Original Research Article

Astrophotography Image Restoration Using a Convolutional Autoencoder

Eiden Han

Brea Olinda High School, 789 Wildcat Way, Brea, CA 92821, United States

ABSTRACT

Noise is commonly present in astrophotography images and can obscure faint astronomical structures and reduce their scientific value. Traditional noise-reduction techniques often blur important details and fail to preserve fine astronomical structures. This study investigated whether a deep learning–based approach could more effectively reduce noise while preserving fine structural features in astronomical images. It was hypothesized that a U-Net-style convolutional autoencoder (CAE) trained on synthetically noised astrophotography data can serve as an effective, innovative tool to reconstruct higher quality images. To test this hypothesis, a CAE was trained on publicly available astrophotography images (107 images from the Messier catalog), and synthetic noise was added to these images to create paired training data, allowing the model to learn a direct mapping from noisy inputs to clean outputs. Across 11 test images, the mean peak signal-to-noise ratio (PSNR) increased from 11.9 dB in noisy inputs to 33.0 dB after denoising, while the mean structural similarity index (SSIM) improved from 0.28 to 0.89. The results showed that the CAE reduced the synthetic noise while preserving the most fine astronomical structure. These findings motivate further evaluation of deep learning–based denoising methods for astronomical image analysis.

Keywords: Astrophotography; Image denoising; Convolutional autoencoder; U-Net; Astronomical image processing; Noise reduction; TensorFlow

INTRODUCTION

Astrophotography specializes in capturing faint celestial objects such as galaxies, nebulae, and star clusters through long-exposure imaging (1). Beyond producing visually striking images, astrophotography is an important tool in astronomical research: it enables the study of stellar evolution by documenting star-forming regions, supports cosmology by mapping large-scale galactic structures, and contributes to exoplanet discovery through precise measurements of stellar brightness variations (2).

Modern astrophotography relies on sensitive electronic detectors such as charge-coupled devices (CCDs), which enable precise measurements of both brightness and spatial information across an image (3). Because they capture a much larger fraction of photons with high accuracy, CCDs have been widely used as modern detectors in astrophotography, allowing clear imaging of faint celestial objects (3). To capture such faint light, astronomers often use long-exposure imaging, keeping the detector active for several minutes or even hours (4). Longer exposures allow the telescope to collect more photons, which improves the signal-to-noise ratio (S/N) and reveals much fainter details in the final image (5).

Corresponding author: Eiden Han, E-mail: eidenhan44@gmail.com.

Accepted July 28, 2026
https://doi.org/10.70251/HYJR2348.44521532

High-resolution astrophotography is also essential for detecting subtle effects such as weak gravitational lensing, which is used to map the distribution of dark matter in the universe (6). Yet, despite its scientific value, astrophotography faces a fundamental obstacle: the extreme faintness of celestial signals makes images highly vulnerable to noise (5).

Because these targets emit very little light, astronomers rely on sensitive detectors like CCDs or complementary metal-oxide semiconductor (CMOS) sensors to accumulate photons over extended periods of time (7). These sensors accumulate photons over time until enough are collected to form a clear image (7). However, this process introduces several types of noise (4). Photon noise arises from the random arrival of photons at the detector (8). Readout noise results from the electronics of the sensor itself, while dark current introduces additional unwanted signals from thermal electrons (5). In addition, atmospheric scattering and light pollution further worsen image quality (4). Together, these effects make fine astronomical structures hard to visualize and make it challenging to obtain meaningful scientific information (4).

Traditional denoising techniques, such as Gaussian smoothing, median filtering, or other mathematical filters, reduce image noise by averaging or replacing pixel values based on their neighbors (9). While these methods can make images look cleaner, they treat all parts of the image equally and do not distinguish between random noise and real astronomical details that need to be preserved (9). As a result, they often blur faint but scientifically important features, such as thin nebular filaments (9). Classical calibration frames (bias, dark, and flat) can correct for systematic errors in the detector but cannot fully remove random photon noise or complex background patterns (10). In contrast, the convolutional autoencoder (CAE) developed in this project takes a fundamentally different approach. Instead of applying fixed mathematical rules, the CAE learns directly from examples of noisy and clean images (11).

Through this learning process, it identifies the statistical patterns of real astronomical structures and distinguishes them from random fluctuations caused by noise (11). This allows the model to selectively remove noise while preserving fine astrophysical features that classical filters tend to erase (12). Furthermore, unlike traditional filters that apply the same fixed operation to every image, a CAE learns its denoising behavior directly from example data rather than from a predefined rule (11). Because of this adaptability, deep-learning-based denoising has the potential to produce higher signal-to-noise ratios and sharper scientific images than conventional methods (12).

It was hypothesized that CAE can effectively remove noise from astrophotography images drawn from public datasets using NASA Messier object list as a sample model (13). To test this hypothesis, an autoencoder was developed and trained to reconstruct clean images from noisy inputs, and its performance was evaluated using both quantitative metrics, including peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) (14), and qualitative visual comparison of the original, noisy, and reconstructed images (Figure 1). By doing so, this work aims to contribute to the growing field of astroinformatics by demonstrating the potential of machine learning to enhance deep-sky images and improve the visibility of faint astronomical structures.

