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.

Bottleneck Width, Loss Design and the Limits of Skip-Free Convolutional Denoising Autoencoders for Simulated Low-Dose CT

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

Indian Institute of Technology Roorkee https://orcid.org/0000-0001-8829-9980

Research Article

Keywords: denoising autoencoder, low-dose CT, bottleneck width, variational autoencoder, DnCNN, non-local means, SSIM loss, latent space

Posted Date: November 16th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11724577/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.

Bottleneck Width, Loss Design and the Limits of Skip-Free Convolutional Denoising Autoencoders for Simulated Low-Dose CT

ANUJ KUMAR 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

Abstract

Convolutional denoising autoencoders (DAEs) are attractive for computed tomography (CT) because they offer a small model and a compact, reusable latent code, but it is unclear whether a skip-free DAE can compete with feed-forward residual networks. We examine this question on 41 abdominal CT volumes from the Medical Segmentation Decathlon (820 real slices, split by volume) and 300 synthetic multi-material phantoms, corrupted with a two-stage noise model (Poisson photon noise followed by Gaussian read noise). We train convolutional DAEs with bottleneck widths of 8 to 256 channels, a variational variant and a composite MSE+SSIM loss, and compare them with a DnCNN-style network and with Gaussian, median and non-local-means filters on identical held-out volumes. DnCNN reached 44.21 dB PSNR and 0.979 SSIM, against 35.07 dB and 0.938 for the best MSE-trained autoencoder; the noisy input scored 35.56 dB and 0.756. The composite loss raised the autoencoder to 35.65 dB and 0.950. Quality was flat between 32 and 256 channels (PSNR spread 0.22 dB) and dropped only at 16 and 8 channels (−0.9 and −2.5 dB relative to width 128). On real slices the width-128 autoencoder lowered PSNR relative to the noisy input in 71% of cases, mainly at thin, high-contrast structures, even though SSIM improved by about 0.12. Latent codes predicted phantom material with 66–84% accuracy (chance 20%) but formed no clusters (silhouette ≈ 0). The autoencoder needed about 34 times less compute per tile than DnCNN, yet was far less accurate. In this setting the reconstruction-through-a-bottleneck design, rather than bottleneck size, appears to limit denoising quality; the role of skip connections remains to be isolated.

Keywords: denoising autoencoder; low-dose CT; bottleneck width; variational autoencoder; DnCNN; non-local means; SSIM loss; latent space

1. Introduction

Computed tomography (CT) exposes patients to ionising radiation, and lowering the dose reduces the number of photons reaching the detector, which raises image noise and artefacts [2,4]. Remedies fall into three families: filtering the projection data before reconstruction, iterative reconstruction, and post-processing of the reconstructed image [4]. Image-domain post-processing is attractive because it needs no access to raw projection data. It is also difficult, because the statistics of noise in reconstructed images are hard to model, and conventional filters tend to remove detail along with the noise [2,4]. Deep networks trained on pairs of noisy and reference images have therefore become a standard image-domain approach [2–4].

Autoencoders offer a different design. A denoising autoencoder (DAE) is trained to reconstruct a clean input from a corrupted copy, which encourages the network to represent the underlying structure rather than the corruption [8,9]. Convolutional DAEs have been used on small medical image sets [10]. Their practical appeal is threefold: the network can be small, its compact latent code is a by-product that may serve other tasks such as anomaly detection, and a variational extension [11] gives a regularised latent space. The design also carries a risk. Every detail of the output must be rebuilt from the bottleneck code, so detail that the code cannot hold is lost. Networks that estimate the noise residual [14] or that add shortcut connections [2,15] let fine detail bypass such a bottleneck.

This paper asks five questions about convolutional DAEs for simulated low-dose CT.

1. Can a skip-free DAE with a compact bottleneck approach a feed-forward residual CNN of the DnCNN type [14]?
2. How does the width of the bottleneck affect reconstruction fidelity and sharpness?
3. Do a variational formulation or a loss that adds a structural-similarity term change the answer?
4. Is the learned code interpretable, in the sense that it organises synthetic tissue materials?
5. How do the models behave across photon-count and read-noise levels, and where do they fail?

We answer them under one controlled protocol: real CT volumes split by volume, synthetic phantoms with known materials, a physically motivated two-stage noise model, identical fixed test sets for every method, and hyperparameter tuning on validation data only. The main findings are as follows. A DnCNN-style network outperformed the autoencoders by roughly 9 dB of PSNR. Bottleneck width had no measurable effect between 32 and 256 channels and mattered only at 16 and 8. A composite MSE+SSIM loss improved the autoencoder, and a variational formulation did not change it. Evaluation on phantoms alone would have suggested that the autoencoder denoises well, whereas on real slices it more often degraded than improved the image in terms of PSNR. The latent code carried information about phantom material that a linear classifier could read, but it showed no cluster structure.

2. Related work

CT denoising. Non-local means [6] is a classic image-domain filter that averages similar patches; as our results show, it is a strong baseline when noise is approximately white. Learning-based methods followed. RED-CNN [2] combines an autoencoder, deconvolution layers and shortcut connections in a residual encoder–decoder network for low-dose CT. Kang et al. [3] apply a residual CNN to directional wavelet coefficients and placed second in the 2016 Low-Dose CT Grand Challenge, whose design and results are described by McCollough et al. [5]. Yang et al. [4] argue that networks trained with a mean-squared-error loss can compromise the visibility of structural detail after aggressive denoising, and propose a Wasserstein generative adversarial network with a perceptual loss. These works typically treat the network as a mapping from noisy to clean images and evaluate it on clinical or simulated low-dose scans.

Denoising autoencoders. Autoencoders learn a compressed representation by reconstructing their input [7]. Vincent et al. showed that corrupting the input and training the network to recover the clean version yields robust features [8,9]. Gondara [10] applied convolutional DAEs to medical images and reported that they work with small training sets. Variational autoencoders [11] add a probabilistic latent space with a Kullback–Leibler term whose weight must be balanced against reconstruction; annealing that weight is a common remedy for unstable training [12,13].

Residual networks and losses. DnCNN [14] predicts the noise component with residual learning and batch normalisation, and a single model can handle an unknown noise level. U-Net-style networks [15] pass high-resolution features across the bottleneck through skip connections. For losses, Zhao et al. [16] compared pixel-wise and structural-similarity-based objectives for image restoration; the structural similarity index (SSIM) [17] compares local luminance, contrast and structure.

Gap addressed here. In the sources we reviewed, convolutional DAEs for CT are usually presented as a model rather than as the subject of a controlled comparison, and we did not find a study that sweeps the bottleneck width of a skip-free DAE over a wide range, compares it with a residual CNN and classical filters on identical held-out volumes, and also reports where it fails. This paper provides such a comparison, including results that run against the expectation that a compact bottleneck acts as a useful prior.

3. Materials and methods

3.1 Data

Real slices. We used the 41 training volumes of Task 09 (spleen) of the Medical Segmentation Decathlon [18], a collection of portal-venous contrast-enhanced abdominal CT volumes. Each volume has 512 × 512 axial slices (31 to 168 slices per volume), an in-plane spacing of 0.61–0.98 mm and a slice thickness of 1.5–8.0 mm. All 41 volumes were used. Intensities were clipped to [−1000, 400] HU, normalised to [0, 1] by min–max scaling computed per volume, and resized from 512 × 512 to 256 × 256 with area interpolation. From each volume we took 20 evenly spaced slices among those that contain body (at least 25% of pixels above −300 HU), giving 820 slices. Volumes were assigned to splits at random with a fixed seed (42), so that no volume contributes to more than one split: 29 training, 6 validation and 6 test volumes (580, 120 and 120 slices). The test volumes are spleen_3, spleen_19, spleen_26, spleen_32, spleen_47 and spleen_63; the validation volumes are spleen_14, spleen_21, spleen_38, spleen_40, spleen_59 and spleen_61. These clinical scans contain their own noise. As in most supervised CT denoising work, we treat them as the clean reference, and area interpolation during resizing attenuates part of that native noise.

Synthetic phantoms. To support controlled analyses we generated 300 phantom slices of 256 × 256 pixels, 60 for each of five material classes: air-like, fat, enhanced soft tissue, calcification and bone (Table 2). Each phantom consists of an elliptical body with a water-like background, five to eight large blobs of its dominant material (radius 7–22 pixels), two or three blobs of other materials, and four to eight small dots (radius 1.5–3.5 pixels) of the dominant material that act as fine-detail probes. Phantoms were rendered at twice the final resolution and area-downsampled. The class label is the dominant material. Phantoms were split 42/9/9 per class (210/45/45).

Table 1. Dataset composition.

Split	Volumes	Real slices	Phantoms	Tiles used for training or validation
Training	29	580	210	3,160 (2,320 real + 840 phantom)
Validation	6	120	45	660 (480 real + 180 phantom)
Test	6	120	45	none (165 full slices are evaluated)
Total	41	820	300	–

Table 2. Phantom materials. Values are normalised attenuation; approximate Hounsfield units follow from HU = 1400 × value − 1000, the mapping of the clipped range.

