Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

CNN-Based Denoising of Industrial CT Images: A Residual-Learning Benchmark Against Classical Filters on Synthetic Phantoms

Pushkar
Indian Institute of Technology Roorkee
Mayank Goswami
mayankfph@iitr.ac.in

Indian Institute of Technology Roorkee

Research Article

Keywords: computed tomography, image denoising, convolutional neural network, DnCNN, residual learning, synthetic phantoms, non-destructive testing

Posted Date: November 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11642694/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

CNN-Based Denoising of Industrial CT Images: A Residual-Learning Benchmark Against Classical Filters on Synthetic Phantoms

Pushkar[1] and Mayank Goswami [1, 2, #]
[1]Department of Physics, IIT Roorkee
[2]*Mehta Family School of Data Science and Artificial Intelligence*, IIT Roorkee,
Roorkee, Uttarakhand-247667, India.
[#]mayank.goswami@ph.iitr.ac.in

Manuscript Overview and Experimental Scope

Attribute	Experimental Specification	Methodology and Models	Primary Benchmark Outcome
Research domain	Industrial X-ray CT image denoising; synthetic, non-biological phantoms	Residual-learning CNN (DnCNN, 17 layers) against Gaussian, median and non-local means filters	DnCNN: 50.28 dB PSNR, 0.998 SSIM
Dataset scale	5,000 phantoms (256 × 256 px); 5 outer shapes; 3 dose classes	Poisson plus Gaussian noise; seed 42	400 held-out test images; DnCNN wins on 400 of 400
Co-author	Professor Mayank Goswami (course professor, PHC-351)	MSE loss, Adam, cosine schedule, 30 epochs, 7,860 steps	+12.34 dB over the noisy input; +4.72 dB over the best classical filter

Abstract

X-ray computed tomography (CT) lets engineers look inside manufactured parts without cutting them open. The catch is that the images are noisy, and the noise gets worse as the X-ray dose drops. That grain can hide the small pores and inclusions an inspector is actually looking for, and classical filters, which follow fixed rules, tend to blur the very edges they should protect. In this project, carried out with co-author **Professor Mayank Goswami** for course PHC-351 (Project 9, Category B), I built a fully synthetic, non-biological benchmark of 5,000 industrial phantoms (256 × 256 pixels). The phantoms have five outer shapes, bright rims, 0 to 6 dense inclusions and 0 to 9 pores, and the noise combines Poisson photon statistics with Gaussian electronic noise at three dose levels. I trained a 17-layer residual convolutional network (DnCNN, 556,096 parameters) on 4,200 images, validated it on 400, and compared it with Gaussian, median and non-local-means filters on 400 images it had never seen.

DnCNN reached a mean peak signal-to-noise ratio (PSNR) of 50.28 dB and a structural similarity (SSIM) of 0.998. For comparison, non-local means scored 45.56 dB and 0.910, the median filter 41.38 dB and

0.964, the Gaussian filter 37.44 dB and 0.857, and the noisy input 37.93 dB and 0.809. DnCNN beat the best classical filter on all 400 test images (smallest margin +1.87 dB, mean +4.72 dB) and at every dose level (low 48.92 dB, medium 50.06 dB, high 50.88 dB). One caveat matters: training and test noise come from the same simulator, so these numbers are a best case, and testing on real scanner data is the obvious next step.

Keywords: computed tomography, image denoising, convolutional neural network, DnCNN, residual learning, synthetic phantoms, non-destructive testing

1. Introduction and Motivation

1.1 Problem Statement

Industrial CT shows the inside of a part without cutting it open. It is used to find pores, cracks, inclusions and thin or missing walls in cast, printed and sintered components. What limits the image quality is noise, and that noise has a physical cause. A detector counts individual X-ray photons, and the count in each pixel changes from one measurement to the next by about the square root of the mean count (Poisson statistics). So the relative noise falls as $1/\sqrt{N}$: fewer photons, whether from a lower dose, a shorter scan or a thicker and denser part, give a grainier image. The detector electronics add a second, roughly Gaussian, noise term on top.

1.2 Need for the Study

The obvious fix is to scan for longer or with more X-ray power, but that costs time, money and tube life, and it is not possible for every part. Cleaning the image in software is cheaper, as long as the filter removes the grain without erasing real structure. Three things make this hard:

- **Edges versus grain.** Fixed-rule filters average neighbouring pixels. That smooths the noise, but it also blurs thin walls, rounds off sharp corners and shrinks small pores, and those are the features an inspector needs to see.
- **No clean reference in real scans.** Supervised learning needs pairs made of a noisy image and a perfectly clean one. A noise-free scan of a real part does not exist, so the true error of a denoiser cannot be measured on real data.
- **Fair comparison.** A new method is only convincing if the classical filters it is compared with are tuned honestly and the test images play no part in any choice.

1.3 Aim and Contributions

To deal with these three points, I built my own data, where the clean answer is known exactly, and tested whether a small convolutional neural network (CNN) can beat the standard filters. The work has four parts:

1. A reproducible synthetic dataset of 5,000 industrial CT-like phantoms with five outer shapes, mixed inclusions and pores, and three dose classes, generated with a fixed random seed (42). It contains nothing biological.
2. A DnCNN trained on this dataset with a fully written-down recipe: loss, optimiser, schedule, crops, augmentation and checkpoint rule.

3. A head-to-head comparison against Gaussian, median and non-local-means filters, each tuned on held-out validation images and scored by PSNR and SSIM on 400 unseen test images, overall, by dose class and by outer shape.
4. A plain account of what a purely synthetic study can and cannot show, so that the numbers are read for what they are.

2. Related Work

Classical filters. The Gaussian filter replaces each pixel with a weighted average of its neighbours. It removes noise, but it blurs edges in the same way. The median filter takes the middle value of a small window; it keeps edges better, but it can round off corners and thin features. Non-local means (Buades et al., 2005) averages pixels whose surrounding patches look alike, even if they are far apart in the image, and it is a strong baseline when structures repeat. BM3D (Dabov et al., 2007) goes a step further: it groups similar patches into three-dimensional stacks and filters each stack together, and it is still one of the strongest classical denoisers. All of these need a strength setting, which is chosen by hand or tuned on data.

Learned denoisers. Zhang et al. (2017) introduced DnCNN, a deep CNN with batch normalisation that learns the noise itself (residual learning) rather than the clean image. Predicting the noise is the easier problem, because noise is simpler and has less structure than the image, and it let a single network handle a range of noise levels. In the same year, Chen et al. (2017) applied a residual encoder-decoder CNN (RED-CNN) to low-dose CT and tested it on simulated low-dose images and on clinical data from a low-dose CT challenge. Lehtinen et al. (2018) then showed with Noise2Noise that a network can be trained from pairs of noisy images alone, with no clean target, which helps whenever a clean reference is hard to get. I use the original DnCNN form because it is small, well understood and easy to reproduce.

Synthetic data. Real industrial scans rarely come with a noise-free reference. Simulated phantoms give an exact ground truth, so the error of any method can be measured exactly and the comparison with classical filters is fair. The price is that a simulation can differ from a real scanner, which I come back to in Section 6.

Position of this work. This project does not propose a new network. What it offers is a controlled, reproducible benchmark on synthetic industrial phantoms, in which a small DnCNN and three classical filters are tuned, trained and tested under identical conditions. BM3D, RED-CNN and Noise2Noise are described for context but were not tested, and this short survey is a selection of well-known work, not a complete review.