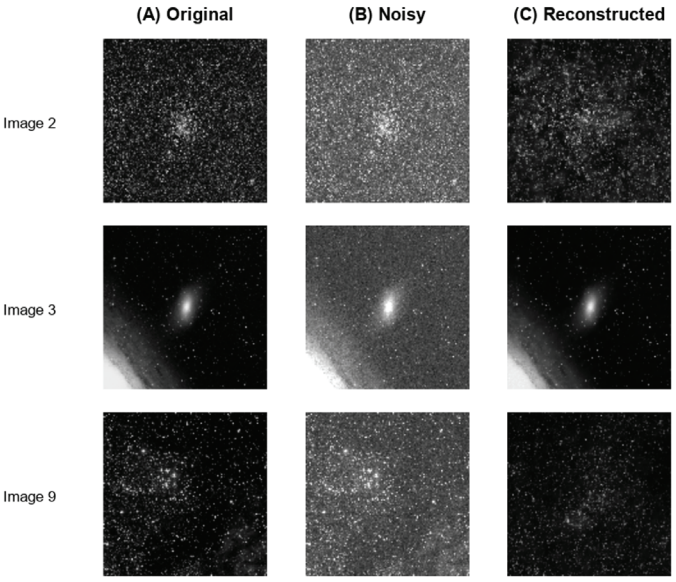

Figure 1. ***Representative denoising results produced by the U-Net-style convolutional autoencoder (CAE).*** *Three of the 11 held-out test images are shown to span the observed performance range. Columns: (A) the original clean astrophotography image, (B) the same image with added Gaussian noise (mean = 0.25, σ = 0.05), and (C) the CAE reconstruction. The rows were chosen to include both the lowest-performing cases and a high-performing case: the top row is Image 2, which had the lowest structural similarity (SSIM 0.821); the bottom row is Image 9, which had the lowest peak signal-to-noise ratio (PSNR 28.56 dB); and the middle row is Image 3, a high-performing example (PSNR 36.68 dB, SSIM 0.937). The complete comparison for all 11 test images is provided in Supplementary Figure S1, and per-image metrics are reported in Table 1.*

Table 1. Quantitative evaluation of noisy and denoised images using PSNR and SSIM. *For each of the 11 held-out test images, the table reports PSNR and SSIM for the noisy input, the Gaussian-blur baseline, and the convolutional autoencoder (CAE) output; the final row gives the mean across all 11 images. PSNR (Noisy): peak signal-to-noise ratio of the noisy image, measured relative to the clean reference image. It quantifies how much noise is present before denoising. PSNR (Gaussian) and SSIM (Gaussian): PSNR and SSIM of a Gaussian-blur baseline ($\sigma = 0.5$, chosen to maximize its PSNR) applied to the same noisy inputs. PSNR (CAE): peak signal-to-noise ratio of the image after denoising by the CAE, measured relative to the clean reference image. SSIM (Noisy): structural similarity index of the noisy image, compared to the clean reference image. Values range from 0 to 1, where values closer to 1 indicate greater structural similarity. SSIM (CAE): structural similarity index of the denoised image produced by the CAE, compared to the clean reference image.*

Image	PSNR (Noisy)	PSNR (Gaussian)	PSNR (CAE)	SSIM (Noisy)	SSIM (Gaussian)	SSIM (CAE)
1	11.89	11.83	29.71	0.443	0.415	0.921
2	11.87	11.93	29.93	0.265	0.260	0.821
3	11.96	12.03	36.68	0.218	0.309	0.937
4	11.87	11.98	35.63	0.151	0.189	0.875
5	11.86	11.98	35.50	0.197	0.250	0.902
6	11.90	11.99	35.61	0.266	0.364	0.908
7	11.97	12.03	32.78	0.229	0.255	0.838
8	11.89	11.82	30.99	0.374	0.351	0.924
9	11.93	11.76	28.56	0.529	0.480	0.952
10	11.88	11.94	33.97	0.175	0.195	0.869
11	11.89	11.96	33.36	0.175	0.206	0.858
Mean	11.9	11.93	32.98	0.275	0.298	0.891

METHODS AND MATERIALS

Dataset Preparation, Preprocessing, and Partitioning

Astrophotography images are often obtained from specialized cameras or telescope archives, commonly stored in FITS (Flexible Image Transport System) format (16). FITS is a standard in astronomy for raw image data because it can contain high dynamic range pixel values and metadata such as exposure time or telescope information (16). To use these images for neural network training, they must be converted into a numerical array format. Each pixel has a number that represents brightness. Although FITS files are commonly used in astronomy, the Messier dataset analyzed here consisted of grayscale JPEG images. In this study, images were loaded from JPEG files using OpenCV in grayscale mode and converted to floating-point arrays scaled to the [0, 1] range by dividing pixel values by 255. Once loaded, the raw images were calibrated and normalized. Because the images were stored as 8-bit grayscale JPEGs, pixel values were normalized by dividing by 255 to map intensities into the [0, 1] range. Following initial scaling, each image was further normalized using per-image min–max normalization to reduce contrast variability between samples.

Another preprocessing step required ensuring all images share the same dimensions and format. All images were resized to a uniform resolution of 128×128 pixels to ensure consistent input dimensions for the neural network. Images were converted to grayscale and represented as a single-channel input, as many deep-sky astronomical images primarily encode intensity information. Standardizing image size and format ensured compatibility with the fixed input requirements of the autoencoder architecture (17).

The dataset consisted of 107 astrophotography images, which were randomly partitioned into training, validation, and test subsets using an approximately 80%/10%/10% split. The images were divided into 85 training images, 11 validation images, and 11 test images (85 + 11 + 11 = 107), corresponding to about 79.4%, 10.3%, and 10.3% of the dataset, respectively.

