Simulator-Based Self-Supervision for Learned 3D Tomography Reconstruction

Onni Kosomaa[*]
Aalto University

Samuli Laine
NVIDIA

Tero Karras
NVIDIA

Miika Aittala
NVIDIA

Jaakko Lehtinen
Aalto University and NVIDIA

Abstract

We propose a deep learning method for 3D volumetric reconstruction in low-dose helical cone-beam computed tomography. Prior machine learning approaches require reference reconstructions computed by another algorithm for training. In contrast, we train our model in a fully self-supervised manner using only noisy 2D X-ray data. This is enabled by incorporating a fast differentiable CT simulator in the training loop. As we do not rely on reference reconstructions, the fidelity of our results is not limited by their potential shortcomings. We evaluate our method on real helical cone-beam projections and simulated phantoms. Our results show significantly higher visual fidelity and better PSNR over techniques that rely on existing reconstructions. When applied to full-dose data, our method produces high-quality results orders of magnitude faster than iterative techniques.

1 Introduction

Computed tomography (CT) is a versatile medical imaging technique for producing tomographic images of body tissues from two-dimensional X-ray projections. Modern systems reconstruct a 3D volume instead of individual 2D slices, so that various cross-sections can be examined easily. For medical CT scans, the most popular mode of acquisition is moving a point-like radiation source and a 2D X-ray detector along a helical trajectory. Reconstructing tomographic volumes from such helical cone-beam (CB) data is a challenging inverse problem whose difficulty is further exacerbated by the increased noise inherent to scans taken using low radiation doses. Yet, as CT uses ionizing radiation, minimizing the dose is of paramount importance when operating with living subjects.

There has been increasing interest in applying machine learning methods to the reconstruction problem. Yet, previous approaches all share a potentially significant shortcoming: They rely on the results of traditional reconstruction algorithms as training targets ("process the input so that it matches this reference reconstruction"), and often also as model inputs ("improve this existing reconstruction"). Clearly, if the targets contain systematic errors, the model will inherit those errors.

We present a deep learning method for CBCT reconstruction with an unprecedented combination of desirable properties: It has the same benefits as other machine learning methods, including high inference speed and the ability to learn from data, but at the same time, it avoids the need for reference reconstructions through the use of a novel simulator-based self-supervised training scheme.

Similar to prior work, we base our method on the classical weighted filtered backprojection algorithm (wFBP) [1] and introduce learned components in crucial points to enable the use of data-driven priors for reducing noise and correcting any remaining approximation errors. Our model reconstructs a full

[*]Part of work done during an internship with NVIDIA Research.

Preprint. Under review.

3D volume in a single forward pass based on the thousands of raw 2D X-rays (i.e., the sinogram) taken from it, guaranteeing tomographic consistency along all axes and improving the quality of coronal and sagittal cross-sections. As no iterative refinement is required, the technique is very fast.

In a crucial step beyond prior work, we train our model to maximize data consistency in the 2D projection domain. This is enabled by incorporating a differentiable CT simulator in the training loop: Intuitively, the output volume is good when simulated X-rays from it look similar to real held-out X-rays from the respective training data scan. The training process is self-supervised in the sense that it requires no reference data besides the noisy 2D projections. This means that the achievable output quality is not limited by the quality of, e.g., clean reference data or a reference reconstruction method. While projection consistency is often employed by iterative methods (e.g., [2, 3]), it has, to our knowledge, not been utilized as a training objective in learned end-to-end reconstruction before. This may be due to the formidable memory and computation requirements posed by the training-time backpropagation through the processing of thousands of input X-rays in each iteration. Indeed, one of our contributions is to show that this is possible in the first place through gradient checkpointing, sparse backpropagation, custom CUDA kernels, etc.

We present a suite of results that demonstrates clear benefit from the self-supervised training of the full sinogram-to-volume model, as opposed to using classical reconstructions as training targets and/or inputs. The benefits are further corroborated by considerable PSNR improvements on synthetic phantoms where a ground truth volume is available. Although our main focus is on low-dose inputs, our method can be applied to full-dose data as well. In these cases, our method produces high-quality solutions orders of magnitude faster than iterative techniques.