3. Synthetic Dataset and Noise Model

To get a benchmark with no experimental imaging noise and an exactly known answer, I generated 5,000 clean and noisy image pairs by code. Each image is a single-channel 256 × 256 slice stored as 8-bit integers, and the whole dataset can be recreated from the random seed 42. Figure 1 shows the complete workflow, from phantom generation to the final test.

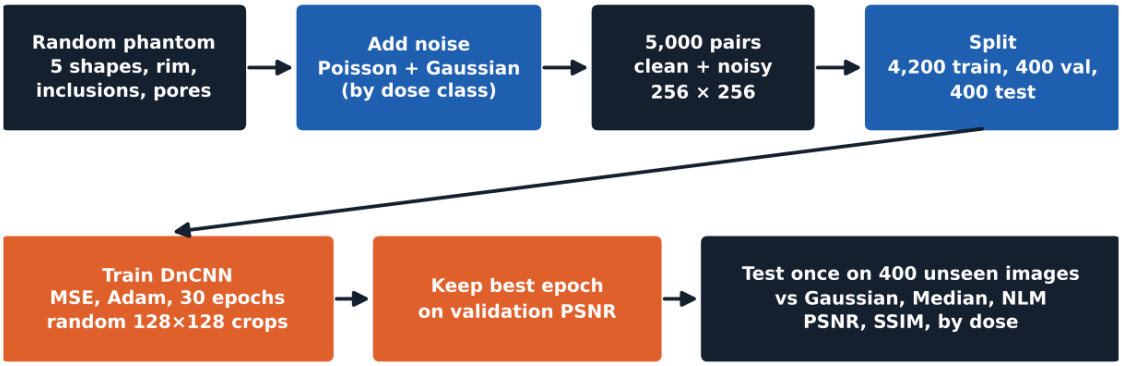


Figure 1. Overall workflow. Random phantoms are drawn and noise is added to make 5,000 clean and noisy pairs, which are split into 4,200 training, 400 validation and 400 test pairs. DnCNN is trained on the training pairs, the best epoch is chosen on the validation pairs, and every method is scored once on the test pairs.

3.1 Phantom Generation

Each clean image is a cross-section of an imaginary industrial part on a black background. The outer shape is picked at random from five kinds: circle, ellipse, rotated rectangle, polygon (3 to 8 sides) and irregular blob. Its size is 30 to 46 percent of the image width, and it sits a few pixels off the centre. The material has a random base intensity between 0.20 and 0.65, and a brighter rim 1 to 4 pixels thick (at 128-pixel scale) imitates the edge effect seen in real CT. Inside the part I draw 0 to 6 dense inclusions (intensity 0.50 to 0.98) and 0 to 9 pores (intensity 0.00 to 0.08). Inclusions and pores are also circles, ellipses, rectangles, polygons or blobs, and they are only drawn inside the part. Each image is drawn at twice the final resolution and averaged down, which gives smooth edges. Figure 2 shows five examples of each outer shape.

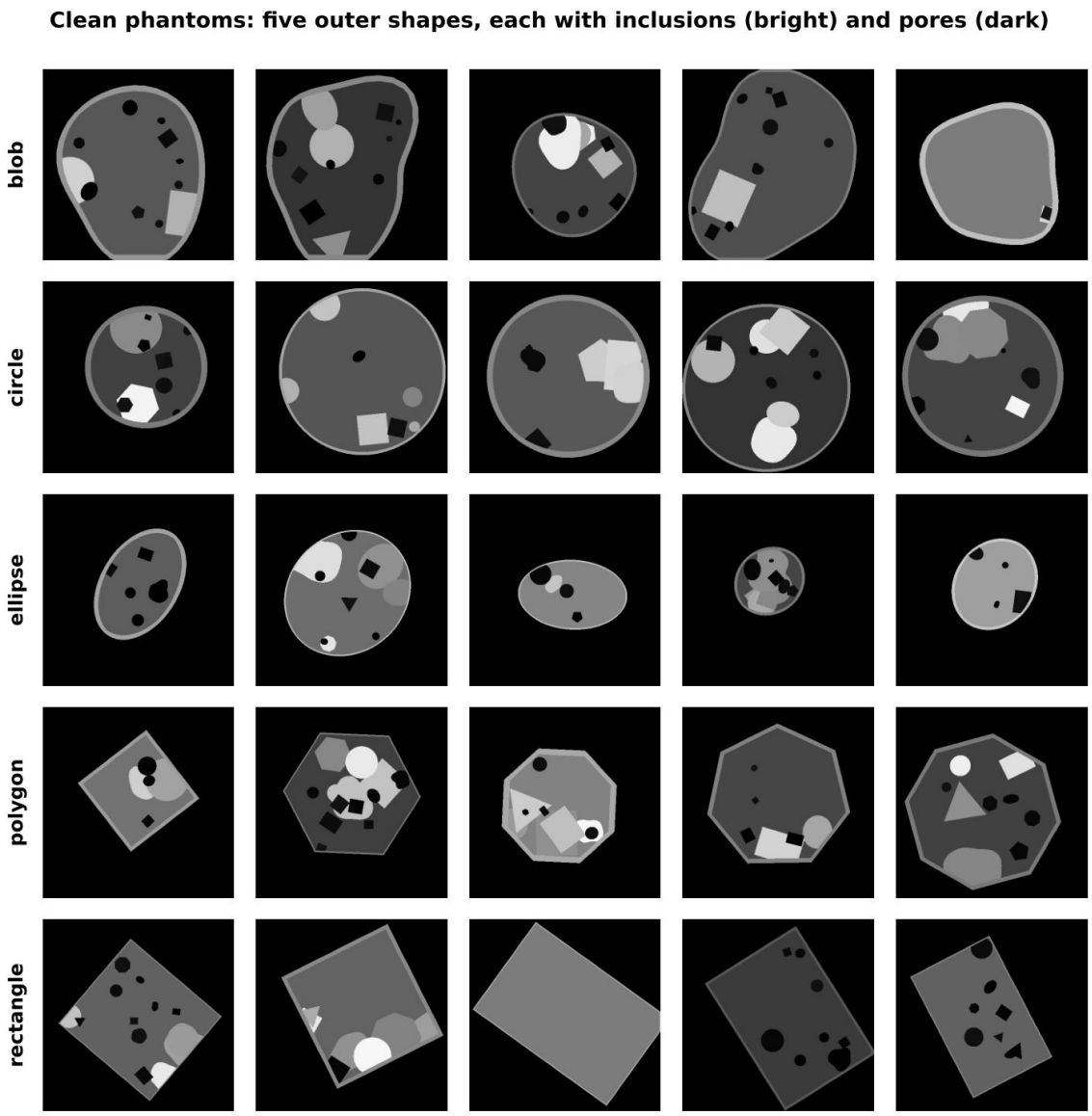


Figure 2. Clean phantoms. Five random examples of each outer shape (rows: blob, circle, ellipse, polygon, rectangle). Bright regions are dense inclusions, dark regions are pores, and the thin lighter outline is the rim.

3.2 Noise Model

Each phantom gets a noisy copy in two steps. First comes photon counting noise: the clean intensity x (0 to 1) is multiplied by a photon count N, a Poisson random number is drawn for every pixel, and the result is divided by N again. Then I add a little Gaussian noise with standard deviation σ to stand in for the detector electronics:

```
y = \frac{1}{N}\,\mathrm{Poisson}(N x) + \varepsilon, \qquad \varepsilon \sim
\mathcal{N}(0,\sigma^{2}), \qquad
\mathrm{std}\left[\tfrac{1}{N}\mathrm{Poisson}(Nx)\right] = \sqrt{x/N}
```

The photon count is drawn for each image between 800 and 6,000, and σ between 0.005 and 0.02. Images are grouped by photon count into three dose classes: low dose (below 1,800 photons), medium dose (1,800 to 3,199) and high dose (3,200 and above). Fewer photons mean larger relative noise, as Figure 3 shows for one test image from each class. Across the whole dataset, the PSNR of the noisy image against its clean image averages 37.88 dB and ranges from 31.59 to 46.57 dB (Figure 4).