The partition was generated with a fixed random seed (random_state = 42) so that the same subsets were used across all runs, and all three subsets were held fixed prior to training to ensure consistent evaluation. The 11 test images were withheld from both training and validation and were used exclusively for final quantitative evaluation.

By the end of preprocessing, a collection of clean astrophotography images of uniform size and normalized pixels was obtained. These were used as the clean targets for the denoising model.

Adding Synthetic Noise for Training

To train a denoising neural network, examples of noisy images and their clean counterparts are needed. Because paired noisy and clean images of the same faint objects were difficult to obtain, synthetic noise was added to the clean images to simulate observational noise during training.

For simplicity, a Gaussian noise model, a type of random variation in pixel intensity that follows a normal (bell-shaped) probability distribution, was used (9). Gaussian noise was added to each clean image using tf.random.normal with mean = 0.25 and standard deviation = 0.05, and the resulting pixel values were clipped to the range [0, 1]. An example clean astrophotography image and the corresponding image after synthetic Gaussian noise was added are presented (Figure 2). The left image shows a clean astrophotography frame, while the right image shows the same frame after Gaussian noise (mean = 0.25, σ = 0.05) was added (Figure 2).

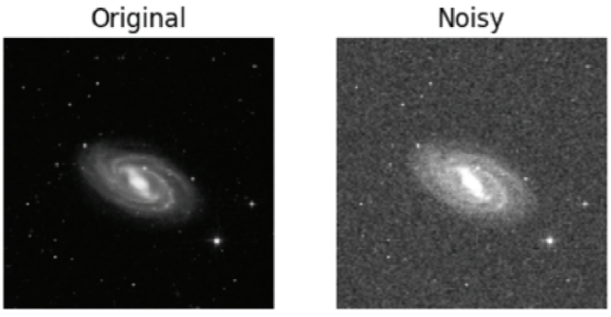

Figure 2. Original and noise-added astrophotography image. *Example grayscale astrophotography image showing the original clean input (left) and the corresponding image after synthetic Gaussian noise was added (mean = 0.25, σ = 0.05) (right). Pixel values were normalized to the [0, 1] range prior to processing.*

A nonzero mean of 0.25 was chosen, rather than the zero mean typical of a purely random noise model, so that the added perturbation would emulate a strong, uniform additive background which is analogous to sky background, light pollution, and a sensor bias/dark offset. The 0.05 standard deviation then added the small random fluctuations. Synthetic noise is a common way to model detector effects when paired noisy and clean astronomical images are unavailable (18). The mean and standard deviation used here were chosen empirically and were not calibrated to the noise statistics of any specific CCD detector, so the model is a simplification rather than a physically accurate representation of real detector noise. In practice, the mean raised the intensity of every pixel by approximately 0.25 before clipping to [0, 1], noticeably brightening each frame and compressing its dynamic range as the brightest pixels saturated at 1.0. Because a constant 0.25 offset contributes about $0.25^2 = 0.0625$ to the per-pixel squared error, whereas the 0.05 standard deviation contributes only about $0.05^2 = 0.0025$, the background offset was the dominant component of the degradation, consistent with the low mean PSNR of roughly 11.9 dB measured for the noisy inputs (Table 1). The denoising task therefore required the network to remove both this background pedestal and the finer random fluctuations, rather than random noise alone. No additional noise was added beyond this Gaussian model.

After adding noise, pixel values were clipped to the valid range (0–1) so that no pixel fell below 0 or exceeded 1. This produced a paired dataset for each original image: noisy_input, clean_target. To increase the effective size and diversity of the training data, on-the-fly data augmentation was applied using the TensorFlow dataset pipeline. During training, each clean image was randomly flipped horizontally and vertically, rotated by a random multiple of 90°, and cropped to a random 64×64-pixel patch; Gaussian noise was then added to the augmented patch using tf.random.normal, producing a new noisy–clean pair on every pass. The pipeline was repeated indefinitely and sampled using a fixed number of steps per epoch (300 steps of 16 patches, i.e., 4,800 patches per epoch), exposing the autoencoder to a wide range of image orientations, crops, and noise realizations derived from the underlying clean images. After the noisy–clean image pairs were prepared, they were used as input for training the CAE. Augmentation and synthetic-noise generation were applied only to the training set; the validation and test images were not augmented, and all reported metrics were computed on the full 128×128 test images. Because images were

assigned to the training, validation, and test subsets by the fixed split (random_state = 42) before augmentation, only training images entered the augmentation pipeline, preventing any leakage of validation or test content into training. The split seed was fixed, but the augmentation transforms and the Gaussian-noise draws were not separately seeded, so the specific patches and noise realizations differed between runs while the data partition stayed fixed. Each training example was a single random 64×64 patch taken from one source image, and the patches were shuffled and grouped into batches of 16. Because the shuffled training images were repeated across epochs, a given batch could occasionally include more than one patch originating from the same source image.

U-Net-Style Convolutional Autoencoder Architecture for Denoising

The denoising model is a CAE, a type of neural network designed to learn efficient encodings of input data and then reconstruct the data from those encodings (19). The autoencoder has two main components: an encoder and a decoder (19). The encoder compresses the input image into a smaller, abstract representation, and the decoder uses this representation to reconstruct the image back to its original dimensions (19). By training this network end-to-end to reproduce clean images from noisy inputs, the autoencoder learns to filter out noise while retaining the important structural content of the image (11).

