Low-Light Image Enhancement
MAT 494: Computational Methods for Image Processing

Achintya Jha Logan Van Pelt Noah Pack

May 8, 2025

Abstract

Brightening a low-light photograph without amplified noise is a delicate balancing act. We explore the problem from three complementary angles. First, we design an entirely analytic, three-stage pipeline—variance-adaptive CLAHE for local contrast, brightness-aware gamma correction for global tone, and noise-level-controlled non-local means for denoising—whose hyperparameters are computed directly from first-order image statistics. Second, we introduce an ultralight percentile-guided power-law remapping that can be implemented even in resource-constrained settings. Third, we perform a principled "failure analysis" of a gamma-based forward process for diffusion models, showing why its deterministic nature breaks the probabilistic assumptions of score-based generative modeling. Experiments on the publicly available LOw-Light (LOL) dataset demonstrate that our classical pipeline produces visibly cleaner, brighter images in real time on a laptop CPU, while the percentile variant offers a pragmatic fallback and the diffusion study charts a roadmap for future learning-based work.

1 Introduction

Low-light photography poses two coupled challenges: insufficient photons yield dark images, and the high sensor gain needed to compensate injects signal-dependent electronic noise. These artifacts impede consumer snapshots, surveillance feeds, autonomous navigation pipelines, and scientific measurements alike. Although recent deep networks can restore such images, they often require heavy compute, large labelled datasets, and offer little interpretability.

Motivated by scenarios in which transparency, reproducibility, or real-time execution trump sheer benchmark scores, we revisit classical enhancement strategies and ask:

> *Can a fully analytic pipeline rival more complex approaches while remaining lightweight and easy to reason about—and what lessons does that teach us when bridging to modern generative models?*

We answer through three contributions:

1. **Variance-adaptive pipeline.** A cascade of CLAHE, gamma correction, and non-local means, each governed by simple functions of the input's mean and variance (Section 3).
2. **Ultralight percentile remapping.** A two-parameter power-law adjustment driven by a user-chosen percentile and target brightness, yielding a one-function baseline for embedded or mobile devices (Section 4).
3. **Diffusion-model insight.** A controlled experiment replacing Gaussian noise with a deterministic gamma schedule reveals why stochasticity is indispensable for score-based diffusion, guiding future hybrid designs (Section 5).

2 Dataset & Acquisition

Experiments use the publicly available *LOw-Light (LOL) dataset* [1]. LOL contains 500 perfectly aligned image pairs: each scene has a short-exposure, high-ISO *low-light* image and a long-exposure, low-ISO *normal-light* reference. Images are 400×600 pixels and mostly depict indoor scenes. We follow the official split of 485 pairs for training and 15 for testing, enabling direct comparison with prior work.

3 Adaptive Techniques

3.1 Notation

Let the RGB input be $\mathbf{I} \in [0, 255]^{H \times W \times 3}$. Its grayscale mean and standard deviation are $\mu_I = \text{mean}(\text{gray}(\mathbf{I}))$, $\sigma_I = \text{std}(\text{gray}(\mathbf{I}))$.

3.2 Adaptive CLAHE

The luminance channel Y (from an RGB$\rightarrow$YUV conversion) is equalized by contrast-limited adaptive histogram equalization (CLAHE). The clip limit is set adaptively:

$$\text{CL} = \text{clip}\Big(\alpha \, \frac{\sigma_I}{64}, \, 1.0, 4.0\Big), \quad \alpha = 2.0, \tag{1}$$

where $\text{clip}(x, a, b) = \min(\max(x, a), b)$ confines the value to $[a, b]$. Tiles of 8×8 pixels are processed independently and bilinearly blended.

```
def adaptive_clahe(img, clip_limit_factor=2.0, tile_grid_size=(8, 8)):
    img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
    contrast = calculate_contrast(img)
    clip_limit = max(1.0, min(4.0, clip_limit_factor * (contrast / 64.0)))  # Adjust clip limit
    clahe = cv2.createCLAHE(clipLimit=clip_limit, tileGridSize=tile_grid_size)
    img_yuv[:, :, 0] = clahe.apply(img_yuv[:, :, 0])
    return cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
```