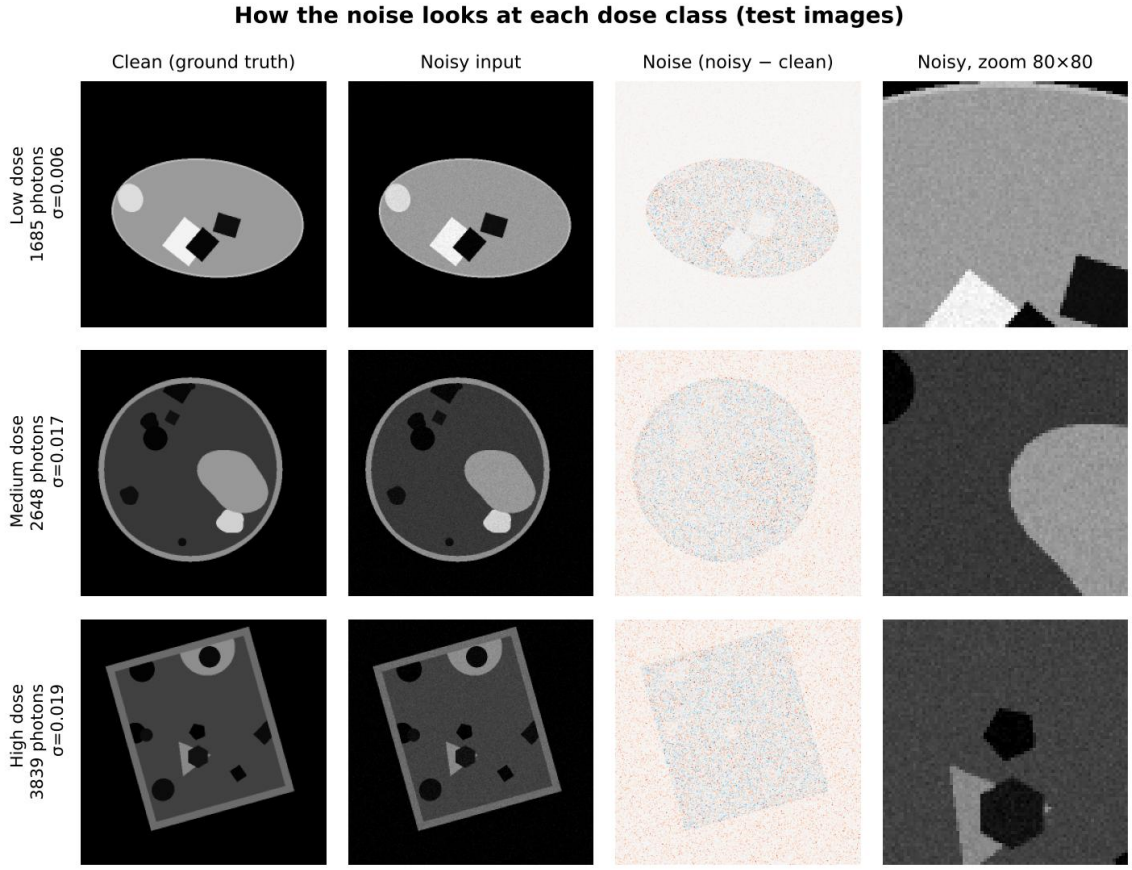


Figure 3. Noise at each dose class. For one test image per dose class: the clean ground truth, the noisy input, the noise itself (noisy minus clean, blue and red = negative and positive), and an 80 × 80 pixel zoom of the noisy image. The photon count and Gaussian σ of each image are printed on the left.

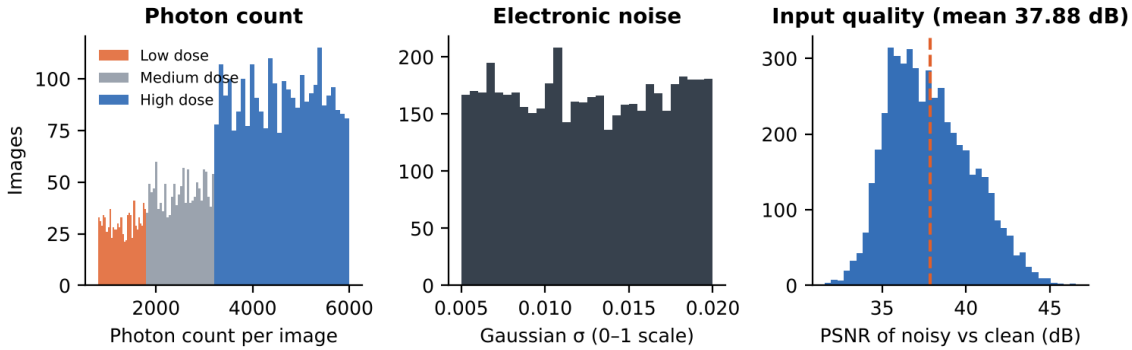


Figure 4. Noise parameters over the 5,000 images. Left: photon count, coloured by dose class. Middle: Gaussian standard deviation. Right: PSNR of each noisy image against its clean image; the dashed line marks the mean (37.88 dB).

3.3 Dataset Statistics

The phantoms are well balanced across their random properties. Each outer shape makes up about one fifth of the images (blob 990, circle 987, ellipse 994, polygon 1,010, rectangle 1,019). Every inclusion count from 0 to 6 occurs in 686 to 752 images, and every pore count from 0 to 9 in 476 to 519 images, so the network sees clean, crowded and empty parts alike (Figure 5). The dose classes are not the same size, because photon counts were drawn uniformly between 800 and 6,000: there are 907 low-dose, 1,349 medium-dose and 2,744 high-dose images.

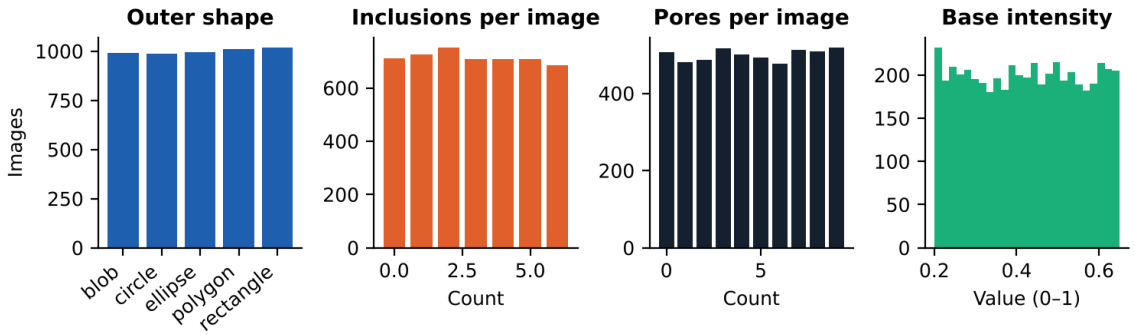


Figure 5. Phantom statistics over the 5,000 images. Outer-shape counts, inclusions per image, pores per image, and the base intensity of the material.