The final model was a CAE with a U-Net-style architecture that adds skip connections between the encoder and decoder, and it contained approximately 1.95 million trainable parameters. Rather than compressing each image into a single low-dimensional vector, the encoder produced a stack of spatial feature maps that were passed to the decoder both through the bottleneck and directly through skip connections, allowing fine spatial detail to be preserved during reconstruction. Because the network was fully convolutional, it could be trained on 64×64-pixel patches and applied to full 128×128 images at evaluation time. The full layer configuration, including output dimensions and per-stage parameter counts, is summarized in Table 2.

Table 2. Architecture of the U-Net-style convolutional autoencoder (CAE). *Layer configuration of the fully convolutional U-Net-style CAE used for denoising. Each encoder and decoder stage applies two 3×3 convolutions with ReLU activation; downsampling uses 2×2 max pooling and upsampling uses 2×2 nearest-neighbor resizing followed by convolution, with skip connections concatenating the matching encoder feature maps into the decoder. Output dimensions (height × width × channels) are given for a 128×128 evaluation input; training used 64×64 patches, which halve the height and width at each stage. The model contains 1,946,305 trainable parameters (≈1.95 million).*

Stage	Layer(s)	Filters	Kernel	Activation	Output (H×W×C)	Params
Input	—	—	—	—	128×128×1	0
Encoder 1	Conv2D ×2	32	3×3	ReLU	128×128×32	9,568
	MaxPooling2D	—	2×2	—	64×64×32	0
Encoder 2	Conv2D ×2	64	3×3	ReLU	64×64×64	55,424
	MaxPooling2D	—	2×2	—	32×32×64	0
Encoder 3	Conv2D ×2	128	3×3	ReLU	32×32×128	221,440
	MaxPooling2D	—	2×2	—	16×16×128	0
Bottleneck	Conv2D ×2	256	3×3	ReLU	16×16×256	885,248
	Dropout (0.3)	—	—	—	16×16×256	0
Decoder 3	UpSampling2D + concat (Enc 3)	—	2×2	—	32×32×384	0
	Conv2D ×2	128	3×3	ReLU	32×32×128	590,080
Decoder 2	UpSampling2D + concat (Enc 2)	—	2×2	—	64×64×192	0
	Conv2D ×2	64	3×3	ReLU	64×64×64	147,584

Continued Table 2. Architecture of the U-Net-style convolutional autoencoder (CAE). *Layer configuration of the fully convolutional U-Net-style CAE used for denoising. Each encoder and decoder stage applies two 3×3 convolutions with ReLU activation; downsampling uses 2×2 max pooling and upsampling uses 2×2 nearest-neighbor resizing followed by convolution, with skip connections concatenating the matching encoder feature maps into the decoder. Output dimensions (height × width × channels) are given for a 128×128 evaluation input; training used 64×64 patches, which halve the height and width at each stage. The model contains 1,946,305 trainable parameters (≈1.95 million).*

Stage	Layer(s)	Filters	Kernel	Activation	Output (H×W×C)	Params
Decoder 1	UpSampling2D + concat (Enc 1)	—	2×2	—	128×128×96	0
	Conv2D ×2	32	3×3	ReLU	128×128×32	36,928
Output	Conv2D	1	1×1	Sigmoid	128×128×1	33
Total						**1,946,305**

Encoder: Compressing the Image

In practice, the encoder consists of a sequence of convolutional layers, each followed by a nonlinear activation such as the rectified linear unit (ReLU), which introduces nonlinearity so the network can learn more complex features, and some form of downsampling (20). Downsampling can be done by using convolutional layers with a stride greater than 1, or by inserting pooling layers (20). In the code, MaxPooling2D layers were used for downsampling to reduce the spatial resolution of the feature maps. At each downsampling stage, the number of filters was increased. For example, input images were processed through three successive convolution-and-pooling stages, halving the spatial resolution at each stage while increasing the number of feature channels from 32 to 64 to 128, and finally to 256 at the bottleneck. No global average pooling or dense bottleneck was used; the encoder output remained a spatial feature map, so positional information was retained for reconstruction.

Each convolutional stage was expected to capture increasingly abstract features, and visualizing the encoder's feature maps for a representative test image was consistent with this pattern (Figure 3): the early-layer maps retained the full 128×128 resolution and responded

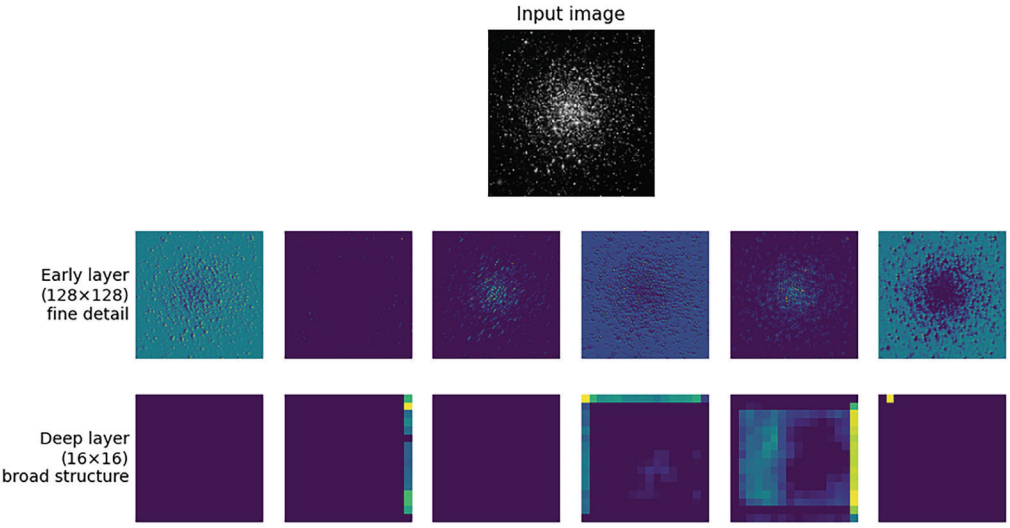

Figure 3. Encoder feature maps at different network depths. *Feature maps produced by the encoder for a representative test image (a globular star cluster). Top: the input image. Middle row: six feature maps from the first convolutional layer (128×128 resolution), which respond to fine, localized detail such as individual stars and edges. Bottom row: six feature maps from the deepest, bottleneck convolutional layer (16×16 resolution), which encode coarse, broad spatial structure; several channels are only sparsely activated, consistent with a compressed representation.*