3.3 Adaptive Gamma Correction

Global brightness is corrected with a power-law transform:

$$\gamma = \text{clip}\Big(1 + \tfrac{128 - \mu_I}{128}, \, 0.5, 2.5\Big), \tag{2}$$

$$I_{\text{out}}(p) = 255 \left(\frac{I_{\text{in}}(p)}{255} \right)^{1/\gamma}, \qquad p \in \Omega. \tag{3}$$

```
def adaptive_gamma_correction(img):
    brightness = calculate_brightness(img)
    gamma = 1.0 + (128 - brightness) / 128.0  # Adjust gamma based on brightness
    gamma = max(0.5, min(2.5, gamma))  # Limit gamma to a reasonable range
    inv_gamma = 1.0 / gamma
    table = np.array([(i / 255.0) ** inv_gamma * 255 for i in np.arange(0, 256)]).astype("uint8
    return cv2.LUT(img, table)
```

3.4 Adaptive Non-Local Means Denoising

Sensor noise variance is estimated as $\sigma_n = \sigma_I/255$. OpenCV's fast non-local means filter is applied with strength

$$h = \mathrm{clip}\big(\lfloor 30\sigma_n \rfloor,\, 5, 20\big).$$

```
def adaptive_denoise(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    noise_level = np.std(gray) / 255.0  # Estimate noise level
    denoise_strength = max(5, min(20, int(30 * noise_level)))  # Adjust denoising strength
    return cv2.fastNlMeansDenoisingColored(img, None, denoise_strength, denoise_strength, 7, 2
```

3.5 Final Brighten & Denoise Pass

If the intermediate result is still dim ($\mu_I < 50$), an additional gamma lift $\gamma^\star = 2.0\,(\mu_I < 30)$ else 1.5 is followed by a second denoising pass.

```
# Step 4: Brighten and denoise
brightness = calculate_brightness(denoised)
if brightness < 50:
    gamma = 2.0 if brightness < 30 else 1.5
    enhanced_img = np.array(255 * (denoised / 255) ** (1 / gamma), dtype=np.uint8)
    brightened_denoised = adaptive_denoise(enhanced_img)
```

3.6 Summary

Overall, the entire pipeline can be summarized as:

$$\mathbf{I}_{\text{final}} = \mathcal{D}_h\big(\mathcal{G}_{\gamma^\star} \circ \mathcal{D}_h \circ \mathcal{G}_\gamma \circ \mathcal{C}_{\text{CL}}(\mathbf{I})\big).$$