3.4 Data Split

Every image is generated independently with random settings, so I split the dataset by index into 4,200 training, 400 validation and 400 test images. No image appears in two sets, so nothing leaks from training into the test, and all three sets have a similar mix of dose classes (Table 1, Figure 6). The validation images are used to choose the best epoch and to tune the classical filters. The test images are used once, at the end.

Table 1. Dataset partition and dose-class mix.

Partition	Images	Low dose	Medium dose	High dose	Role in the study
Training	4,200	743	1,146	2,311	Fitting the DnCNN weights
Validation	400	81	108	211	Choosing the best epoch, tuning the filters
Test	400	83	95	222	Final scoring (used once)
Total	**5,000**	**907**	**1,349**	**2,744**	Complete benchmark dataset

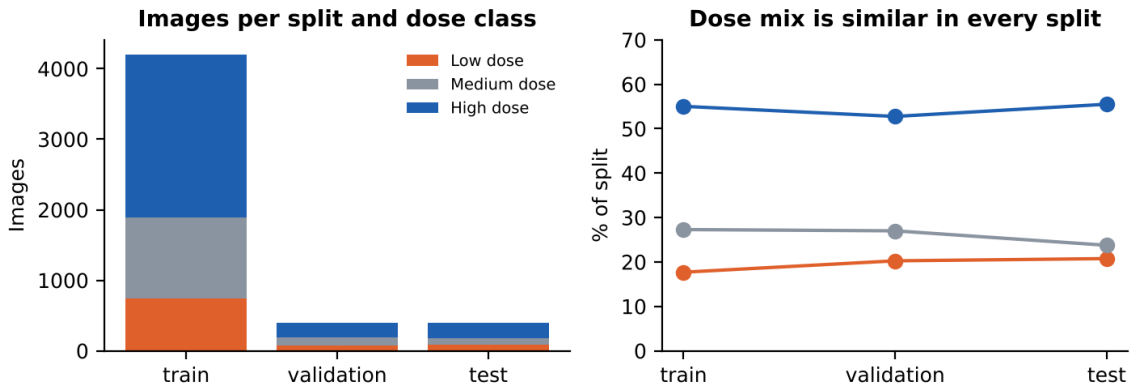


Figure 6. Composition of the three splits. Left: images per split, stacked by dose class. Right: share of each dose class in each split, which is similar in all three.

4. Network, Baselines and Metrics

4.1 DnCNN with Residual Learning

The network is a DnCNN with 17 convolutional layers of 3 × 3 filters (Figure 7). The first layer maps the 1-channel image to 64 channels and is followed by a ReLU activation. Each of the 15 middle layers has 64 filters, batch normalisation and a ReLU; their filters have no bias, because batch normalisation already learns a shift. The last layer maps the 64 channels back to 1. There are 556,096 trainable parameters in total, and each output pixel depends on a 35 × 35 pixel window of the input (2 × 17 + 1). The body of the network does not output the clean image. It predicts the noise, and the result is the input minus that prediction:

```
\hat{x} = y - R(y;\theta), \qquad \mathcal{L}(\theta) =
\frac{1}{B}\sum_{b=1}^{B}\left\lVert \hat{x}_b - x_b \right\rVert_2^{2}
```

Here y is the noisy image, x the clean image, R the network body with weights θ, and B the number of images in a batch.

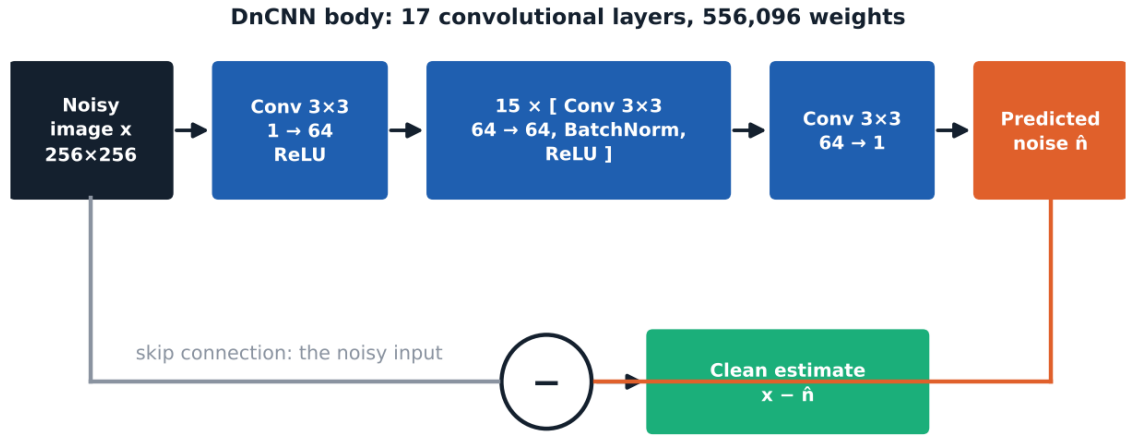


Figure 7. DnCNN architecture. A first convolution with ReLU, 15 blocks of convolution, batch normalisation and ReLU, and a last convolution produce the predicted noise, which is subtracted from the noisy input (skip connection) to give the clean estimate.

4.2 Training Recipe

The network is trained to minimise the mean squared error between its output and the clean image. After every epoch I score it on the 400 validation images and keep the checkpoint with the best validation PSNR. The test images are never used to choose any setting. At test time the network is applied to the whole 256 × 256 image. Table 2 lists every setting so the run can be repeated.

Table 2. Complete training recipe.

Setting	Value
Model	DnCNN, 17 layers, 64 channels, 556,096 weights
Loss	Mean squared error against the clean image
Optimiser	Adam, learning rate 0.001, cosine decay to 0
Batches	16 random 128 × 128 crops, random 90° rotation and flip
Length	30 epochs × 262 steps = 7,860 steps
Numerical extras	Mixed precision (fp16), gradient clipping at norm 1.0
Checkpoint rule	Keep the epoch with the best validation PSNR (50.38 dB, epoch 29)
Hardware, seed	Free Google Colab GPU, random seed 42

4.3 Classical Baselines

Three classical filters serve as the competition. Each has one setting tuned on 40 validation images by maximising PSNR: a Gaussian filter with standard deviation 0.5 pixel, a median filter with a 3 × 3 window, and non-local means with patch size 5, search distance 6 and filter strength h = 0.03. The Gaussian and non-local-means settings came from a small grid of candidates (for h: 0.01, 0.02, 0.03, 0.05, 0.08), and no test image was used in the tuning.

4.4 Image-Quality Metrics

Every method is scored against the clean image by the same two standard scikit-image functions, on intensities scaled to 0 to 1. The peak signal-to-noise ratio (PSNR) measures the pixel-by-pixel error on a logarithmic scale (higher is better; every +10 dB means ten times less squared error):

$$\mathrm{PSNR} = 10\,\log_{10}\left(\frac{1}{\mathrm{MSE}}\right)\ \mathrm{dB}, \qquad \mathrm{MSE} = \frac{1}{P}\sum_{p=1}^{P}\left(\hat{x}_p - x_p\right)^{2}$$

The structural similarity index (SSIM, 0 to 1, higher is better) compares local brightness, contrast and structure, which is closer to what the eye notices. I report both metrics because a filter can do well on one and badly on the other: smoothing a flat area raises SSIM, while smearing edges lowers PSNR. I

report means and standard deviations over the 400 test images, overall, by dose class and by outer shape, and I count the test images on which DnCNN beats the best classical filter.

