud University - Computer and Information Sciences*, 34(2), 505–519 (2022)
4. Odat, A., Otair, M.A.: Image Denoising by Comprehensive Median Filter, (2015)

References

5. Piao, W., Yuan, Y., Lin., H.: A digital image denoising algorithm based on gaussian filtering and bilateral filtering., pp. 10–13 (2018)
6. Haeyer, J.P.F.D.: Gaussian filtering of images: A regularization approach (1989)
7. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. (2017)