to fine, localized detail such as individual stars and edges, whereas the deep (bottleneck) maps were coarse (16×16) and encoded broad spatial structure, with many channels sparsely activated as expected for a compressed representation (20).

After downsampling, the encoder's output formed a compact bottleneck that summarized the image at low spatial resolution while retaining spatial structure rather than collapsing it into a single latent vector (19). Although this bottleneck provided a lower-dimensional summary, the skip connections allowed finer spatial detail to reach the decoder directly rather than passing solely through the bottleneck, helping to preserve structure during reconstruction (19).

Decoder: Reconstructing the Image

In the model, the decoder mirrored the encoder, increasing spatial resolution with upsampling layers each followed by convolutional layers. At every upsampling stage, the corresponding encoder feature maps were concatenated through skip connections to recover fine spatial detail. Upsampling was performed by image resizing followed by standard convolutional layers, and the final decoder layer produced an output with the same spatial dimensions and single-channel depth as the input. A sigmoid activation at the output layer constrained reconstructed pixel values to the [0, 1] range to match the normalized image format.

Training the Autoencoder: Noisy-Clean Image Pairs

With the architecture defined, the CAE was trained using the prepared dataset of noisy and clean image pairs. Training is the process of optimizing the network's filter weights so that a noisy input image produces an output that closely matches the clean target image (21). The model was trained using supervised learning, where each noisy input image was paired with its corresponding clean image as the ground truth. In practice, the training code looped over many epochs and gradually refined the network's parameters using an optimization algorithm.

Loss Function (MSE)

To quantify how well the network is performing and give it feedback, a loss function was incorporated. In this project, the loss was mean squared error (MSE) between the network's output image and the target clean image. MSE was calculated by taking the difference between each output pixel value and the true pixel value, squaring those differences, and then averaged over all pixels. For example, if a reconstructed pixel is 0.1 brighter than its target, it adds $0.1^2 = 0.01$ to the loss. The objective of training was to minimize the MSE across all pixels and all images. MSE was chosen as a standard and suitable choice for image reconstruction tasks because it directly measures pixel-wise fidelity. A low MSE means the output image is very close to the ground truth in terms of overall pixel intensity (21).

In this equation represents the true pixel value, is the predicted pixel value produced by the model, and is the total number of pixels in the image. Lower MSE values indicate better reconstruction quality and less residual noise (21).

Training Algorithm

During training, batches of noisy input images were passed through the encoder–decoder network to generate reconstructed output images. At the start of training, the model outputs differed greatly from the corresponding clean target images due to randomly initialized weights.

Network parameters were optimized by minimizing the loss through backpropagation, which updated the convolutional filter weights across successive training batches (21). Over multiple epochs, these updates enabled the model to learn feature representations that improved reconstruction quality.

Training proceeded iteratively over successive batches of images. Each epoch consisted of 300 training steps with a batch size of 16 (4,800 augmented patches per epoch), and the training loss decreased over successive epochs. Training was performed for up to 60 epochs using the Adam optimizer (22). Early stopping was applied based on validation loss with a patience of 10 epochs, and the learning rate was adaptively reduced using a ReduceLROnPlateau scheduler with a factor of 0.5 and a patience of 4 epochs.

During training, both the training loss (MSE on the training images) and the validation loss (MSE on the held-out validation images) were monitored after each epoch. Because the validation images were not used to update the weights, the validation loss provided a check on how well the model generalized and served as the criterion for early stopping.

Visualizing Denoised Results and Performance

The denoised outputs were saved to disk as PNG files and plotted alongside the corresponding noisy inputs and clean reference images for comparison (Figure 1). For quantitative evaluation, PSNR and SSIM were computed on all 11 held-out test images using the TensorFlow functions tf.image.psnr and tf.image.ssim with max_val

= 1.0, consistent with the normalized pixel range. PSNR and SSIM were calculated for the noisy input images relative to the clean targets and again for the autoencoder outputs relative to the same clean targets, and mean values across the 11 test images were reported (23). In addition, per-image PSNR and SSIM were correlated (Pearson correlation) with the brightness (mean intensity), contrast (standard deviation), and structural complexity (variance of the Laplacian) of each clean test image to assess whether reconstruction quality varied with image content. For a baseline comparison, a Gaussian blur (σ = 0.5) was applied to each noisy test image, and PSNR and SSIM were computed between the blurred images and the same clean references used to evaluate the CAE.

RESULTS

Quantitative evaluation (PSNR and SSIM).

To evaluate denoising performance, PSNR and SSIM were computed for all 11 held-out test images. Across all eleven samples, reconstructed images consistently exhibited higher PSNR and SSIM values than the noisy inputs (Table 1). The mean PSNR increased from 11.9 dB for noisy images to 33.0 dB after denoising, an average gain of roughly 21 dB. Because PSNR is measured on a logarithmic scale, this indicates a substantial reduction in reconstruction error relative to the clean reference images. Similarly, the mean SSIM improved from 0.275 in noisy inputs to 0.891 after denoising, reflecting greater preservation of structural information (Table 1).