Project page with supplemental results is available at `https://users.aalto.fi/~kosomao1/self-sup-ct/`

2 Previous work

Mathematically exact tomography reconstruction algorithms for helical CBCT are known only in the limit of infinitely fine discretization and perfect sampling [4]. In the practical case, all known feedforward methods are approximate. This has given rise to a large family of iterative techniques that directly optimize the volume to minimize a projection consistency loss — difference between synthetic X-rays computed from the volume and the input X-rays — employing sophisticated regularization to deal with the severe ill-conditioning [2, 3]. Unfortunately, iterative techniques are orders of magnitude slower than feedforward methods.

FBP and wFBP. Most non-iterative methods in wide use are based on filtered backprojection (FBP) [5] and particularly its weighted variant (wFBP) [1]. The basic idea is to "pull" the input X-ray projections back from the sensor onto the voxel grid along straight lines — the origin of the term "backprojection" — and average the results, after first applying a linear ramp filter to counteract the overrepresentation of low frequencies that the average would otherwise exhibit. To handle the challenging CBCT case, methods in this family rebin and reweight the projections to approximate a more tractable projection geometry. Despite the much greater efficiency compared to iterative techniques, these approximations cause characteristic artifacts. We seek a feedforward method as efficient as wFBP but without its flaws, and therefore concentrate on prior feedforward methods.

Machine learning methods. A growing body of research applies machine learning to CT reconstruction. To our knowledge, all prior work on feedforward reconstruction is based on supervising a model with the result of an existing (non-learned) technique, typically wFBP. Many techniques take an existing low-quality volume and attempt to remove artifacts and noise from it by a neural network that is trained to match a high-quality reference volume (e.g., [6, 7]). The input and target volumes are typically obtained by running wFBP on matching low-dose and full-dose scans, respectively. Another branch of techniques operates directly on the raw input projections, but still supervises the reconstruction by a wFBP reference (e.g., [8, 9]). Some previous works employ the Noise2Noise principle [10] in an attempt to mitigate the lack of ideal, noise-free target volumes [11, 12, 13]. This entails splitting the input projections into two sets and computing the wFBP target from projections that are not used as model inputs. Some techniques employ adversarial losses to mitigate the blurring caused by minimizing MSE [14, 15, 16]. The apparent increase in detail comes at the risk of introduc-

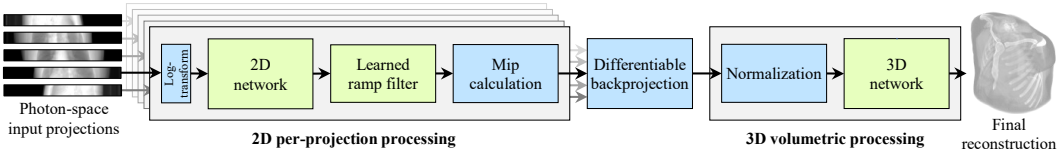
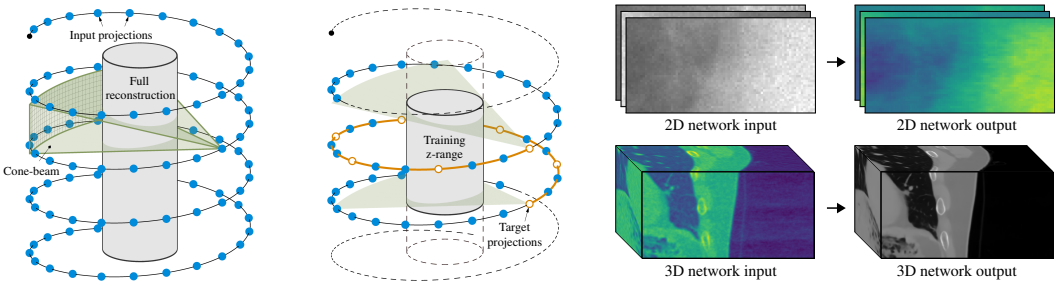


Figure 1: Our pipeline, consisting of a combination of learned (green) and fixed-function (blue) components, reconstructs a 3D voxel volume directly from a set of raw cone-beam projections.