Material	Normalised value	Approximate HU
Air-like (lung)	0.15	−790
Fat	0.64	−104
Body background (water-like)	0.72	+8
Enhanced soft tissue	0.82	+148
Calcification	0.90	+260
Bone	0.99	+386

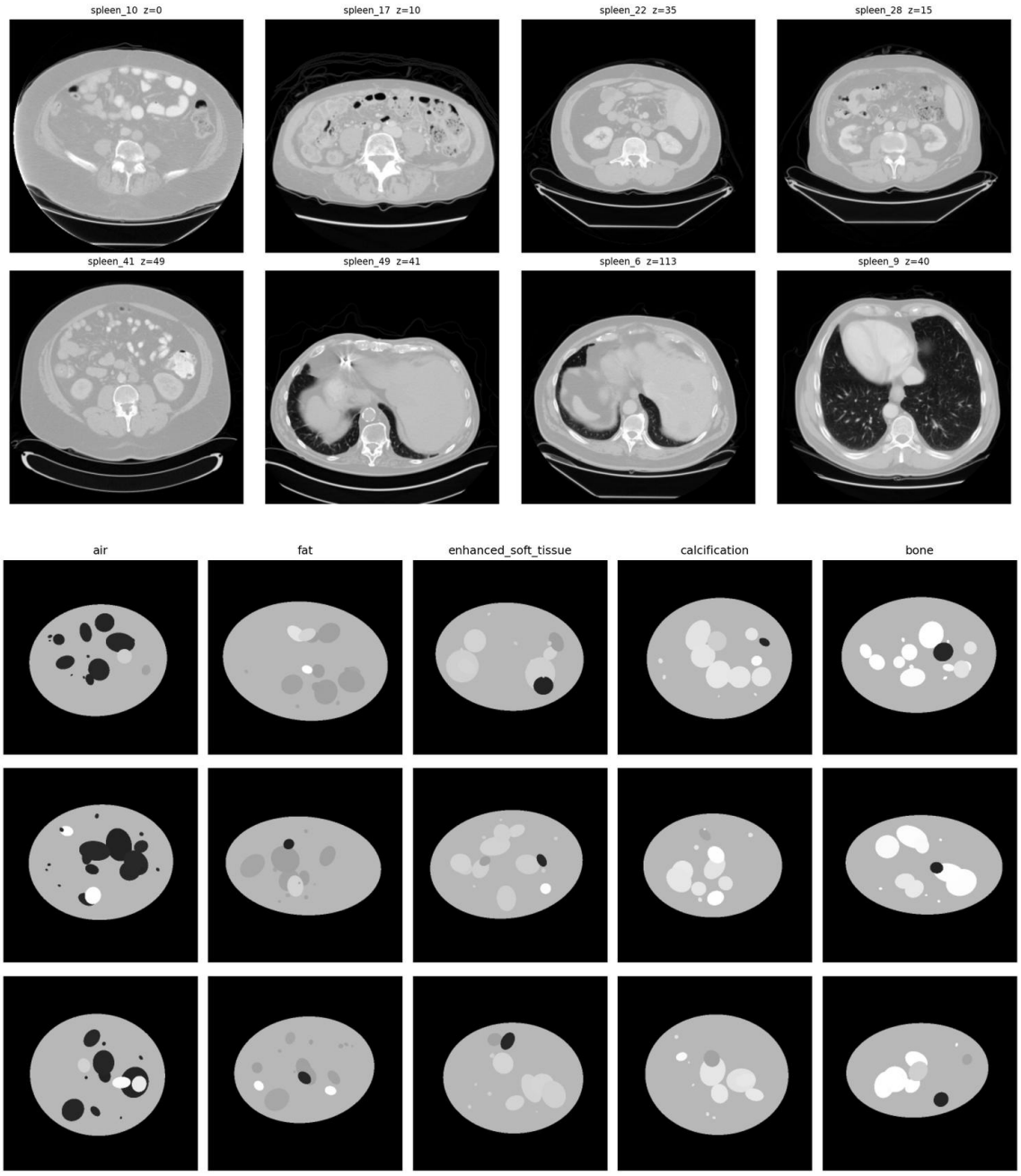


Figure 1. Examples of the data. Top: real slices from eight training volumes after clipping, normalisation and resizing. Bottom: phantoms from the training split, one column per dominant material (air-like, fat, enhanced soft tissue, calcification, bone).

3.2 Noise model

Let $x \in [0, 1]$ be the normalised attenuation image and $\mu = \mu_max \cdot x$ the corresponding line integral, with $\mu_max = 6$. For a reference photon count N_0 per detector element the corrupted image is generated in two stages:

- photon stage: $I = N_0 \cdot \exp(-\mu)$; $\tilde{I} \sim \text{Poisson}(I)$, with $\tilde{I} \geq 1$; $x_P = -\ln(\tilde{I} / N_0) / \mu_max$;

- read-noise stage: $\tilde{x} = x_P + \varepsilon$, with $\varepsilon \sim N(0, \sigma^2)$.

The first stage reproduces signal-dependent photon noise [1]. In normalised units its variance is approximately $\exp(\mu) / (N_0\ \mu_max^2)$, so dense, highly attenuating regions are noisier than air. N_0 is drawn log-uniformly from 5×10^4 to 2×10^5 photons and σ uniformly from 0.01 to 0.03 of the normalised range. Noisy images are not clipped, because clipping would bias air regions. During training a new noise realisation is drawn for every tile at every epoch on the GPU; validation uses one fixed realisation (seed 123). The test set was corrupted once, with fixed seeds, under eight conditions: a mixed condition in which N_0 and σ are sampled per slice from the training distribution; five photon counts (5×10^4, 7.5×10^4, 10^5, 1.5×10^5 and 2×10^5) at $\sigma = 0.02$; and $\sigma = 0.01$ and $\sigma = 0.03$ at $N_0 = 10^5$. Every method was evaluated on the same arrays. This is a simplification: noise is added to the image rather than to projections followed by reconstruction, so it is spatially uncorrelated.

We checked the implementation in three ways (Figure 2). The measured standard deviation of the Poisson component matched theory across intensity bins and rose by a factor of 19.2 between the darkest and brightest populated bins at $N_0 = 5 \times 10^4$. At the lowest dose the brightest pixel still receives about 124 expected photons, and no pixel had a zero count. Finally, the noisy-input PSNR varied by only about 0.2 dB across photon counts at $\sigma = 0.02$, but by about 9 dB across read-noise levels (Section 4.5), so the Gaussian read noise dominates the total noise in this simulation.

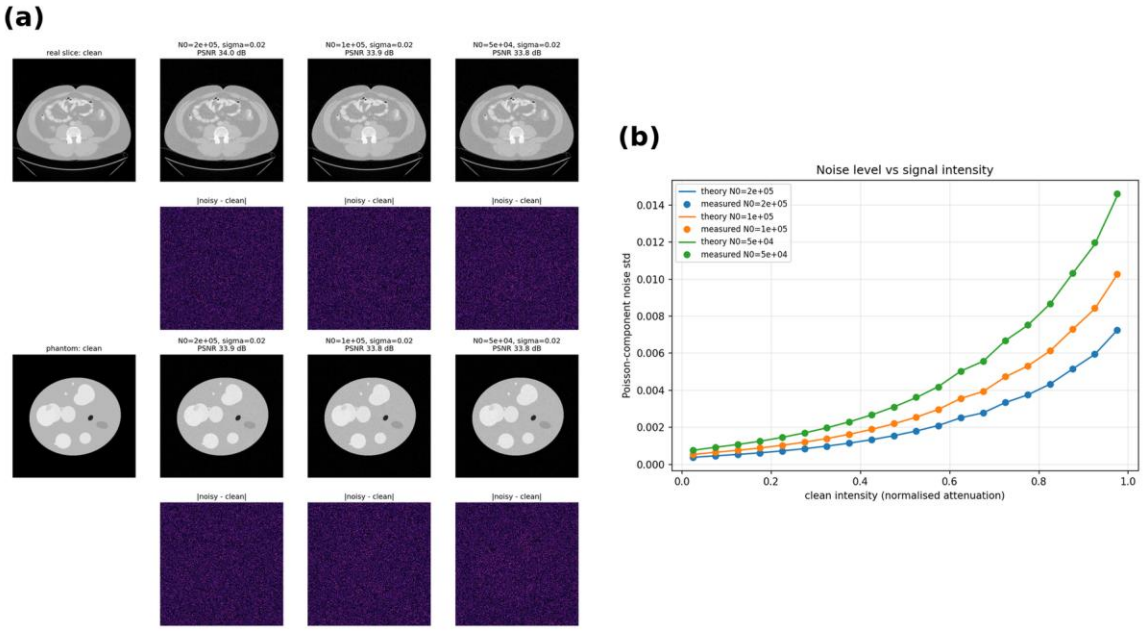


Figure 2. The noise model. (a) A real slice and a phantom, clean and corrupted at $N_0 = 2 \times 10^5$, 10^5 and 5×10^4 with $\sigma = 0.02$, with absolute-error maps. (b) Measured (dots) and theoretical (lines) standard deviation of the Poisson component against clean intensity for three photon counts.