After denoising, PSNR values on the test set ranged from 28.6 to 36.7 dB, with eight of the eleven images above 30 dB (Table 1). The highest structural similarity was obtained for Image 9 (SSIM 0.952), and SSIM values for the majority of images exceeded 0.85, indicating that fine spatial structures were largely preserved. The lowest scores were modest rather than failures: Image 2 had the lowest SSIM (0.821) and Image 9 the lowest PSNR (28.56 dB), yet both still improved substantially over their noisy inputs. Notably, every test image showed higher PSNR and SSIM after denoising than before. However, the amount of improvement varied with image content. PSNR was negatively correlated with structural complexity, measured as the variance of the image Laplacian (Pearson r = -0.84, p = 0.001); as expected, images with more fine detail were somewhat harder to reconstruct, though all remained above 28 dB. SSIM showed a weaker, positive association with brightness (r = 0.63, p = 0.04) and no significant relationship with contrast (Table 3). Given the small sample (n = 11), these trends are indicative rather than definitive. For comparison, a Gaussian-blur baseline (σ = 0.5) applied to the same noisy test images yielded a mean PSNR of only 11.9 dB and mean SSIM of 0.30, essentially unchanged from the noisy inputs (11.9 dB, 0.275), because a linear blur cannot remove the constant background offset. The CAE therefore exceeded the Gaussian baseline by roughly 21 dB in PSNR and 0.60 in SSIM (Table 1).

In addition to the image-quality metrics, model convergence was tracked using the MSE loss during training. The training loss decreased steadily and stabilized at approximately 0.0011 (1.1×10^{-3}), while the validation loss stabilized at approximately 0.0013 (1.3×10^{-3}). The two curves remained close throughout training, the validation loss plateaued within ten epochs, and early stopping restored the best-performing weights.

Visual comparisons of reconstructed outputs

Although denoised outputs were generated for all 107 images in the dataset, Figure 1 presents three

Table 3. Correlations between per-image characteristics and reconstruction quality. *For each characteristic, r is the Pearson correlation coefficient and p is the corresponding two-sided p-value, computed across the 11 held-out test images (n = 11). Brightness was measured as the mean pixel intensity of the clean image, contrast as the standard deviation of pixel intensity, and structural complexity as the variance of the Laplacian. PSNR (r) and PSNR (p) refer to the correlation with the PSNR of the autoencoder output; SSIM (r) and SSIM (p) refer to the correlation with the SSIM of the autoencoder output. A positive r indicates that higher values of the characteristic were associated with higher reconstruction quality. Given the small sample size, these correlations are exploratory, and no correction for multiple comparisons was applied.*

Characteristic	PSNR (r)	PSNR (p)	SSIM (r)	SSIM (p)
Brightness (mean intensity)	+0.01	0.97	+0.63	0.04
Contrast (std. dev.)	-0.13	0.70	+0.32	0.34
Structural complexity (Laplacian var.)	-0.84	0.001	+0.47	0.14

representative held-out test images chosen to span the observed performance range; the complete comparison for all 11 test images is provided in Supplementary Figure S1. Each row shows the original clean image (A), the noisy input (B), and the CAE reconstruction (C). In the noisy inputs, the added Gaussian noise brightens the background and obscures faint stellar features, whereas the reconstructions show a smoother, more uniform sky background with the brighter structures preserved. Even the lowest-performing cases, Image 2 (lowest SSIM) and Image 9 (lowest PSNR), show clear noise suppression relative to their noisy inputs, while the higher-performing Image 3 retains fine structure (Table 1).

DISCUSSION

These results indicate that the CAE reduced the synthetic noise while preserving most of the faint structure that was degraded in the noisy inputs, as reflected in the higher PSNR and SSIM values (Table 1, Figure 1).

The small gap between the final training and validation losses indicates that the model generalized to the held-out data rather than memorizing the training images.

Within the scope of this study, which used a single 107-image dataset, synthetic Gaussian noise, and one network architecture, these results indicate that the learning-based model outperformed a simple linear (Gaussian-blur) filter; establishing a broader advantage of deep-learning-based denoising for astronomical data would require evaluation across additional datasets, noise types, and architectures. These results also suggest potential application in preprocessing astrophotography datasets for scientific analysis, which may improve photometric and astrometric measurements, although this was not evaluated in the present study.

The CAE's mean PSNR of 33.0 dB and SSIM of 0.891 far exceeded a Gaussian-blur baseline evaluated on the same test images (11.9 dB and 0.30; Table 1). For context, convolutional neural network (CNN)- based denoisers such as the denoising convolutional neural network (DnCNN) (12) report PSNR on the order of 29 dB on standard grayscale benchmarks (e.g., BSD68 at $\sigma = 25$). These values are not directly comparable to those reported here, however, because the datasets, noise models, and evaluation conditions differ substantially. Conceptually, the present model follows the denoising-autoencoder framework introduced by Vincent *et al.* (11).