3.7 Results

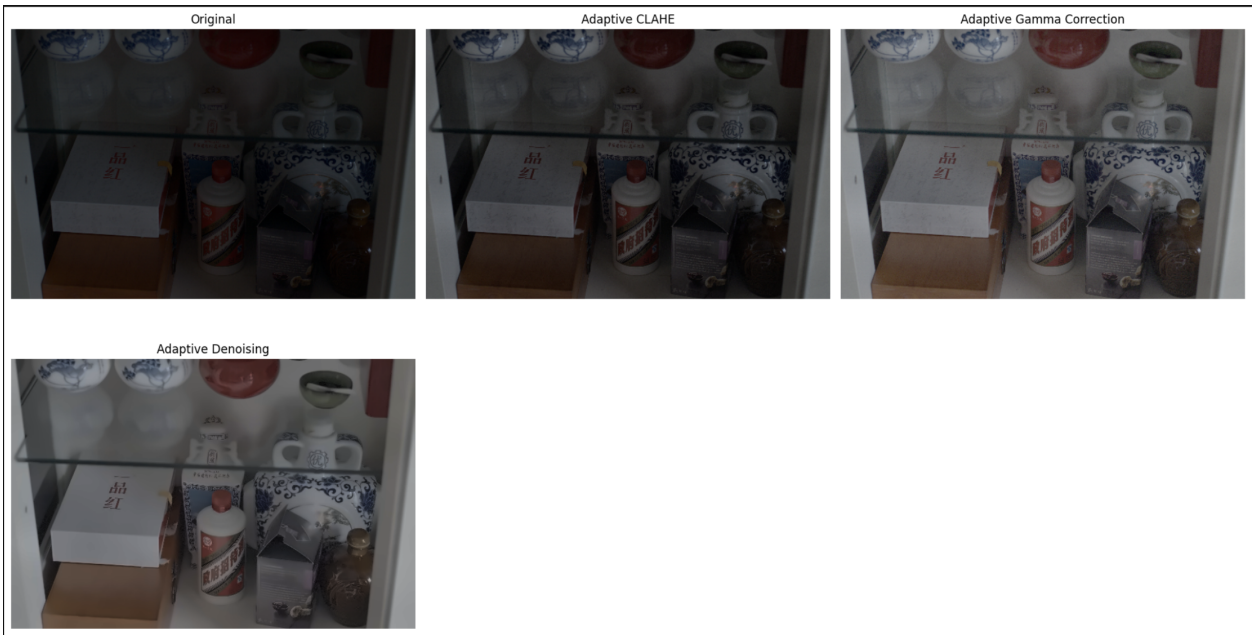

Figure 1: Pipeline output (dark image): Original, Adaptive CLAHE, Adaptive Gamma, Adaptive Denoising, Brightened & Denoised.

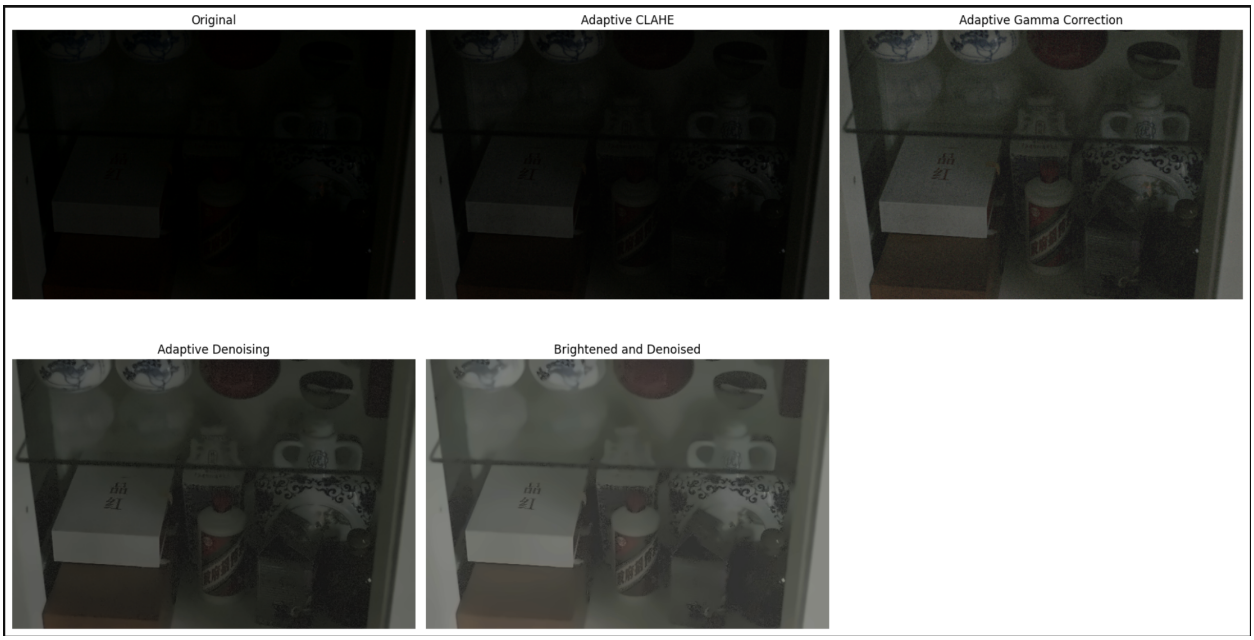

Figure 2: Pipeline output (extremely dark image): Original, Adaptive CLAHE, Adaptive Gamma, Adaptive Denoising, Brightened & Denoised.

4 Alternate Classical Technique

4.1 Notation

Again, the RGB input will be $\mathbf{I} \in [0, 255]^{H \times W \times 3}$. This algorithm takes two tunable input parameters, the percentile and target brightness, $p, t \in [0, 1] \subset \mathbb{R}$.