3.3 Tiles, augmentation and reassembly

Slices were divided into non-overlapping 128 × 128 tiles (four per slice), giving 3,160 training tiles and 660 validation tiles (Table 1). Training used random flips and rotations by multiples of 90° (the eight symmetries of the square), applied per tile on the GPU. At test time a slice was cut into tiles with stride

128 (no overlap, four tiles) or stride 64 (nine tiles), each tile was denoised, and the slice was reassembled; overlapping predictions were combined with a Hann-type weighting window. Reassembling unmodified clean slices reproduced them exactly in both modes (Figure 3).

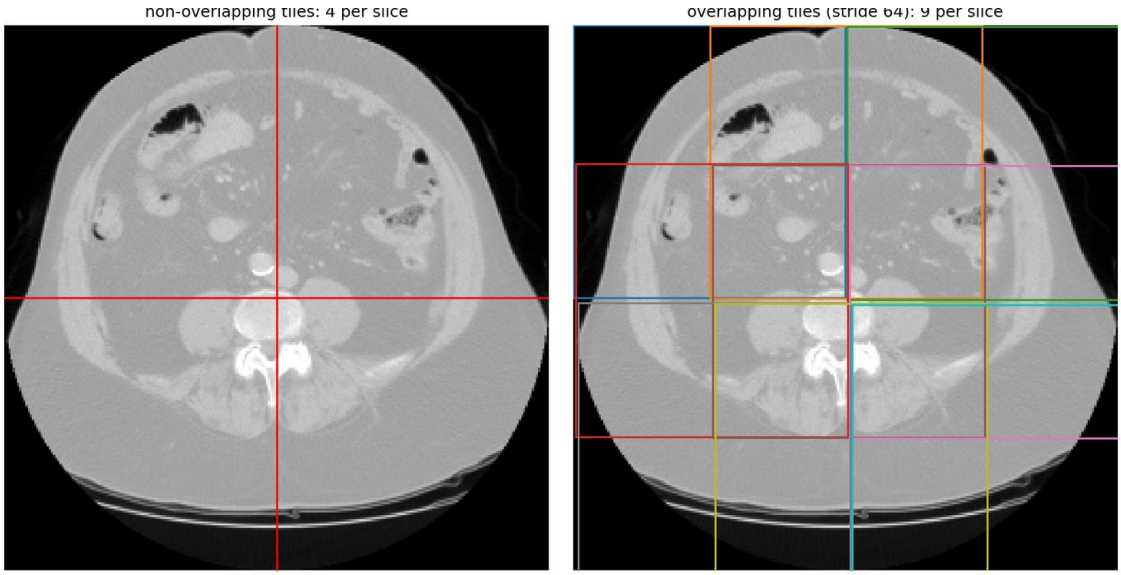


Figure 3. Tiling schemes used at test time: four non-overlapping tiles (left) and nine tiles with stride 64 (right; boxes are offset slightly for visibility).

3.4 Models

Autoencoder. The encoder has four 3 × 3 convolution blocks with stride 2, each followed by batch normalisation [21] and ReLU, with 32, 64, 128 and 256 channels. They reduce a 128 × 128 tile to 8 × 8 × 256. A final 3 × 3 convolution projects this to the bottleneck of W channels (8 × 8 × W values), without activation. The decoder starts with a 3 × 3 convolution from W back to 256 channels with batch normalisation and ReLU, followed by four transposed-convolution blocks (4 × 4 kernel, stride 2; 256 → 128 → 64 → 32 → 32 channels, each with batch normalisation and ReLU) and a 3 × 3 convolution to one channel with a sigmoid output. There are no skip connections. The width W sets the code size, from 512 values per tile (W = 8, a 32-fold compression of the 16,384 pixels) to 16,384 values (W = 256, no compression). Parameters range from 1.13 M to 2.27 M (Table 3).

Variational autoencoder. The VAE shares the encoder trunk. Two 3 × 3 convolutions produce the mean and the log-variance (clamped to [−10, 10]) of the W-channel code. During training the code is sampled with the reparameterisation trick [11]; at test time the mean is used. The loss is the reconstruction loss plus β times the Kullback–Leibler divergence averaged over latent elements, with β increased linearly from 0 to 10^{-4} over the first 20 epochs [13].

Composite loss. The composite objective is MSE + 0.005 (1 − SSIM), with SSIM computed on the 128 × 128 tiles with an 11 × 11 Gaussian window (σ = 1.5) [17]. The weight was chosen so that both terms have a similar magnitude and was not tuned.

DnCNN-style baseline. The network has 17 convolutional layers with 64 channels at full resolution: a convolution with ReLU, fifteen blocks of convolution, batch normalisation and ReLU, and a final

convolution that predicts the noise. The output is the input minus the prediction [14]. It has 0.556 M parameters.

Classical filters. We used Gaussian smoothing, a 3 × 3 median filter and non-local means (patch size 5, search distance 6, fast mode, filtering strength h equal to 0.8 times a noise standard deviation estimated per image with a robust Laplacian-based estimator). Their parameters were selected by PSNR on 80 validation tiles from the grids Gaussian σ_s ∈ {0.3, 0.4, 0.5, 0.6, 0.7} pixels (selected 0.4), median size ∈ {3, 5, 7} (selected 3) and non-local-means h-scale ∈ {0.6, 0.8, 1.0, 1.2, 1.5} (selected 0.8).

Table 3. The learned models.

Model	Bottleneck	Code values per tile	Compression of the 16,384 pixels	Parameters (M)
Autoencoder	W = 8	512	32×	1.131
Autoencoder	W = 16	1,024	16×	1.168
Autoencoder	W = 32	2,048	8×	1.242
Autoencoder	W = 64	4,096	4×	1.389
Autoencoder	W = 128	8,192	2×	1.684
Autoencoder	W = 256	16,384	1×	2.274
VAE	W = 128	8,192	2×	1.979
DnCNN-style	none	–	–	0.556

3.5 Training

All networks were trained with Adam [19] on one NVIDIA Tesla T4 GPU (Google Colaboratory) using PyTorch [23] and mixed-precision arithmetic [22]. The final recipe, selected on validation data (Section 4.1), was as follows. All autoencoders used a batch size of 8 and an initial learning rate of 10^{-3} decayed to 10^{-5} by cosine annealing [20] over 200 epochs (about 79,000 optimisation steps), and the checkpoint with the lowest validation loss was kept (early-stopping patience of 60 epochs, which never triggered). DnCNN used the same optimiser, batch size and initial learning rate for 60 epochs (about 23,700 steps) with a patience of 30, and received no separate tuning. The loss was MSE unless stated otherwise, and the random seed was 42 throughout. Autoencoder runs took 11.5 to 14.6 minutes and DnCNN 29.0 minutes. No choice described here used test data.

3.6 Evaluation protocol

Outputs of every method, including the noisy input, were clipped to [0, 1] before scoring. We report PSNR (data range 1), SSIM [17] (scikit-image defaults [24]), MSE, and sharpness, measured as the variance of the Laplacian of the image (0.0114 for the clean test slices). Sharpness above the clean value indicates residual noise, and sharpness below it indicates smoothing. Each metric is averaged per slice over all 165 test slices and separately over the 120 real and the 45 phantom slices.

For the latent-space analysis, noisy phantom tiles from the validation and test sets (90 phantoms, 18 per class) were encoded. Each phantom was described by the spatial mean of the code per channel, averaged over its four tiles, or in a second descriptor by the mean, maximum and standard deviation per channel. We computed silhouette scores [27] on the full descriptor and on two-dimensional PCA and t-SNE embeddings [26] (perplexity 15), and trained linear probes (standardisation followed by logistic regression) evaluated by stratified five-fold cross-validation [28]. A 32-bin intensity histogram of the same tiles served as a reference descriptor.

Cost was measured as parameters, convolution FLOPs per 128 × 128 tile (PyTorch FlopCounterMode, which counts convolutions only), GPU latency for one slice (four tiles) on the T4, CPU latency, and GPU throughput at batch size 64. For the failure analysis we computed, for each slice, the PSNR gain over the noisy input and its Spearman correlation with slice detail, defined as the Laplacian variance of the clean slice.

Each configuration was trained once. We therefore use the 0.22 dB spread among widths 32 to 256 as a practical yardstick for differences that are too small to interpret, and we report no confidence intervals.

4. Results

4.1 Training recipe

Our first recipe used common defaults: Adam at 10^{-3} with a learning-rate reduction on plateau, batch size 64 and 100 epochs. Under it, the four autoencoders of widths 32 to 256 finished at 33.2–33.9 dB validation PSNR, below the 35.49 dB of the noisy input, with slowly rising curves and epoch-to-epoch fluctuations of several decibels. Because a denoiser that does worse than its input is uninformative for the questions above, we ran two small pilot studies on validation data (Table 4). Cosine annealing alone gave a smoother curve but no gain at equal epochs. Two extra 3 × 3 convolutions at the high-resolution stages added 0.28 dB but made each epoch 1.8 times slower. Reducing the batch size gave the largest gain: going from 64 to 32 raised validation PSNR by 1.17 dB at almost the same time per epoch, and the second pilot selected batch size 8 with a learning rate of 10^{-3}. With the final recipe (batch 8, cosine annealing, 200 epochs) the widths 32, 64, 128 and 256 reached 35.66, 35.65, 35.83 and 35.78 dB on the validation tiles, above the noisy input (Figure 4).