However, some limitations and challenges were encountered during this project. The model was trained on a small dataset of only 107 images from Messier Catalog, which raised concerns about its ability to generalize to the wide variety of astrophotography conditions. An initial attempt was made to use the Hubble Legacy Archive and connect to that server to ensure it worked for every image. However, due to an unexpected error in connecting to the entire dataset, NASA Messier catalog images were used as a source, and the CAE was trained to make it applicable for other astrophotography as well. Furthermore, the noise added during training was purely synthetic Gaussian noise. While convenient for experimentation, this may not capture the full complexity of real astronomical noise. This gap means the CAE might not perform as well on genuine noisy telescope images as it did on simulated data. Due to memory limits in the Google Colab environment, reduced image resolutions and small batch sizes had to be used, which constrained the model's capacity and the resolution of outputs. Preprocessing the raw FITS files was another hurdle. Handling the high dynamic range and inconsistent image formats required careful normalization, and some image data had to be discarded or cropped. These constraints limited dataset scale and model capacity.

Another major challenge involved dataset organization. The training, validation, and test sets had to be separated carefully and fixed before training so that the model could be evaluated only on images that were not used during training. This challenge underscored a broader takeaway: proper data management and preprocessing pipelines are just as critical as the model architecture in a successful machine learning project.

Looking forward, several improvements could build upon this work to enhance denoising performance and capture more complex structures. The model could be improved by using deeper or wider convolutional layers, which would increase its capacity to learn more complex features and to preserve finer structural detail, potentially leading to more precise denoised astrophotography outputs. Furthermore, instead of relying exclusively on the 107 Messier images, larger datasets such as Sloan Digital Sky Survey (SDSS), or NASA's deep-sky databases could be used (15). Training on a larger and more diverse dataset could expose the model to a wider range of galactic structures, brightness levels, and noise conditions, potentially improving its applicability across different astrophotography data. Future work could also compare several architectures using the same training, validation, and test partitions.

CONCLUSION

In conclusion, this project demonstrated that the evaluated CAE reduced synthetic Gaussian noise in astrophotography images, improving PSNR and SSIM and yielding cleaner reconstructions. While the results are encouraging, they also serve as a starting point. Continuous development is still needed to improve the methodology and adapt it to real-world conditions. For future researchers building on this project, it may be beneficial to collaborate with the astrophotography community to obtain datasets or feedback, and to use more powerful computing resources when available so higher-resolution imagery can be processed. By following these directions, they can extend the Astrophotography Noise Remover into an even more powerful tool. Finally, it is hoped that this research continues to evolve and helps astronomers analyze faint structures in the universe with greater clarity than before.

CONFLICT OF INTEREST

The author declares that there are no conflicts of interest related to this work.

REFERENCES

1. Schröder K-P, *et al*. Astrophotography. Handbook of Practical Astronomy. Springer, 2009. ISBN: 978-3-540-76377-2, 133-173. https://doi.org/10.1007/978-3-540-76379-6_6
2. Binney, J. and Merrifield, M. "Galactic Astronomy." Princeton University Press, 1998. ISBN: 978-0-691-02565-0.
3. Romanishin W. "CCDs (Charge Coupled Devices)." An Introduction to Astronomical Photometry Using CCDs. Virginia Tech Department of Physics; 2006. p. 73-79, 119-127. Available from: https://www1.phys.vt.edu/~jhs/phys3154/CCDPhotometryBook.pdf (accessed on 2025-12-14)
4. Birney DS, *et al*. "Observational Astronomy." 2nd ed. Cambridge University Press, 2006. ISBN: 978-0-521-85370-5, 82-183.
5. Howell SB. "Characterization of charge-coupled devices." Handbook of CCD Astronomy. 2nd ed. Cambridge University Press, 2006. ISBN: 978-0-521-61762-8, 36-100.
6. Schneider P, *et al*. "Weak gravitational lensing." Gravitational Lensing: Strong, Weak and Micro. Springer, 2006. ISBN: 978-3-540-30310-7, 269-451. https://doi.org/10.1007/978-3-540-30310-7_3
7. McLean IS. Charge-coupled devices. Electronic Imaging in Astronomy: Detectors and Instrumentation. 2nd ed. Springer, 2008. ISBN: 978-3-540-76583-7, 241-275.
8. Mandel, L. and Wolf, E. "Quantum theory of photoelectric detection of light." Optical Coherence and Quantum Optics. Cambridge University Press, 1995. ISBN: 978-0-521-41711-2, 723-738.
9. Gonzalez, RC and Woods, RE. "Intensity transformations and spatial filtering." Digital Image Processing. 4th ed. Pearson, 2018. ISBN: 978-0-13-335672-4, 119-191, 318-326.
10. Newberry MV. Signal-to-noise considerations for sky-subtracted CCD data. *Publ Astron Soc Pac*. 1991; 103 (665): 122-130. https://doi.org/10.1086/132801
11. Vincent P, *et al*. Extracting and composing robust features with denoising autoencoders. *Proc 25th Int Conf Mach Learn*. 2008: 1096-1103. https://doi.org/10.1145/1390156.1390294
12. Zhang K, *et al*. Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. *IEEE Trans Image Process*. 2017; 26 (7): 3142-3155. https://doi.org/10.1109/TIP.2017.2662206
13. Hubble Messier Catalog. NASA Science, NASA. Available from: https://science.nasa.gov/mission/hubble/science/explore-the-night-sky/hubble-messier-catalog/ (accessed on 2025-11-02)
14. Wang Z, *et al*. Image quality assessment: from error visibility to structural similarity. *IEEE Trans Image Process*. 2004; 13 (4): 600-612. https://doi.org/10.1109/TIP.2003.819861
15. York DG, *et al*. The Sloan Digital Sky Survey: technical summary. *Astron J*. 2000; 120 (3): 1579-1587. https://doi.org/10.1086/301513
16. Wells DC, *et al*. FITS - a flexible image transport system. *Astron Astrophys Suppl Ser*. 1981; 44: 363-370.
17. Chollet, F. "Deep learning for computer vision." Deep Learning with Python. Manning Publications, 2018. ISBN: 978-1-61729-443-3, 119-177.
18. Merline WJ, Howell SB. A realistic model for point-sources imaged on array detectors: the model and initial results. *Exp Astron*. 1995; 6 (1-2): 163-210. https://doi.org/10.1007/BF00421131
19. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. *Science*. 2006; 313 (5786): 504-507. https://doi.org/10.1126/