5. Experimental Results

5.1 Head-to-Head Comparison on the 400 Test Images

DnCNN scored highest on both PSNR and SSIM, and it beat the best classical filter, non-local means, on all 400 test images (100 percent). Table 3 gives the mean over the test set, with the standard deviation across images in brackets, plus the median and range of the per-image PSNR.

Table 3. Test-set image quality (400 images; mean, with standard deviation across images in brackets).

Method	PSNR (dB)	PSNR median [min, max]	SSIM	Gain over noisy input
Noisy input	37.93 (2.46)	37.57 [33.01, 45.50]	0.809 (0.087)	–
Gaussian filter	37.44 (1.54)	37.32 [34.01, 43.14]	0.857 (0.071)	−0.49 dB
Median filter	41.38 (2.57)	41.59 [33.70, 48.30]	0.964 (0.019)	+3.45 dB
Non-local means	45.56 (2.02)	45.30 [41.41, 51.61]	0.910 (0.058)	+7.63 dB
DnCNN	**50.28 (1.87)**	**50.08 [45.86, 55.43]**	**0.998 (0.001)**	**+12.34 dB**

The Gaussian filter raised SSIM but lowered PSNR, because blurring removes some noise but also smears the edges. The median filter and non-local means gained 3.45 dB and 7.63 dB over the noisy input, and DnCNN gained 12.34 dB, which is about 17 times less squared error. Figure 8 shows four test images processed by every method. The Gaussian and median outputs still show grain, non-local means is cleaner but visibly further from the clean image than DnCNN, and the DnCNN output is almost impossible to tell from the ground truth, with sharp edges, intact corners and small pores kept.

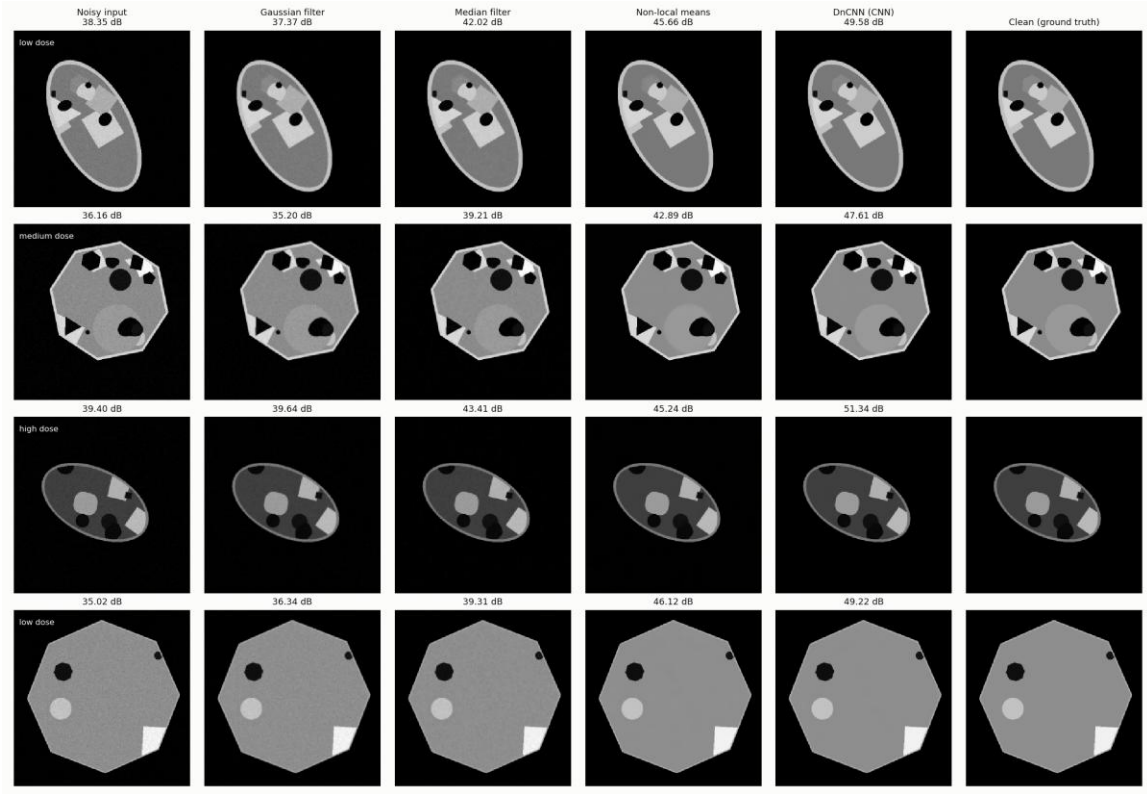


Figure 8. Visual comparison. Four test images (columns: noisy input, Gaussian, median, non-local means, DnCNN, clean ground truth). The PSNR of each result is printed above it.

The spread across images tells the same story (Figure 9). The DnCNN box sits entirely above every other method, and its lowest image (45.86 dB) is better than the median of non-local means. DnCNN is also the most consistent method in SSIM: its standard deviation across images (0.0007) is far smaller than that of any other method, and its worst image (0.9953) is better than the best median-filter image (0.9948) and the best non-local-means image (0.9948).

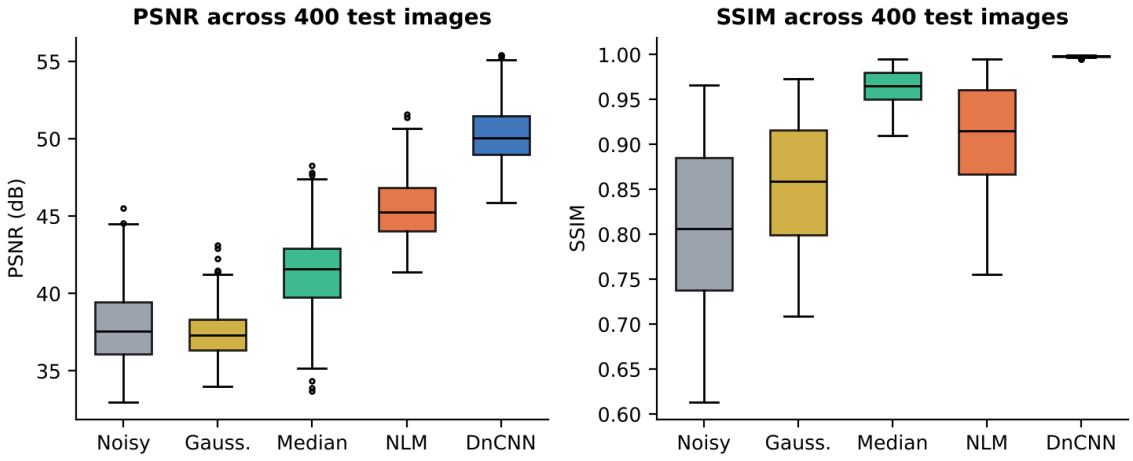


Figure 9. Spread across the 400 test images. Box plots of PSNR (left) and SSIM (right) for each method. Boxes show the quartiles, the line the median, and dots the outliers.

Figure 10 compares DnCNN and non-local means image by image. All 400 points lie above the diagonal, so DnCNN wins on every image, with an average margin of +4.72 dB (minimum +1.87 dB, maximum +8.17 dB). In SSIM, DnCNN also beats both non-local means and the median filter on all 400 images, while the median filter beats non-local means on 373 of 400 (93 percent), which is why the median filter has the higher mean SSIM.