Table 4. Training-recipe pilots on the validation tiles (width 128; noisy-input PSNR on the same tiles is 35.49 dB). The pilots differ in epoch count and compare recipes at a small fixed budget; they are not a ranking of final models.

Run	Architecture	LR schedule	Batch	Initial LR	Epochs	Time per epoch (s)	Best validation PSNR (dB)
Initial recipe	Autoencoder	Reduce on plateau	64	10^{-3}	100	1.9	33.93
Pilot 1a	Autoencoder	Cosine	64	10^{-3}	50	1.9	31.49

Run	Architecture	LR schedule	Batch	Initial LR	Epochs	Time per epoch (s)	Best validation PSNR (dB)
Pilot 1b	Autoencoder + extra convolutions	Cosine	64	10^{-3}	50	3.3	31.77
Pilot 1c	Autoencoder + extra convolutions	Cosine	32	10^{-3}	50	3.5	32.94
Pilot 2a	Autoencoder	Cosine	16	10^{-3}	50	2.5	33.43
Pilot 2b	Autoencoder	Cosine	16	3×10^{-3}	50	2.5	32.89
Pilot 2c (selected)	Autoencoder	Cosine	8	10^{-3}	50	3.5	33.68

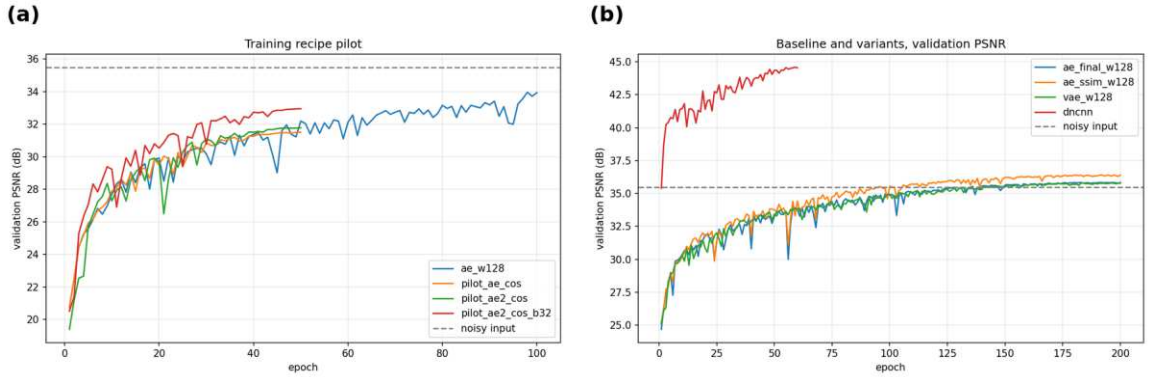


Figure 4. Validation PSNR during training. (a) The recipe pilots of Table 4. (b) Final recipe: the width-128 autoencoder with MSE loss, with MSE+SSIM loss, the VAE and DnCNN. The dashed line is the noisy input.

4.2 Main comparison

Table 5 gives the mixed-dose test results. DnCNN was the best method on every metric (44.21 dB, SSIM 0.979, 8.65 dB above the noisy input), followed by non-local means (42.32 dB, 0.969). Gaussian smoothing gained 0.97 dB, whereas the median filter lost 0.53 dB while raising SSIM from 0.756 to 0.905. The learned autoencoders scored 35.04–35.65 dB. Relative to the noisy input (35.56 dB), the plain autoencoder at width 128 lost 0.49 dB and the VAE 0.52 dB, while the composite-loss autoencoder gained 0.09 dB. In SSIM, by contrast, all three improved on the noisy input by 0.18 to 0.19 (from 0.756 to 0.936–0.950).

The real and phantom columns explain this discrepancy between PSNR and SSIM. On phantoms the autoencoders gained 4.0–4.5 dB over the noisy input (39.85–40.37 against 35.83 dB), whereas on real slices they lost 1.6–2.3 dB (33.17–33.88 against 35.46 dB). SSIM on real slices still rose from 0.800 to 0.915–0.934. The sharpness column shows the mechanism. The clean slices have a Laplacian variance of 0.0114 and the noisy input 0.0174, because noise adds spurious sharpness. DnCNN restored 0.0106 and non-local means 0.0112, whereas all autoencoders produced 0.0076–0.0078, about two thirds of the clean value and comparable to the median filter (0.0074). The autoencoder outputs are therefore cleaner

but smoother than the reference. Figure 5 illustrates this: the autoencoders suppress the noise but blur fine structures and show faint seams at tile borders in the zoomed crops, whereas DnCNN and non-local means retain the structures.

Table 5. Mixed-dose test set (165 slices: 120 real, 45 phantom), no-overlap tiling. Sharpness is the Laplacian variance (clean slices: 0.0114). Learned models are single runs.

Method	PSNR (dB)	SSIM	MSE ($\times 10^{-4}$)	Sharpness	PSNR real (dB)	SSIM real	PSNR phantom (dB)	SSIM phantom
Noisy input	35.56	0.756	3.33	0.0174	35.46	0.800	35.83	0.641
Gaussian filter	36.52	0.802	2.60	0.0124	36.40	0.838	36.86	0.705
Median filter (3×3)	35.03	0.905	4.42	0.0074	32.48	0.907	41.83	0.901
Non-local means	42.32	0.969	0.78	0.0112	40.35	0.962	47.58	0.989
Autoencoder, width 128 (MSE)	35.07	0.938	3.88	0.0076	33.28	0.919	39.85	0.988
Autoencoder, width 128 (MSE+SSIM)	35.65	0.950	3.36	0.0078	33.88	0.934	40.37	0.991
VAE, width 128	35.04	0.936	3.95	0.0076	33.17	0.915	40.03	0.990
DnCNN	44.21	0.979	0.54	0.0106	41.86	0.972	50.48	0.997

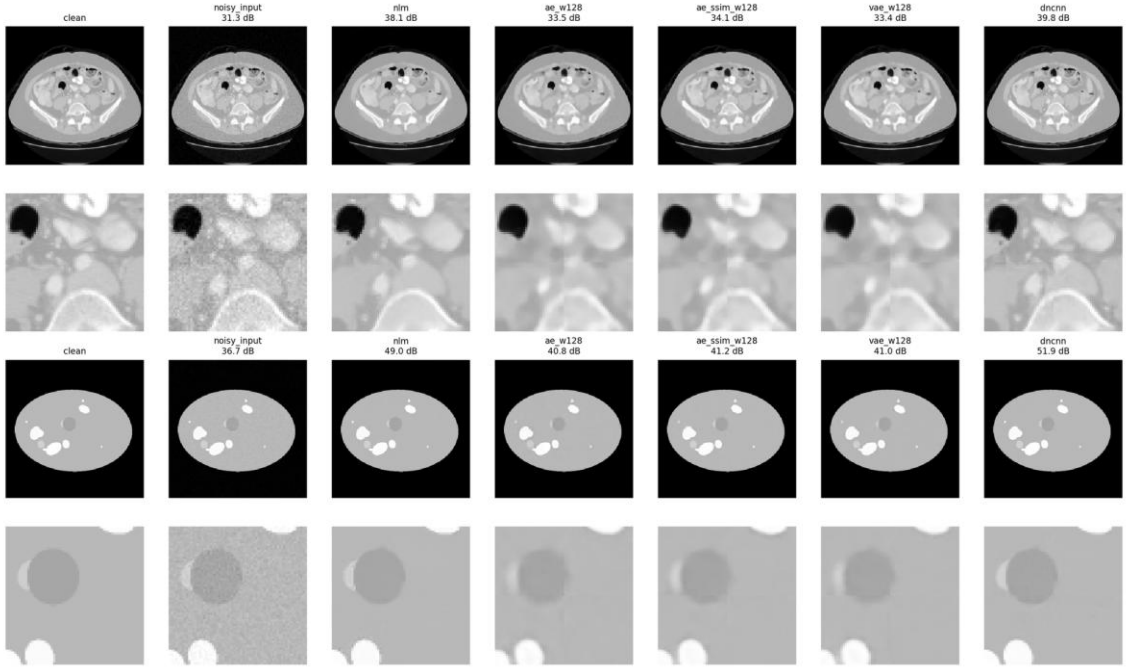


Figure 5. Visual comparison on one real test slice (top two rows: full slice and a 64 × 64 crop) and one phantom (bottom two rows), mixed-dose condition. The PSNR of each full image is printed above it.