(a) Geometric setup (b) Training (c) Inputs & outputs of the learned networks

Figure 2: (a) The scanner travels along a helical trajectory, capturing cone-beam projections at regular intervals with the positions of the X-ray source depicted by the blue dots. During inference, we reconstruct the full volume using all projections. (b) For training, we randomly choose a small z range (gray) corresponding to a single a rotation of the scanner, and use the projections that intersect it as either inputs (blue) or targets (orange). (c) The 2D network turns the log-space projections into feature maps for further processing. The 3D network outputs the final reconstructed volume.

ing spurious features not actually present in the input data. All these techniques share the common shortcoming that the systematic errors in the wFBP training targets limit the fidelity of the results.

Simulator-based supervision. Using explicit simulation of the physical process that produces the measurements, many classical inverse problem solvers — including iterative CT reconstruction methods — seek a solution that gives rise to the same measurements under the simulation as those observed in reality. We note that this concept has also been applied in other branches of deep learning. For example, neural radiance fields (NeRF) [17] make use of differentiable volume tracing operations to seek a view-dependent 3D volume that explains the observed photographs. Similar ideas have been applied to, e.g., material appearance and illumination recovery [18, 19, 20], neural control of physical simulations [21, 22], neural image restoration through differentiable image corruptions [23], and joint design of optics and neural image reconstruction [24].

3 Our reconstruction pipeline

We combine the efficiency of wFBP with the high fidelity of iterative techniques through a careful combination of learned and fixed-function components, trained with a differentiable CT simulator to encourage synthetic projections from the reconstructed volume to match the actual input projections. Coupled with simple augmentations and a randomized leave-out strategy, this gives rise to consistent, high-quality reconstructions without relying explicit volume priors. We will first walk through our pipeline in detail, and then describe our self-supervised training setup in Section 4.

The overall structure of our reconstruction pipeline, illustrated in Figure 1, is shared with wFBP (cf. Section 2). There are four major deviations: a 2D pre-processing neural network for the raw 2D X-ray images, a learned linear ramp filter, a differentiable CB backprojection operator, and a 3D neural network to produce the final volume. A key benefit of our design is that it can be easily adapted to any acquisition setup with variable number of projections, spacing of the helical trajectory, and radiation doses, with the learned components complementing the fixed backprojection in a data-driven manner.

3.1 Pipeline walkthrough

Geometric setup. We target the standard CBCT setting, illustrated in Figure 2a, where a radiation point source and a sensor grid travel along a helical trajectory. They capture a sequence of 2D projections that measure the attenuation along each ray within the cone as it penetrates the volume. In our experiments, each scan consists of 9,000–15,000 projections stored at 736×64 resolution, so that each volume reconstruction is based on roughly half a gigapixel of X-ray images.

2D network. Given a set of projections as an input, we first feed each of them through a learned 2D neural network that outputs a single-channel feature map in the same spatial resolution as the input. While the network has no other task than to prepare the projections for subsequent processing, we observe that it learns to perform 2D denoising to the inputs (Figure 2c). We visualize the resulting feature map in false color, as it is not guaranteed to be in interpretable units. While we could output a higher number of feature maps from the model, we have observed no benefit in doing so.

Learned ramp filter. Next, to prepare for the backprojection operator that transfers the projection information from 2D to 3D, we filter the projections along the cone-beam rows using a learned ramp filter. This is implemented as a convolution with a one-dimensional kernel that is twice as wide as the input projections. The filter taps are initialized to the inverse Fourier transform of the desired ramp frequency response, following wFBP, but they are treated as learnable parameters during training. This allows the pipeline to adjust the frequency spectrum of the projections to facilitate the later operations; in practice, we have observed that the ramp filter changes very little during training.