4.2 Intuition

Assume that the brightness of each pixel for an image follows some distribution. The goal is to develop a function that will remap the distribution of a dark image to the distribution of a standard bright image. A naive assumption about these distributions may be that the pixel brightness of a bright image follows a linear trend, meaning that 10% of the pixels have intensity of 10% or less, 50% of pixels have intensity of 50% or less and so on. For dark images, we assume that they follow the same underlying distribution as a bright image but with a higher gamma value.

4.3 Algorithm

$$\mathrm{I}_{\mathrm{norm}}[i,j] = \frac{\mathrm{I}[i,j] - \min(\mathrm{I})}{\max(\mathrm{I})} \tag{4}$$

$$\mathrm{s} = \mathrm{percentile}(\mathrm{I}_{\mathrm{norm}}, p) \tag{5}$$

$$\mathrm{I}_{\mathrm{adjusted}}[i,j] = \mathrm{I}_{\mathrm{norm}}[i,j]^{\frac{\log(t)}{\log(s)}} \tag{6}$$

The adjustment algorithm was implemented in Python with numpy and Pillow.

```
def normalize_average(img, p, t):
    img = img - np.min(img)
    img = img / np.max(img)
    sample = np.percentile(img, p*100)
    exp = math.log(t) / math.log(sample)
    img = np.pow(img, exp)
    img = np.astype(img * 255, 'uint8')
    return Image.fromarray(img)
```

4.4 Results

This algorithm was very effective at brightening a significant portion of our dataset while preserving detail. One flaw of the algorithm is that it adjusts brightness by exponentiation of the dark image's pixels. This causes areas of the image with one or more channels exactly equal to zero to receive no adjustment. Because of this, pure black pixels will remain pure black, and pixels with only one color channel present will turn pure red, green, or blue.