4.3 Bottleneck width

Table 6 and Figure 6 show the effect of width. Between 32 and 256 channels, PSNR varied by 0.22 dB (34.85–35.07 dB), SSIM by 0.0024 (0.9355–0.9379) and sharpness by 0.0002 (0.0075–0.0077), with no monotonic trend, so we cannot attribute any difference in that range to width. Quality fell below that range only at the two narrowest bottlenecks. Relative to width 128, width 16 (16-fold compression) lost 0.90 dB of PSNR and 0.011 of SSIM, and width 8 (32-fold compression) lost 2.49 dB and 0.037. The loss at width 8 was larger on real slices (−2.75 dB) than on phantoms (−1.79 dB), which is consistent with real anatomy containing more fine structure for the bottleneck to discard. Sharpness fell only from 0.0076 to 0.0069 at width 8. The shortfall against the clean sharpness of 0.0114 is present at every width, including width 256, whose code has as many values as the input has pixels. The blended-tiling results in Table 8 give the same ordering.

Table 6. Effect of bottleneck width on the mixed-dose test set (autoencoder, MSE loss, no-overlap tiling). Code sizes and parameter counts are in Table 3.

Width	Compression	PSNR (dB)	SSIM	Sharpness	PSNR real (dB)	SSIM real	PSNR phantom (dB)
8	32×	32.59	0.9005	0.0069	30.53	0.8705	38.07
16	16×	34.17	0.9267	0.0075	32.23	0.9042	39.36
32	8×	34.86	0.9362	0.0076	33.00	0.9168	39.82
64	4×	34.85	0.9355	0.0075	32.98	0.9156	39.87
128	2×	35.07	0.9378	0.0076	33.28	0.9189	39.85
256	1×	35.00	0.9379	0.0077	33.13	0.9189	39.98

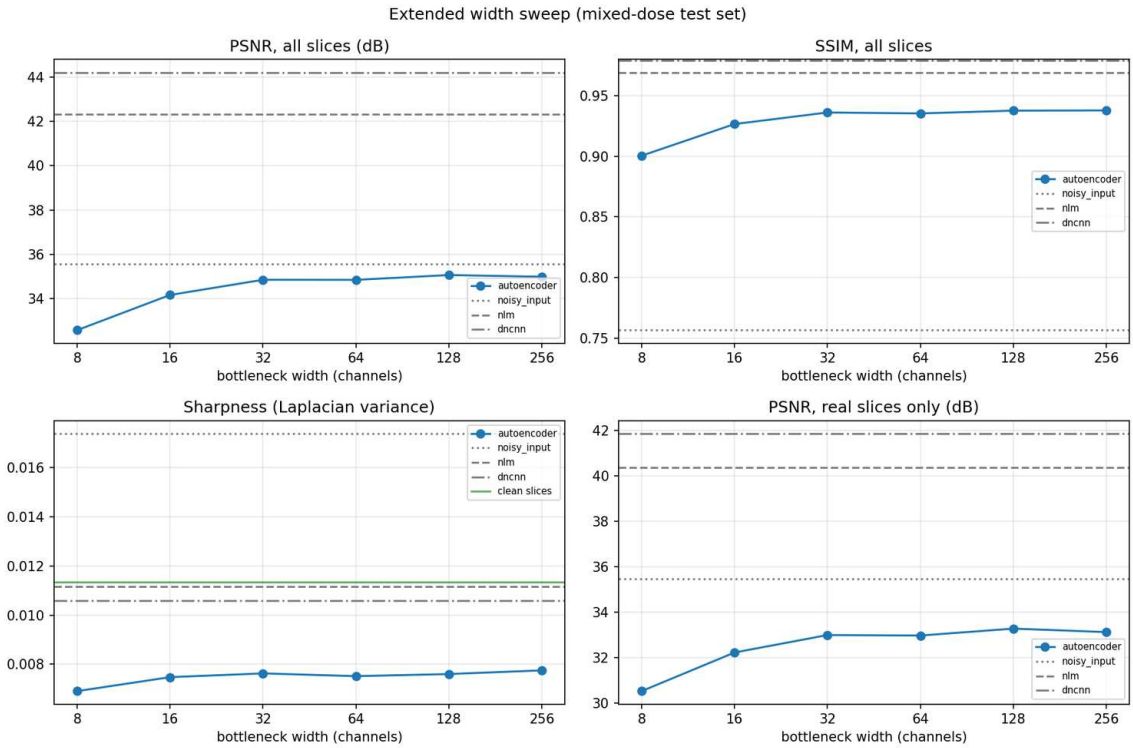


Figure 6. Test PSNR, SSIM, sharpness and real-slice PSNR against bottleneck width, with the noisy input, non-local means and DnCNN as reference lines (and the clean sharpness in green).

4.4 Loss function and variational formulation

At width 128 the composite loss improved every metric over MSE alone: PSNR from 35.07 to 35.65 dB, SSIM from 0.938 to 0.950, real-slice PSNR from 33.28 to 33.88 dB and real-slice SSIM from 0.919 to 0.934, with sharpness essentially unchanged (0.0078 against 0.0076). The PSNR gain of 0.58 dB is more than twice our 0.22 dB yardstick. It cost 26% more time per epoch (4.4 against 3.5 s). The VAE matched the plain autoencoder (35.04 against 35.07 dB; SSIM 0.936 against 0.938), within the yardstick, so the regularised latent space cost no measurable accuracy at $\beta = 10^{-4}$.

4.5 Robustness to photon count and read noise

The test set varied photon count and read noise separately (Table 7, Figure 7). Over the fourfold range of N_0 at $\sigma = 0.02$, the noisy-input PSNR changed by only 0.20 dB and no method changed by more than 0.19 dB (autoencoder 0.01 dB, DnCNN 0.15 dB, non-local means 0.18 dB), because read noise dominates. The read-noise level mattered a great deal: the noisy-input PSNR fell from 40.53 dB at $\sigma = 0.01$ to 31.55 dB at $\sigma = 0.03$. DnCNN and non-local means tracked this change and kept a large margin over the input (DnCNN +6.4, +9.1 and +10.2 dB at $\sigma = 0.01$, 0.02 and 0.03; non-local means +5.1, +7.1 and +8.3 dB). The autoencoders did not: their PSNR stayed between 34.9 and 35.8 dB across the whole range. They therefore degraded the input at low noise (−5.3 dB for the MSE autoencoder and −4.7 dB for the composite one at $\sigma = 0.01$), were roughly neutral at $\sigma = 0.02$ (+0.1 and +0.7 dB), and helped only at $\sigma = 0.03$ (+3.3 and +3.8 dB). Their apparent robustness to dose is the signature of an error floor near 35 dB set by the reconstruction itself, not of effective noise handling.

Table 7. Test PSNR (dB) under the fixed noise conditions. The N_0 columns use σ = 0.02; the σ columns use $N_0 = 10^5$ (the σ = 0.02, $N_0 = 10^5$ result is the 10^5 column).

Method	$N_0 = 5\times10^4$	7.5×10^4	10^5	1.5×10^5	2×10^5	σ = 0.01	σ = 0.03
Noisy input	34.82	34.91	34.95	35.00	35.02	40.53	31.55
Gaussian filter	35.91	35.98	36.03	36.07	36.09	40.88	32.79
Median filter	34.94	34.96	34.99	35.00	35.02	36.56	33.66
Non-local means	41.90	41.98	42.02	42.06	42.09	45.59	39.83
Autoencoder (MSE)	35.08	35.08	35.08	35.09	35.09	35.21	34.87
Autoencoder (MSE+SSIM)	35.66	35.67	35.67	35.67	35.68	35.83	35.39
VAE	35.04	35.05	35.05	35.05	35.05	35.15	34.87
DnCNN	43.92	43.98	44.02	44.06	44.07	46.92	41.77

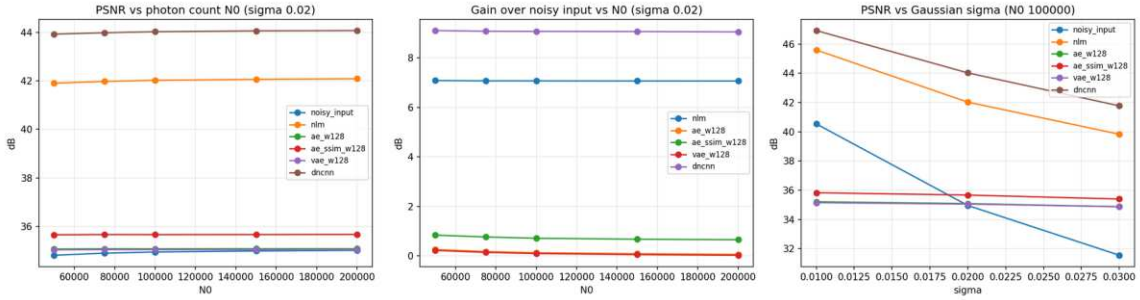


Figure 7. Robustness to photon count and read noise. Left: PSNR against N_0 at σ = 0.02. Middle: gain over the noisy input against N_0. Right: PSNR against σ at $N_0 = 10^5$.

4.6 Tiling artefacts

Overlap-blended reassembly (stride 64) improved every learned model (Table 8): PSNR by 0.14–0.24 dB and SSIM by 0.0005–0.0029, without changing any ranking. The faint cross-shaped discontinuities at tile corners in the autoencoder outputs (zoomed crops of Figure 5) are the visible counterpart. DnCNN also gained 0.18 dB of PSNR but only 0.0005 of SSIM, which suggests that border effects of zero-padded convolutions at tile edges affect all models and are most visible in the autoencoders. Blended results are therefore the fairer measure of the models themselves. We report no-overlap numbers as the main results because they match the training tiling, and we recommend overlap in use.