science.1127647
20. Krizhevsky A, *et al*. ImageNet classification with deep convolutional neural networks. *Adv Neural Inf Process Syst.* 2012; 25.
21. Goodfellow, I, *et al*. "Deep feedforward networks." Deep Learning. MIT Press, 2016. ISBN: 978-0-262-03561-3, 164-223.
22. Kingma DP, Ba J. Adam: a method for stochastic optimization. Int Conf Learn Represent. 2015.
23. Abadi M, *et al*. TensorFlow: a system for large-scale machine learning. *Proc 12th USENIX Symp Oper Syst Des Implement.* 2016: 265-283.

SUPPLEMENTARY MATERIAL

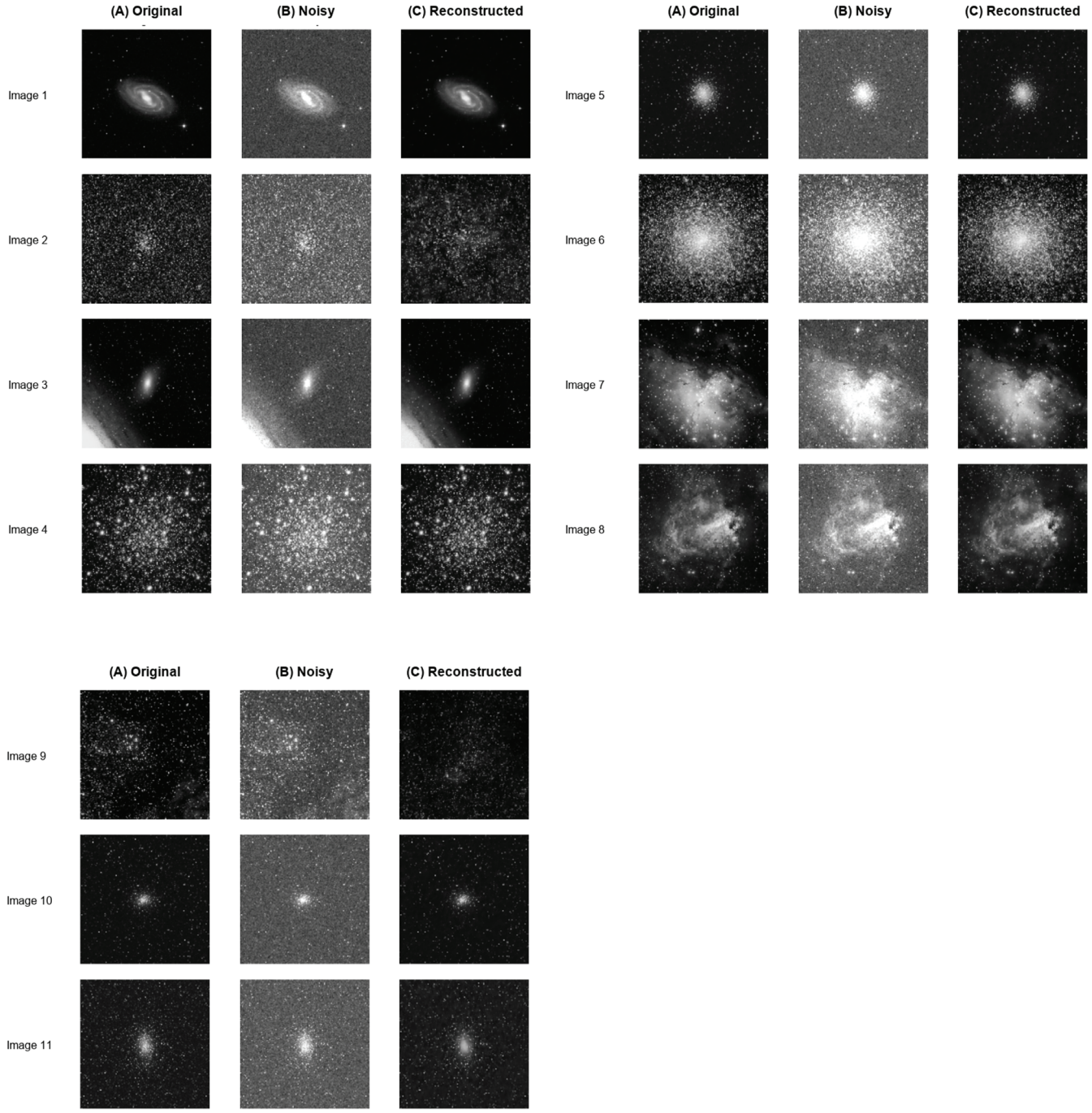

Figure S1. ***Complete denoising results for all 11 held-out test images.*** *Each row corresponds to one test image (Images 1-11 in the same order as Table 1), showing (A) the original clean astrophotography image, (B) the same image with added Gaussian noise (mean = 0.25, σ = 0.05), and (C) the CAE reconstruction. Across the different astronomical objects, the CAE suppressed noise while preserving stellar structures.*