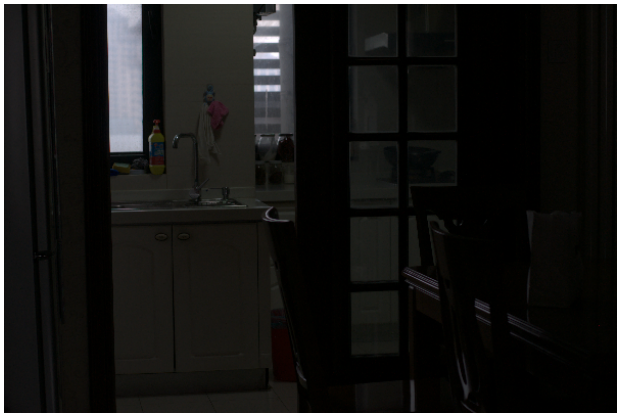

Figure 3: Original dark image

Figure 4: Brightened image

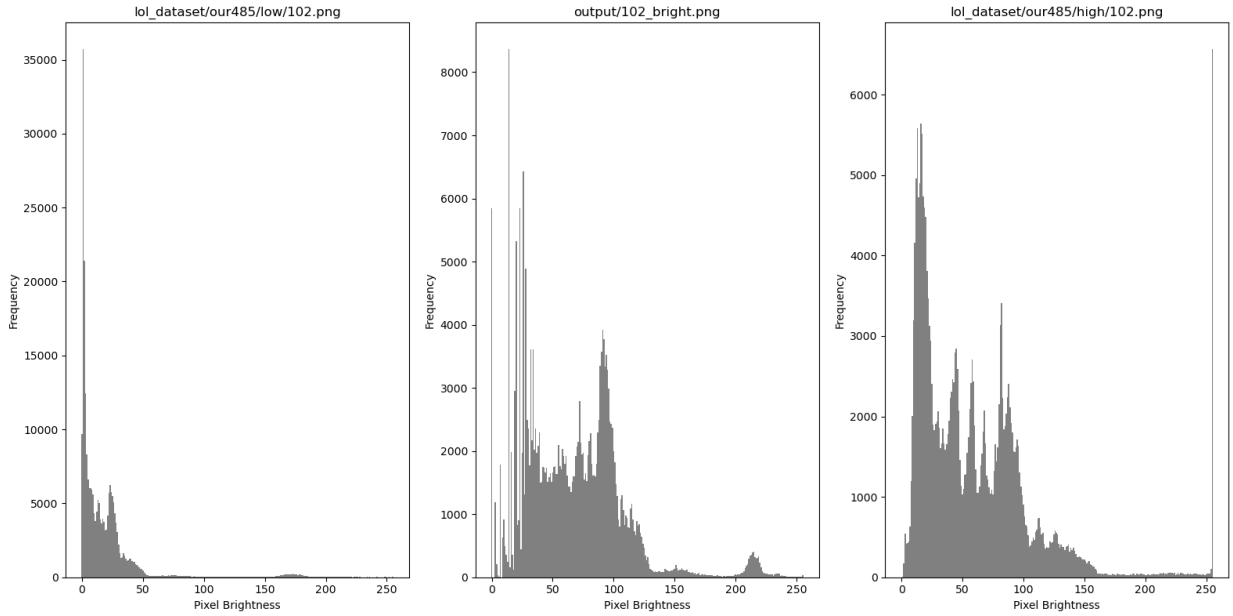

Figure 5: Brightness histogram comparing pixel intensity values in the dark image, our brightened image, and the true bright image.

5 Diffusion Brightening

5.1 Overview of Diffusion Models

Diffusion models are a class of generative models that learn to reverse a gradual noising process to generate data samples, such as images, from random noise. In a forward process, the data is slowly corrupted with Gaussian noise over T time steps, and a neural network is trained to reverse this corruption in what is called the reverse process. From here on we will discuss diffusion assuming images as our data.

The forward process defines a sequence of noisy images $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T$, where $\mathbf{x}_0$ is the input image

and each subsequent $\mathbf{x}_t$ is generated via a Gaussian transition:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}, \mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{7}$$

where β_t is a parameter controlling how much noise is added at each timestep t. β_t is defined as $1 - \alpha_t$ where α_t represents how much of the original image is preserved after each step.

The entire forward process can be described by the joint distribution:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{8}$$

This is a multivariate Gaussian over the vector $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$.

There exists a closed-form solution for sampling $\mathbf{x}_t$ given $\mathbf{x}_0$:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}, \mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{9}$$

Where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

This formulation is useful for training, as it allows for sampling at arbitrary time steps without having to go through a large number of intermediate steps.

5.2 Gamma-Based Forward Process for Image Brightening

In an attempt to use diffusion models for image brightening, the Gaussian noise-schedule in the forward process was replaced by a gamma schedule. Specifically, a new forward process was defined where:

$$\mathbf{x}_t = \mathbf{x}_0^{\gamma_t} \tag{10}$$

where γ_t is a schedule of increasing exponents. The idea was to simulate step-wise image darkening using a gamma curve and then train a model to reverse this process by learning to brighten the images.

5.3 Why This Approach Fails

While intuitive, this gamma-based approach fails as a diffusion model for several key reasons:

- **No stochasticity:** The process $\mathbf{x}_t = \mathbf{x}_0^{\gamma_t}$ is deterministic. There is no random variable involved, so the conditional distribution $q(\mathbf{x}_t|\mathbf{x}_0)$ cannot be defined in a meaningful way. There is no uncertainty for the model to learn.
- **No probabilistic framework:** Diffusion models are built on probabilistic transitions. Without a meaningful probability distribution for $q(\mathbf{x}_t|\mathbf{x}_0)$, the reverse process becomes the model trying to learn the inverse of the gamma curve.
- **Lack of denoising structure:** In standard diffusion, the model learns to map many possible noisy versions of an image back to the same clean image. Darkening according to a gamma schedule doesn't create such variability.

5.4 Conclusion

While gamma correction may seem like a natural fit for image brightening, its deterministic nature makes it unsuitable for direct use in diffusion models. Standard diffusion models require a stochastic forward process to be effective. Future work should focus on designing forward processes that combine perceptual image transformations (like dimming or gamma) with Gaussian noise, or reformulate the task as conditional image generation.

References

[1] C. Wei, W. Wang, W. Yang, & J. Liu. (2018). *Deep Retinex decomposition for low-light enhancement* (Version 1) [Preprint]. arXiv:1808.04560.

[2] J. Ho, A. Jain, & P. Abbeel. (2020). *Denoising diffusion probabilistic models* (Version 1) [Preprint]. arXiv:2006.11239.