Table 8. Effect of overlap-blended tiling on the mixed-dose test set.

Model	PSNR, no overlap (dB)	PSNR, blended (dB)	Δ PSNR (dB)	SSIM, no overlap	SSIM, blended	Δ SSIM
Autoencoder, width 8	32.59	32.82	+0.24	0.9005	0.9034	+0.0029
Autoencoder, width 16	34.17	34.36	+0.18	0.9267	0.9286	+0.0019
Autoencoder, width 32	34.86	35.04	+0.18	0.9362	0.9379	+0.0017
Autoencoder, width 64	34.85	35.04	+0.18	0.9355	0.9372	+0.0017
Autoencoder, width 128	35.07	35.24	+0.17	0.9378	0.9394	+0.0017
Autoencoder, width 256	35.00	35.14	+0.14	0.9379	0.9394	+0.0015
Autoencoder, width 128 (MSE+SSIM)	35.65	35.85	+0.19	0.9498	0.9516	+0.0017
VAE, width 128	35.04	35.23	+0.18	0.9357	0.9373	+0.0016
DnCNN	44.21	44.39	+0.18	0.9790	0.9794	+0.0005

4.7 Latent space

Silhouette scores of the phantom material classes were close to zero for every autoencoder (Table 9): −0.009 to 0.015 on the full code, −0.042 to 0.024 after PCA and −0.084 to 0.015 after t-SNE, and the scatter plots show the five classes interleaved (Figure 8a). The intensity-histogram reference, in contrast, scored 0.314 on the full descriptor and 0.698 after t-SNE, so the classes are separable in principle. Yet the material class was linearly readable from the codes: probe accuracy was 65.6–81.1% with mean pooling and 72.2–84.4% with mean, maximum and standard-deviation pooling, against 20% for chance (Figure 8b). The histogram reference reached 100%. Differences between models lie within the cross-validation spread (±3.5 to ±15.1 points on 90 phantoms): the VAE was highest (84.4 ± 12.4%), and the richer descriptor rose from 72.2% to 81.1% between widths 32 and 256, but we do not treat either ordering as established. In short, the codes hold material information in a form a linear classifier can use, but they do not organise it into compact clusters.

Table 9. Latent-space analysis of 90 held-out phantoms (18 per class). Probe accuracy is the mean ± standard deviation over five folds.

Representation	Silhouette (full code)	Silhouette (PCA, 2-D)	Silhouette (t-SNE, 2-D)	Probe accuracy, mean pooling (%)	Probe accuracy, mean+max+std pooling (%)
Autoencoder, width 32	−0.009	−0.024	−0.054	65.6 ± 6.5	72.2 ± 3.5
Autoencoder, width 64	−0.003	−0.010	0.015	67.8 ± 15.1	75.6 ± 8.3
Autoencoder, width 128	0.008	0.024	−0.007	77.8 ± 9.3	77.8 ± 5.0

Representation	Silhouette (full code)	Silhouette (PCA, 2-D)	Silhouette (t-SNE, 2-D)	Probe accuracy, mean pooling (%)	Probe accuracy, mean+max+std pooling (%)
Autoencoder, width 256	0.015	0.005	−0.062	72.2 ± 6.1	81.1 ± 4.4
Autoencoder, width 128 (MSE+SSIM)	0.006	−0.015	−0.084	72.2 ± 9.3	73.3 ± 4.2
VAE, width 128	0.014	−0.042	0.013	81.1 ± 7.5	84.4 ± 12.4
Intensity histogram (reference)	0.314	0.026	0.698	100.0 ± 0.0	100.0 ± 0.0

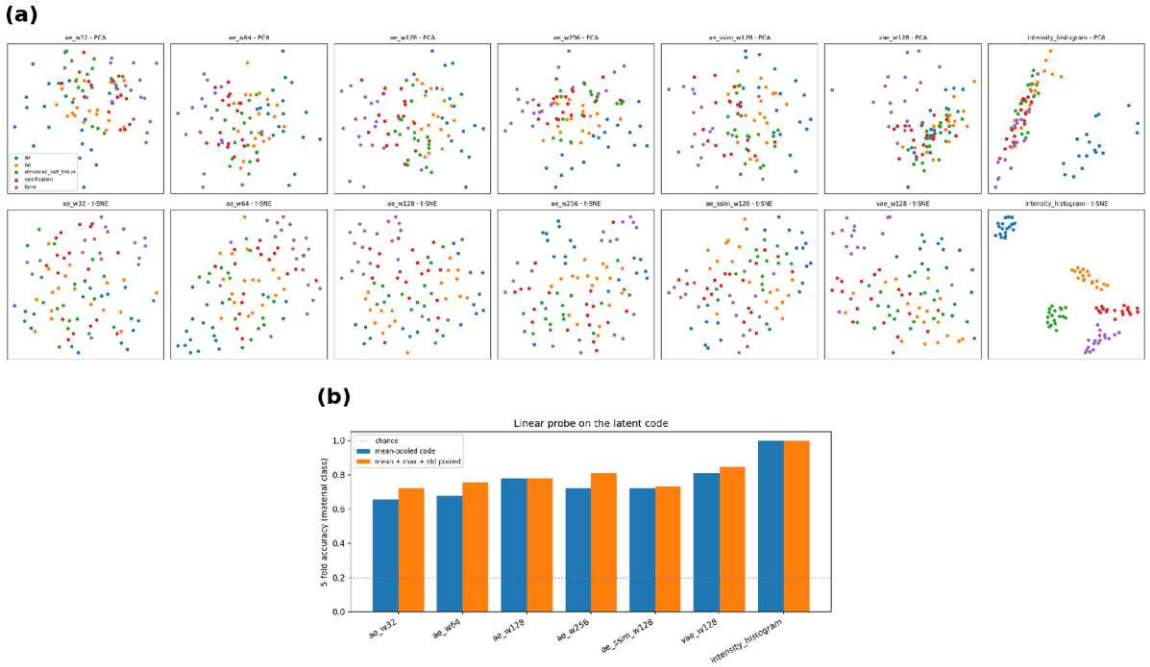


Figure 8. Latent-space analysis. (a) PCA (top) and t-SNE (bottom) of the phantom codes, coloured by material class, for six models and the histogram reference. (b) Linear-probe accuracy for the two descriptors; the dotted line is chance.

4.8 Computational cost

The autoencoders needed 0.48–0.61 GFLOPs per 128 × 128 tile, against 18.16 for DnCNN, a factor of 34 relative to width 128 (Table 10). On the T4 GPU a slice took 2.1–3.5 ms with an autoencoder and 41.7 ms with DnCNN (about 18 times longer); on the CPU 47–78 ms against 1,083 ms (about 22 times). Throughput at batch size 64 was about 2,330–2,390 tiles per second against 227. DnCNN has three times fewer parameters (0.556 M against 1.684 M for width 128) but processes 64-channel feature maps at full resolution through 17 layers, whereas the autoencoder downsamples early, so parameter count is a poor proxy for compute here. The width-32 timings (3.5 ms; 78 ms) were measured first and probably include warm-up, so we do not read any speed difference between autoencoder widths. Training took 11.5–14.6

minutes for 200 epochs for the autoencoders and 29.0 minutes for 60 epochs of DnCNN. For reference, non-local means took about 0.10 s per slice on the CPU in our runs (17.7 s for the 165 mixed-dose slices) and reached 42.32 dB.

Table 10. Cost and quality. FLOPs count convolutions only. GPU latency is for one slice (four tiles) on a Tesla T4; test PSNR and SSIM are for the mixed-dose set.

Model	Params (M)	GFLOPs per tile	GPU ms per slice	GPU tiles per s	CPU ms per slice	Training (min)	Epochs	Test PSNR (dB)	Test SSIM
Autoencoder, width 32	1.242	0.48	3.5	2,331	78	11.5	200	34.86	0.936
Autoencoder, width 64	1.389	0.50	2.1	2,358	49	11.5	200	34.85	0.935
Autoencoder, width 128	1.684	0.54	2.3	2,391	49	11.6	200	35.07	0.938
Autoencoder, width 256	2.274	0.61	2.8	2,352	52	11.9	200	35.00	0.938
Autoencoder, width 128 (MSE+SSIM)	1.684	0.54	2.3	2,388	47	14.6	200	35.65	0.950
VAE, width 128	1.979	0.57	2.4	2,350	49	13.1	200	35.04	0.936
DnCNN	0.556	18.16	41.7	227	1,083	29.0	60	44.21	0.979

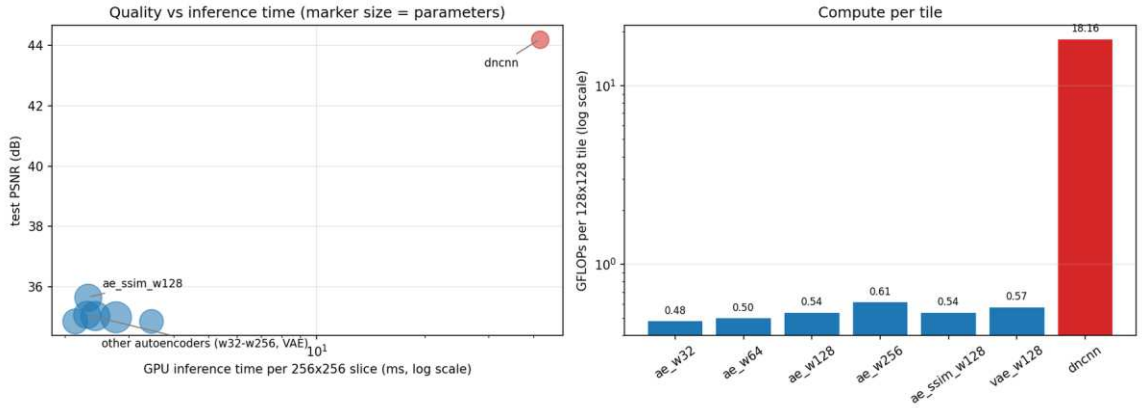


Figure 9. Quality against inference cost. Left: test PSNR against GPU time per slice (log scale; marker size is proportional to the number of parameters). Right: convolution FLOPs per tile (log scale).

