PROJECT ABSTRACT

Low-Light Image Denoising using DnCNN

A DnCNN-based denoiser that cleans noise from low-light smartphone photos, built on Zhang et al.'s residual-learning network, with a Flask demo, noise-level slider, PSNR/SSIM evaluation notebook, report, PPT and viva kit — delivered as a built-to-order final year project.

01. Title

Low-Light Image Denoising using DnCNN

02. Objective

Smartphone photos taken in dim light come out grainy, and the phone's built-in cleanup often smears fine detail away with the noise. This project implements DnCNN — 'Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising' (Zhang et al., arXiv 1608.03981, TIP 2017) — in PyTorch and wraps it in a Flask demo where a noisy low-light photo is uploaded, the noise level is set with a slider, and the noisy and cleaned versions are compared side by side. The deliverable includes an evaluation notebook that computes PSNR and SSIM on the BSD68 benchmark during the build, giving the report a genuine, auditable experiment.

03. Method

DnCNN's core idea is residual learning: instead of predicting the clean image directly, the network learns to predict the noise, which is then subtracted from the input. Noise has simpler structure than natural images, so the residual mapping is easier to learn and trains faster and more stably. The architecture is a deep stack of 3x3 convolutions with batch normalization and ReLU, which keeps training stable at depth. The network is conditioned on noise level, so one model covers the range of grain real low-light photos exhibit.

04. How the Demo Works

- A noisy low-light photo (grayscale or color) is uploaded through the Flask demo app.
- The image is split into overlapping patches so arbitrary image sizes are handled.
- Each patch passes through the DnCNN, which outputs a predicted noise residual — not the clean image.
- The predicted noise is subtracted from the input patch, producing the clean estimate.

- Patches are stitched back with overlap blending to avoid visible seams.
- The noise-level slider selects the model conditioning matching the photo's grain; presets cover light, medium and heavy noise.
- A batch mode runs a whole folder of images — used for the BSD68 evaluation.
- The cleaned result is shown side by side with the original and is downloadable.

05. Software and Data

Python 3.10 with PyTorch for the DnCNN implementation; OpenCV, NumPy and PIL for patch handling, stitching and I/O; Matplotlib for PSNR/SSIM logging and comparison plots; a Flask web app for the demo. Evaluation uses the BSD68 (68 natural grayscale images) and Set12 (12 images) benchmarks — the standard sets from the original paper — covering textures, edges and smooth regions. Trained DnCNN weights ship with the build.

06. Key Results (Design Targets)

No scores are claimed in advance: the model is trained to order, so the evaluation notebook computes PSNR and SSIM on BSD68 during the build and the report documents the measured numbers. Design targets are the paper's operating regime — one conditioned model across noise levels — with the honest caveat that real smartphone noise differs from synthetic training noise, so real captures are the tougher test and the report discusses this gap explicitly.

07. Limitations

- The model assumes roughly Gaussian noise; heavy JPEG artifacts or motion blur need different methods.
- An almost-black image has no signal to recover — denoising removes noise, it does not invent detail never captured.
- High noise-level settings can over-smooth fine texture, trading grain for softness.
- This is an educational prototype, not a replacement for a camera's image pipeline.

08. Conclusion

The project delivers a working, built-to-order DnCNN denoising pipeline: trained weights, a polished Flask demo with noise-level control, and an evaluation notebook that produces auditable PSNR/SSIM numbers on BSD68. It gives students a clean, well-scoped deep-learning build — residual learning, batch normalization and patch-based inference — with honest scope and real experimental material for the report and viva.