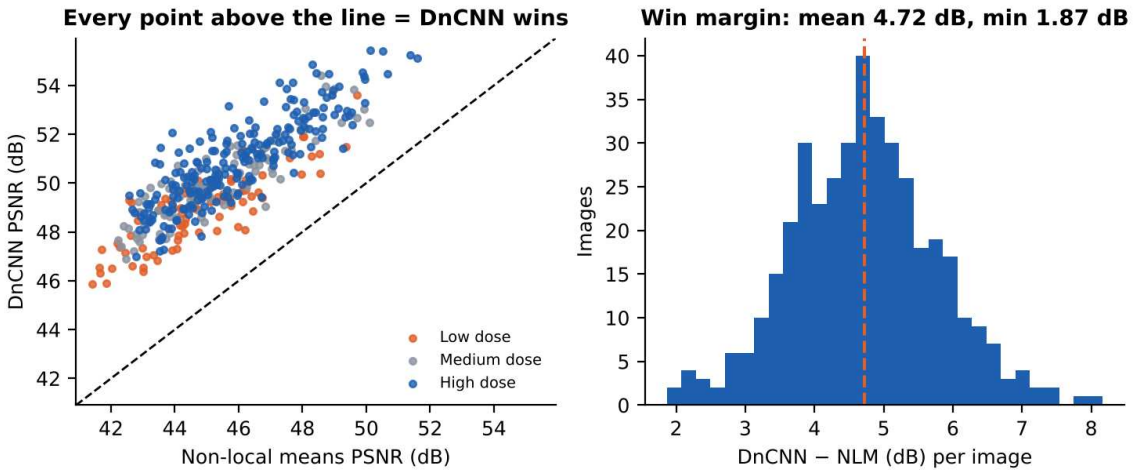


Figure 10. DnCNN versus non-local means, image by image. Left: PSNR of the two methods for each test image, coloured by dose class; points above the dashed line are DnCNN wins. Right: histogram of the per-image PSNR margin.

5.2 Effect of Dose

Table 4 splits the test set by dose class. DnCNN is best in every class, and its gain over the noisy input is almost the same in all three (12.22, 12.60 and 12.28 dB). Its lead over non-local means grows with dose, from 4.08 dB at low dose to 4.95 dB at high dose. Quality is lowest at low dose, as expected, but even there DnCNN (48.92 dB) is above non-local means at high dose (45.93 dB).

Table 4. PSNR (dB) / SSIM by dose class (test set).

Method	Low dose (83 images)	Medium dose (95 images)	High dose (222 images)
Noisy input	36.70 / 0.794	37.46 / 0.796	38.60 / 0.820
Gaussian filter	36.98 / 0.847	37.31 / 0.848	37.67 / 0.865
Median filter	40.28 / 0.960	41.53 / 0.961	41.73 / 0.967
Non-local means	44.84 / 0.909	45.33 / 0.904	45.93 / 0.913
DnCNN	**48.92 / 0.998**	**50.06 / 0.998**	**50.88 / 0.998**

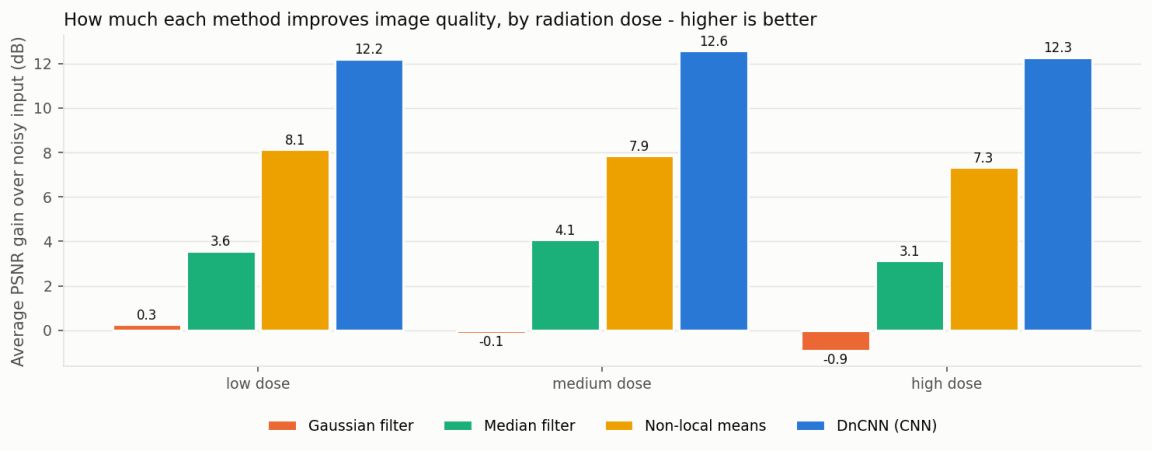


Figure 11. Gain by dose class. Average PSNR gain over the noisy input for each method, by dose class.

Plotting every test image against its photon count (Figure 12) shows that all methods improve as the dose rises, but the DnCNN gain over the noisy input stays flat. It averages 12.34 dB (standard deviation 1.19 dB, range 8.56 to 15.55 dB) and is essentially uncorrelated with the photon count (correlation −0.04). In this simulated setting, then, the network handles low-dose and high-dose noise about equally well.

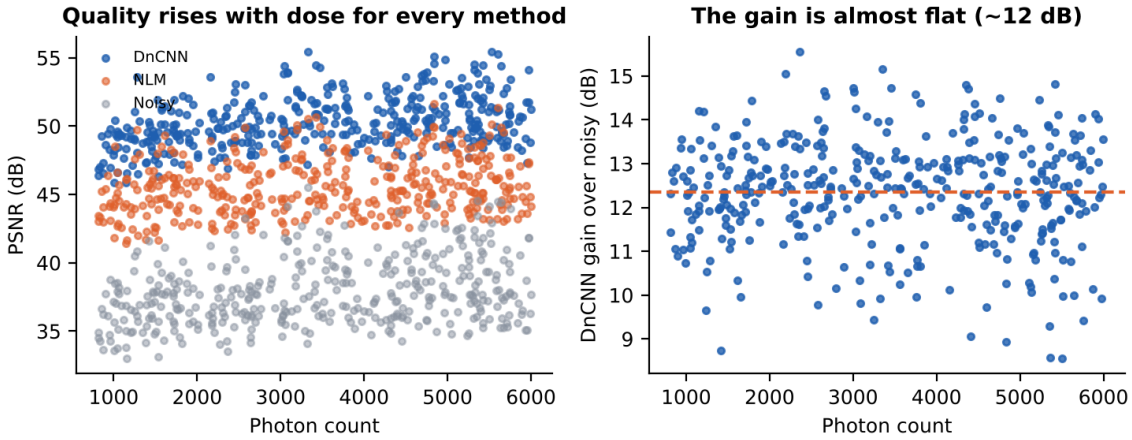


Figure 12. Dependence on dose. Left: PSNR of the noisy input, non-local means and DnCNN against the photon count of each test image. Right: DnCNN gain over the noisy input, which stays near 12 dB at every dose.

5.3 Effect of Outer Shape

Every phantom has one of five outer shapes (blob, circle, ellipse, polygon or rectangle). Table 5 and Figure 13 show the mean PSNR for each. DnCNN is best for every shape, between 49.88 dB (blob) and 50.99 dB (ellipse), so the spread across shapes is only 1.1 dB, smaller than the spread of the noisy input (37.09 to 38.96 dB). The ordering of the methods never changes with shape.

Table 5. Mean PSNR (dB) by outer shape (test set).