4.9 Failure analysis

On the 120 real test slices the width-128 autoencoder produced a lower PSNR than the noisy input in 70.8% of cases (mean change −2.18 dB), and the composite-loss autoencoder in 65.8% (−1.57 dB). DnCNN and non-local means never did (mean gains +6.40 and +4.90 dB; Table 11). The correlation between a slice's detail (Laplacian variance of the clean slice) and the autoencoder's gain was weak (Spearman ρ = −0.21 for both autoencoders), and the gains ranged from about −9 to +3 dB at similar detail levels (Figure 10a). Global detail therefore does not explain the failures. Inspection of the four worst slices for the width-128 autoencoder (Figure 10b) shows the pattern: the error concentrates on thin, high-contrast lines, mainly the edges of the CT couch and bone edges, which the autoencoder blurs (−7.6 to −9.3 dB relative to the noisy input) while DnCNN improves the same slices by 5.7 to 7.1 dB. These four slices are neighbours from one test volume, so they illustrate one failure mode rather than four independent cases, and the couch is not anatomy. They nevertheless show the mechanism by which thin, high-contrast structure is lost, and the same mechanism applies to bone edges.

Table 11. Failure statistics on the 120 real test slices (mixed-dose condition, no-overlap tiling). Gain is the per-slice PSNR change relative to the noisy input.

Method	Slices worse than the noisy input (%)	Mean gain (dB)	Spearman ρ between detail and gain (p)
Autoencoder, width 128 (MSE)	70.8	−2.18	−0.21 (0.020)
Autoencoder, width 128 (MSE+SSIM)	65.8	−1.57	−0.21 (0.022)
DnCNN	0.0	+6.40	+0.16 (0.081)
Non-local means	0.0	+4.90	+0.10 (0.26)

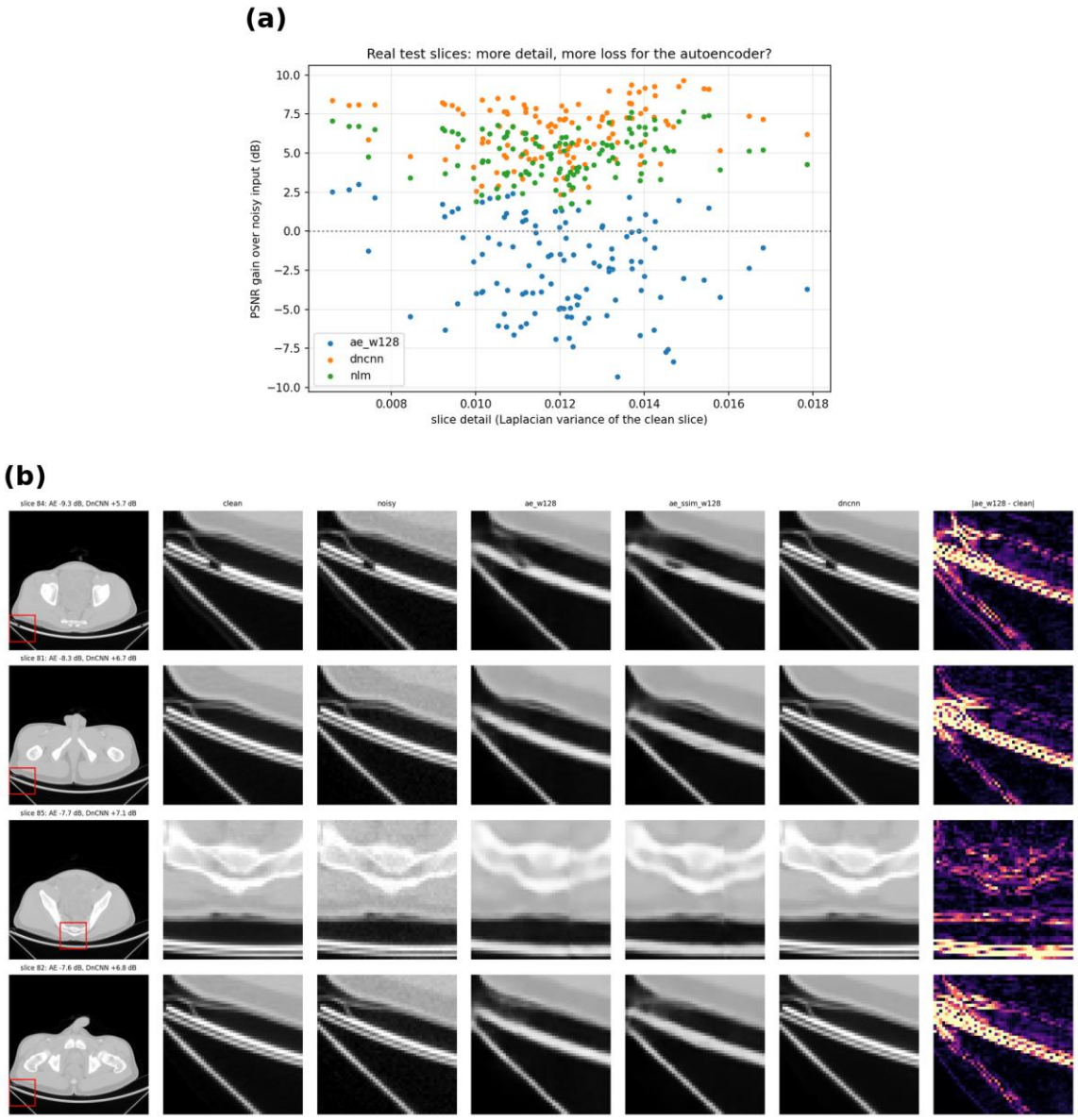


Figure 10. Failure analysis on real test slices. (a) PSNR gain over the noisy input against slice detail for the width-128 autoencoder, DnCNN and non-local means. (b) The four worst slices for the autoencoder: full slice with the cropped region boxed in red, then the 48 × 48 crop of the clean, noisy, autoencoder, MSE+SSIM autoencoder and DnCNN images, and the absolute error of the autoencoder.

5. Discussion

Can a skip-free denoising autoencoder match a residual CNN? Not in this study. The best autoencoder trailed DnCNN by 8.6 dB of PSNR (composite loss) to 9.1 dB (MSE) and by 0.029 to 0.041 of SSIM, and the width-128 autoencoder reduced PSNR relative to the noisy input on 71% of the real test slices. DnCNN achieved this with a third of the parameters and fewer optimisation steps (about 23,700 against 79,000), at roughly 34 times the compute per tile. The result agrees with the design argument of Section 1. A residual network estimates a small noise term and keeps all image detail from its input. An autoencoder

must rebuild every pixel from its code, so reconstruction errors add to the error budget. In our data the autoencoder output had an error floor near 35 dB that did not depend on the noise level (Section 4.5), it was smoother than the reference (Section 4.2), and its losses concentrated on thin, high-contrast structure (Section 4.9). We stress what our experiments do not show. DnCNN differs from the autoencoder in more than residual learning: it works at full resolution throughout and has a different receptive-field geometry. A model that adds skip connections to the same encoder and decoder would be needed to isolate their role.

Bottleneck width. Quality was unchanged from 256 channels (no compression) to 32 (eight-fold compression) and fell only at 16- and 32-fold compression. If the capacity of the code limited the autoencoder, width 256 should have been clearly better than width 32 and sharpness should have depended on width. Neither held: the smoothing and the error floor were present at width 256. The bottleneck is therefore not what limits fidelity in the 32–256 range, and a code eight times smaller than the input was available at no measurable cost, which is useful if the code is wanted for other purposes. Two explanations for the remaining limit are untested: the decoder's ability to regenerate high-frequency detail from strided features, and slow optimisation. In the pilots, doubling the number of weight updates at equal epochs raised validation PSNR by about 1.2 dB, so much longer training than our 200 epochs might narrow the gap, although the final curves had flattened under the cosine schedule.