Outer shape	Noisy input	Median filter	Non-local means	DnCNN
Blob	37.09	41.06	45.13	**49.88**
Circle	37.44	41.60	45.59	**50.12**
Ellipse	38.96	42.37	45.85	**50.99**
Polygon	38.09	41.15	45.43	**50.23**
Rectangle	38.25	40.78	45.87	**50.26**

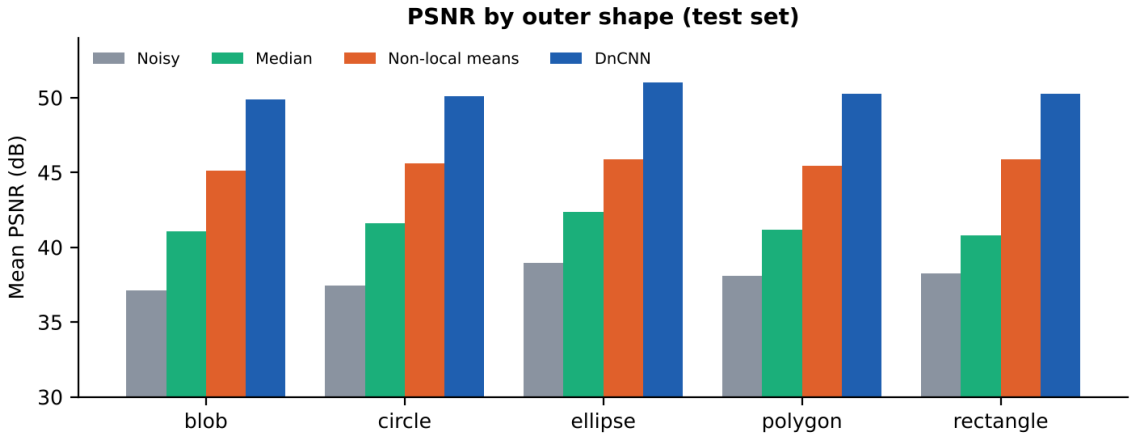


Figure 13. PSNR by outer shape. Mean test PSNR of the noisy input, median filter, non-local means and DnCNN for each of the five outer shapes.

5.4 Training Behaviour

Figure 14 shows the training and validation curves. The training loss falls from 6.96×10^{-3} in epoch 1 to 1.6×10^{-5} in epoch 30. The validation PSNR is bumpy at the start: it stays near 38 dB for the first few epochs (38.45, 37.13, 38.51 and 38.22 dB at epochs 1, 2, 3 and 5) while the network is still learning to separate noise from structure. Then it climbs quickly to 43.34 dB at epoch 10 and 47.98 dB at epoch 15, and flattens out as the cosine schedule drives the learning rate to zero (49.53 dB at epoch 20, 50.26 dB at epoch 25). The best validation PSNR, 50.382 dB, came at epoch 29, and that checkpoint was used for testing. The validation score (50.38 dB) and the test score (50.28 dB) are within 0.1 dB, so I see no sign of over-fitting to the validation set.

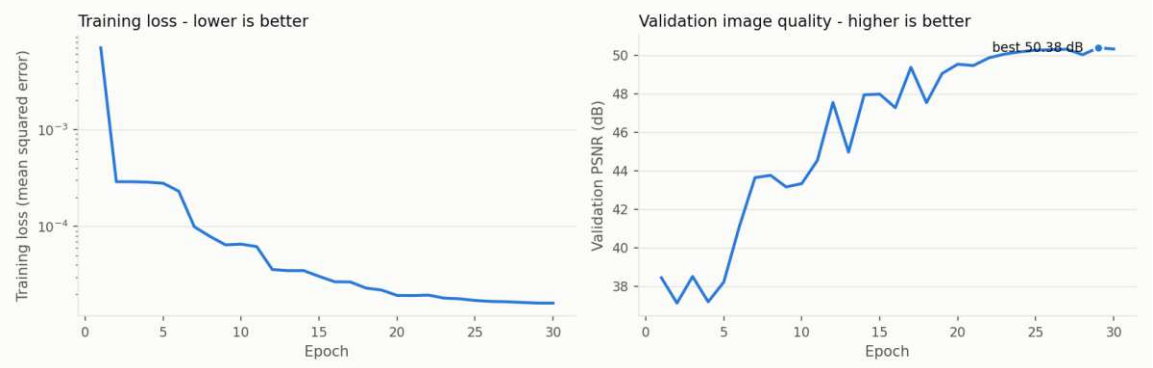


Figure 14. Training curves. Training loss (log scale) and validation PSNR against epoch for the final run.

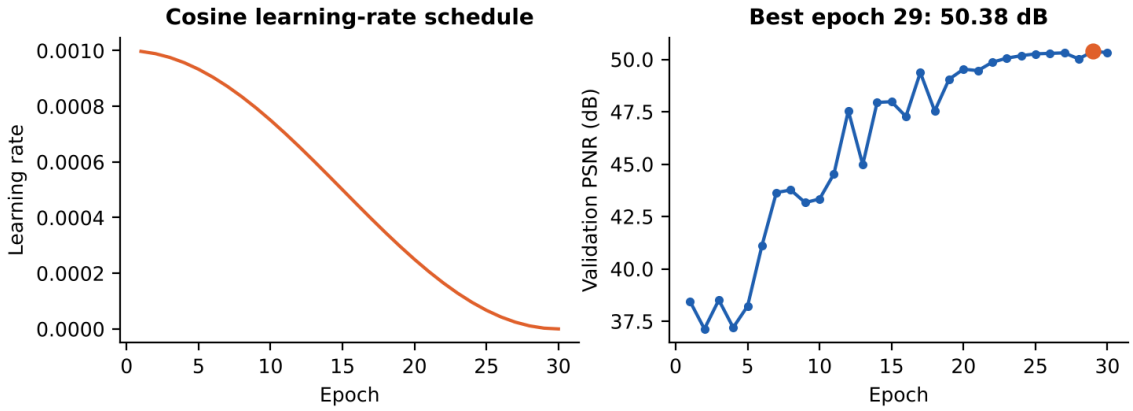


Figure 15. Learning-rate schedule. The cosine decay from 1×10^{-3} to zero (left) and the validation PSNR it produced (right). The late gains come while the learning rate is small.

5.5 Line Profile Across an Edge

PSNR and SSIM average over a whole image, so Figure 16 also shows the pixel values along the centre row of one test image. A good denoiser should follow the clean profile closely on the flat plateaus and at the sharp steps where a material boundary lies. The DnCNN curve lies on top of the clean profile, including at the two steep edges and the small step near pixel 125, while the noisy curve scatters around it. The Gaussian filter follows the noisy curve closely and leaves most of the grain in place.

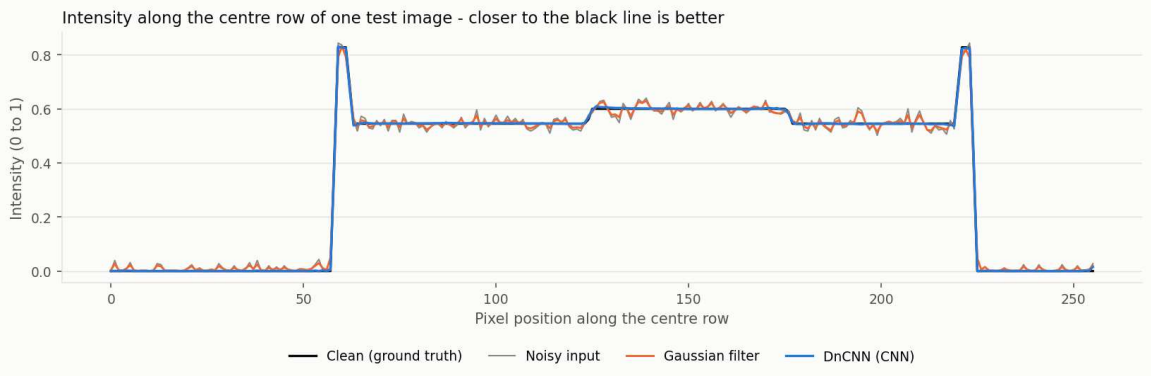


Figure 16. Line profile. Intensity along the centre row of one test image for the clean ground truth, the noisy input, the Gaussian filter and DnCNN.