Choice of metric and the phantom trap. The autoencoders look good by SSIM (0.936–0.950 against 0.756) because SSIM rewards flat, noise-free regions, and the median filter also reaches 0.905. PSNR penalises the structure that smoothing removes. On smooth phantoms the autoencoders improved PSNR by about 4 dB, whereas on real slices they lost about 2 dB. A study that used only phantoms, or only SSIM, would have drawn the opposite conclusion about whether the autoencoder improves on the noisy input. We therefore recommend reporting real anatomy separately from phantoms, and reporting PSNR, SSIM and a sharpness measure together.

Loss design and the variational formulation. The composite loss improved every metric at a cost of 26% more training time per epoch. A plausible reason, which we did not test, is that the SSIM term penalises local structural errors that MSE under-weights, in line with the motivation for structure-aware losses [16]. The variational formulation neither helped nor hurt reconstruction at the small Kullback–Leibler weight used here.

Latent interpretability. The silhouette scores and the linear probes give a split answer. The codes are not organised into material clusters, yet a linear classifier recovers the class with 66–84% accuracy. This is consistent with codes that represent local content spread over space rather than a global material label, but we did not test that explanation, and our descriptors, which average the code over space, may discard relevant structure.

Behaviour across noise levels. The flat PSNR of the autoencoders across noise levels has a practical consequence. Models trained over a range of noise levels degraded the input at the low-noise end (σ = 0.01) by about 5 dB, whereas DnCNN and non-local means improved it at every level. A deployed autoencoder of this type would need to be matched to the expected noise level, or it would risk degrading cleaner scans.

Practical implications. For image quality, residual learning or skip connections should be preferred over a skip-free autoencoder. If a compact latent code is wanted, a width of 32 matched width 256 in our setting.

A composite MSE+SSIM loss is a low-cost improvement, overlap-blended tiling should be used at inference, and evaluation should include real slices and a sharpness measure.

6. Limitations

- **Simulated noise.** Noise was added to images, not to projections before reconstruction, so it is spatially white and lacks the correlation and streaks of real low-dose CT. The models may behave differently on real low-dose scans. The read-noise range also dominates the photon noise, so the dose axis (N_0) was weakly exercised, with changes of about 0.2 dB or less.
- **Reference images are not noise-free.** We used clinical-dose scans, attenuated by area-averaging, as the clean reference. Metrics measure agreement with these references, and their residual native noise limits the attainable scores.
- **One dataset.** All real data come from one collection of portal-venous abdominal CT (41 volumes, six test volumes). The window of [−1000, 400] HU saturates dense bone. Larger multi-site collections such as LIDC-IDRI [25] or the Low-Dose CT Grand Challenge data [5] would test generality.
- **Single runs.** Each configuration was trained once, with no confidence intervals. We avoid conclusions about differences below 0.22 dB. The composite-loss gain (0.58 dB) exceeds that yardstick but is still a single-run result.
- **Unequal tuning and confounded comparison.** The recipe was tuned on the autoencoder; DnCNN reused its optimiser settings, without separate tuning and with fewer epochs, which if anything favours the autoencoder. The autoencoder–DnCNN contrast also confounds several architectural differences (Section 5).
- **Training budget.** The autoencoders were trained for 200 epochs under a cosine schedule. Longer training was not tested.
- **Failure analysis.** It rests on a weak correlation and on four neighbouring slices from one volume. The thin-structure explanation is supported by those examples but not proven.
- **Latent analysis.** It uses 90 phantoms with synthetic labels, spatially averaged descriptors and one t-SNE setting, so its results are indicative.
- **Cost measurements.** Timings come from one T4 GPU and a shared cloud CPU; FLOPs count convolutions only; the first model timed includes warm-up.
- **No task-based evaluation.** PSNR, SSIM and sharpness do not establish diagnostic quality, and nothing here is intended for clinical use.

7. Conclusion

On simulated low-dose CT, a skip-free convolutional denoising autoencoder was much less accurate than a DnCNN-style residual network (35.07 against 44.21 dB PSNR; 0.938 against 0.979 SSIM) and did not, on average, improve on the noisy input in PSNR, although it raised SSIM substantially and needed about 34 times less compute per tile. Bottleneck width between 32 and 256 channels made no measurable difference, and quality fell only at 16 and 8 channels, so the bottleneck was not the limiting factor over that range. A composite MSE+SSIM loss improved the autoencoder by 0.58 dB, a variational formulation did not change it, and overlap-blended tiling recovered about 0.2 dB. The latent codes were linearly informative about phantom material but not clustered, and evaluation on phantoms alone would have given a misleadingly favourable picture. Future work should isolate skip connections with a U-Net-style

variant of the same encoder and decoder, test perceptual or adversarial losses [4], use projection-domain noise simulation and real low-dose data [5], train longer and over several seeds, and evaluate task-based performance.

8. Data and code availability

The imaging data are the public Medical Segmentation Decathlon Task 09 (spleen) collection [18]. The phantom generator, the noise model, the training and evaluation code, the volume lists of every split, the trained weights of the nine final models (autoencoders of widths 8, 16, 32, 64, 128 and 256, the MSE+SSIM autoencoder, the VAE and DnCNN) and all result tables are provided in an accompanying package [https://drive.google.com/drive/folders/1Z4IUAOpOr0JKOenL45gRdUxUKR-v2arv?usp=sharing]. Random seeds were 42 for the volume split and for training, 123 for the validation noise and 1000 to 1007 for the eight fixed test-noise conditions. Experiments ran on Google Colaboratory with PyTorch 2.11, NumPy 2.1.3, scikit-image 0.25.2 and scikit-learn 1.6.1.

Video: https://youtu.be/Vm_AA4L-gQo

References

1. Hsieh J. Computed Tomography: Principles, Design, Artifacts, and Recent Advances. 3rd ed. Bellingham, WA: SPIE Press; 2015.
2. Chen H, Zhang Y, Kalra MK, et al. Low-dose CT with a residual encoder-decoder convolutional neural network. IEEE Transactions on Medical Imaging. 2017;36(12):2524–2535.
3. Kang E, Min J, Ye JC. A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction. Medical Physics. 2017;44(10):e360–e375.
4. Yang Q, Yan P, Zhang Y, et al. Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss. IEEE Transactions on Medical Imaging. 2018;37(6):1348–1357.
5. McCollough CH, Bartley AC, Carter RE, et al. Low-dose CT for the detection and classification of metastatic liver lesions: results of the 2016 Low Dose CT Grand Challenge. Medical Physics. 2017;44(10):e339–e352.
6. Buades A, Coll B, Morel JM. A non-local algorithm for image denoising. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). 2005;2:60–65.
7. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006;313(5786):504–507.
8. Vincent P, Larochelle H, Bengio Y, Manzagol PA. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning (ICML). 2008:1096–1103.
9. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research. 2010;11:3371–3408.
10. Gondara L. Medical image denoising using convolutional denoising autoencoders. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). 2016:241–246.
11. Kingma DP, Welling M. Auto-encoding variational Bayes. In: International Conference on Learning Representations (ICLR). 2014. arXiv:1312.6114.

12. Higgins I, Matthey L, Pal A, et al. beta-VAE: learning basic visual concepts with a constrained variational framework. In: International Conference on Learning Representations (ICLR). 2017.
13. Bowman SR, Vilnis L, Vinyals O, Dai AM, Jozefowicz R, Bengio S. Generating sentences from a continuous space. In: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL). 2016:10–21.
14. Zhang K, Zuo W, Chen Y, Meng D, Zhang L. Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. IEEE Transactions on Image Processing. 2017;26(7):3142–3155.
15. Ronneberger O, Fischer P, Brox T. U-Net: convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI). Lecture Notes in Computer Science, vol 9351. 2015:234–241.
16. Zhao H, Gallo O, Frosio I, Kautz J. Loss functions for image restoration with neural networks. IEEE Transactions on Computational Imaging. 2017;3(1):47–57.
17. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing. 2004;13(4):600–612.
18. Antonelli M, Reinke A, Bakas S, et al. The Medical Segmentation Decathlon. Nature Communications. 2022;13:4128.
19. Kingma DP, Ba J. Adam: a method for stochastic optimization. In: International Conference on Learning Representations (ICLR). 2015. arXiv:1412.6980.
20. Loshchilov I, Hutter F. SGDR: stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (ICLR). 2017. arXiv:1608.03983.
21. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning (ICML). PMLR 37. 2015:448–456.
22. Micikevicius P, Narang S, Alben J, et al. Mixed precision training. In: International Conference on Learning Representations (ICLR). 2018. arXiv:1710.03740.
23. Paszke A, Gross S, Massa F, et al. PyTorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32 (NeurIPS). 2019:8024–8035.
24. van der Walt S, Schönberger JL, Nunez-Iglesias J, et al. scikit-image: image processing in Python. PeerJ. 2014;2:e453.
25. Armato SG III, McLennan G, Bidaut L, et al. The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): a completed reference database of lung nodules on CT scans. Medical Physics. 2011;38(2):915–931.
26. van der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of Machine Learning Research. 2008;9:2579–2605.
27. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830.