5.6 Summary of Findings

- DnCNN reached 50.28 dB PSNR and 0.998 SSIM, against 45.56 dB and 0.910 for the best classical method (non-local means).
- It improved the noisy input by 12.34 dB and the best classical filter by 4.72 dB on average, and it won on all 400 test images.
- Its gain was nearly constant across dose classes (12.2 to 12.6 dB) and its lead over non-local means grew with dose.
- It performed similarly on all five outer shapes (49.9 to 51.0 dB).

- Its SSIM spread across images was tiny (standard deviation 0.001), so its results were consistent as well as high.

6. Discussion and Limitations

6.1 Why DnCNN Performed Best

The classical filters apply one fixed rule to every pixel. Gaussian smoothing averages neighbours whether or not an edge lies between them. The median filter removes isolated outliers but also removes small bright or dark features. Non-local means averages similar patches and needs one strength setting, even though the noise level changes from image to image. DnCNN instead learns from 4,200 examples what the noise looks like and what the phantoms look like, so with its 35×35 receptive field it can decide, pixel by pixel, what is noise and what is structure. Predicting the noise rather than the image (residual learning) also makes the task easier, since the network only has to model a small, noise-like signal while the clean structure passes straight through the skip connection. The noise in my data comes from one known process, which is exactly the situation where a learned denoiser has the biggest advantage.

6.2 Limitations

These numbers should be read with the following limits in mind.

1. **Same simulator for training and testing:** The test noise comes from the same Poisson-plus-Gaussian model as the training noise, with the same ranges. I did not try noise types, dose ranges or phantoms the network has never seen, so I cannot say how well it generalises.
2. **Simplified physics.** I added noise directly to the slice: A real scanner adds it to the projections, and the reconstruction turns it into correlated, streaky noise with ring and beam-hardening artefacts. My noise is simpler than that.
3. **Flat, synthetic phantoms:** The phantoms are piecewise-constant shapes with pores and inclusions. Real industrial parts have texture and fine detail, which a denoiser can mistake for noise.
4. **One training run:** I trained a single model with seed 42. Without several seeds I cannot give a confidence interval on the DnCNN score, although the 400-image test set and the per-image win rate suggest the ranking is stable.
5. **Light baseline tuning:** The classical filters were tuned on 40 validation images with a small grid. Heavier tuning, per-image tuning or stronger classical methods such as BM3D could narrow the gap.
6. **Limited comparison:** I did not compare against other deep models (RED-CNN, Noise2Noise, U-Net variants) or against BM3D.
7. **Metric limits:** PSNR and SSIM do not tell you whether a small defect survived. A denoiser could score well and still blur a one- or two-pixel crack. The images are also stored with 8-bit precision, which puts a floor on the error.
8. **Compute:** Training used one NVIDIA T4 GPU in a Google Colab notebook for 30 epochs. I did not explore longer training or a larger model.

7. Conclusions and Practical Guidelines

I built a synthetic industrial CT benchmark of 5,000 phantom slices (4,200 for training, 400 for validation, 400 for testing) with Poisson and Gaussian noise at three dose levels, and compared a 17-layer DnCNN with Gaussian, median and non-local means filters. DnCNN reached 50.28 dB PSNR and 0.998 SSIM on the test set. That is 12.34 dB above the noisy input and 4.72 dB above the best classical filter, and it beat that filter on every one of the 400 images. Its gain was steady across dose classes and outer shapes. In short, when the noise process is known and training data are plentiful, a small residual CNN with about 556 thousand parameters is a much stronger denoiser than the standard classical filters.

Based on this study, I suggest the following guidelines for similar work:

1. Train a residual denoiser (predict the noise, subtract it) rather than a direct image-to-image network; it is simple and was easy to train.
2. Draw the photon count and electronic noise from ranges, not single values, so that the network sees a wide set of dose levels during training.
3. Split the data by index before any processing and keep the test set untouched until the end.
4. Tune the classical baselines on validation images only, and report them with the same metrics as the network.
5. Report more than the mean: the per-image win rate, the spread across images and the results by dose class show whether a gain is reliable.
6. Keep the checkpoint with the best validation score and check that the validation and test scores agree.

Next steps: The most useful extensions are (i) testing the trained network on noise types and dose ranges it has not seen; (ii) simulating the full CT chain (projection, noise on the sinogram, filtered back-projection) so that the noise is correlated as in real scans; (iii) adding texture to the phantoms and, later, testing on real scans; (iv) training with several random seeds to give confidence intervals; (v) adding BM3D, RED-CNN and Noise2Noise as comparisons; and (vi) measuring whether the size and shape of small defects, such as pores and cracks, are preserved after denoising.

Reproducibility and Resources

All code, data and documents are shared in one Google Drive folder (no GitHub repository, as required by the course). The folder contains: (1) a zip file with the code (data generation, training and evaluation notebooks) and the library list; (2) the Training Data (4,200 training and 400 validation images with their metadata); (3) the Testing Data (400 test images with their metadata and the baseline results); (4) the presentation (.pptx); (5) the video presentation; and (6) this article as a PDF.

link: https://youtu.be/n429nHmvkTA

Training and evaluation ran on one NVIDIA T4 GPU in Google Colab. The software used was Python with NumPy, SciPy, scikit-image, pandas, Matplotlib, Pillow and PyTorch. Every random choice (data generation, split, noise, crops, network initialisation) used seed 42, so rerunning the notebooks with the same library versions should reproduce the dataset and the reported numbers up to small GPU differences. Training used the Adam optimiser, mean squared error loss, batch size 16, 128×128 random crops with rotation and flip augmentation, 30 epochs of 262 steps, mixed precision, and gradient clipping at 1.0.

References

1. K. Zhang, W. Zuo, Y. Chen, D. Meng and L. Zhang, "Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017. doi:10.1109/TIP.2017.2662206
2. A. Buades, B. Coll and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 60–65, 2005. doi:10.1109/CVPR.2005.38
3. Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. doi:10.1109/TIP.2003.819861
4. K. Dabov, A. Foi, V. Katkovnik and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.
5. H. Chen, Y. Zhang, M. K. Kalra, F. Lin, Y. Chen, P. Liao, J. Zhou and G. Wang, "Low-dose CT with a residual encoder-decoder convolutional neural network," *IEEE Transactions on Medical Imaging*, 2017. arXiv:1702.00288
6. J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala and T. Aila, "Noise2Noise: learning image restoration without clean data," in *Proc. 35th International Conference on Machine Learning (ICML)*, PMLR vol. 80, pp. 2965–2974, 2018.
7. D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.
8. A. Paszke et al., "PyTorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019.
9. S. van der Walt et al., "scikit-image: image processing in Python," *PeerJ*, vol. 2, e453, 2014. doi:10.7717/peerj.453
10. P. Virtanen et al., "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020. doi:10.1038/s41592-020-0772-5