Differentiable backprojection. To transfer the 2D feature maps into the 3D volume, we pass each of them to a fixed-function differentiable backprojection operator that accumulates log-space attenuation to all voxels intersected by the cone-beam in question. In contrast to wFBP, we perform backprojection directly using cone-beam geometry — i.e., along lines that connect sensor pixels with the radiation source — without rebinning to parallel beam projections first.

The backprojected value for a voxel is obtained by first projecting its center onto the 2D sensor using the radiation source as the center of projection, and then interpolating along the sensor's pixel grid. The projection lines converge onto the radiation source, which causes the local frequency content of the backprojected signal to vary significantly: Close to the source, the projection lines are packed densely, while near the sensor their spacing is sparser. As using a voxel grid fine enough to capture the densest beam bundles is impractical, we carefully anti-alias the result to ensure the backprojected signal can be represented by the voxel grid faithfully, following the standard Shannon–Nyquist sampling and reconstruction theory [25]. In practice, we efficiently approximate the required pre-filtering through *mipmapping* [26], a technique common in computer graphics. Further details are given in Appendix C. Following Stierstorfer et al. [1], we apply a cosine-tapered weight function to detector rows and normalize each voxel by dividing the accumulated value by the total weight of contributing backprojections. Compared to wFBP, our wider taper ($Q = 0.8$) and lack of rebinning results in stronger spiral artifacts, but we find them to be easily corrected by subsequent processing.

Learned 3D processing. As the final step of the reconstruction pipeline, we pass the volume through a learned 3D network that outputs a voxel grid in log-attenuation space (Figure 2c). As its receptive field is relatively large, it can correct for blur, spiraling, and other artifacts caused by the earlier stages. In practice, the voxel resolution of our final reconstruction varies between 576×576×272 and 1024×1024×644, depending on the case.

3.2 Network details

We use U-Nets [27], i.e., autoencoders with skip connections, for the 2D and 3D networks as they have been shown to perform well on a variety of tasks including denoising and removal of image artifacts [28]. Our 2D network architecture matches the one used by Lehtinen et al. [10]. The 3D network is similar, except that the intermediate channel counts have been halved to conserve memory, and 3×3 convolution kernels have been replaced with 3×3×3 kernels to enable volume processing. Network weights were initialized using He initialization [29].

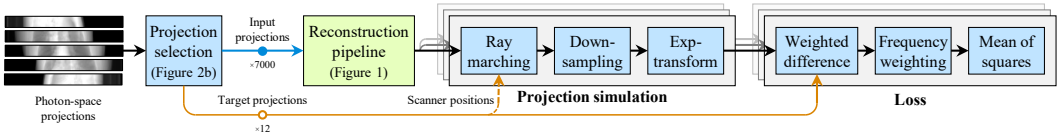


Figure 3: We employ a self-supervised training setup that does not require reference data. In each iteration, we select a handful of target projections from a randomly chosen rotation of the scanner and feed the remaining ones as inputs to our pipeline (Figure 1). Based on the resulting volume, we simulate projections from the same locations as the targets, and minimize their weighted difference.

4 Training

We now turn to training the learned 2D neural network, ramp filter weights, and 3D neural network. We train the pipeline in an end-to-end fashion, meaning that only the fidelity of the final reconstruction provides the signal that guides the components to a joint optimum. We describe the overall architecture of the self-supervised loss function and training loop in Section 4.1, deal with photon noise in the training data in Section 4.2, and discuss projection simulation and the remaining challenges in Section 4.3. The training process is illustrated in Figure 3.

4.1 Simulator-based self-supervision

To enable training without known reference 3D volumes, we combine a projection consistency loss, similar to many iterative reconstruction techniques, with a leave-out strategy that resembles cross validation: A volume reconstruction is considered faithful if left-out real X-rays look the same as simulated X-rays computed using the same scanner position. A key benefit of this approach is that it requires no reference data in either 2D or 3D.

Each training iteration begins by selecting a random slab of the volume from a scan in the dataset, and identifying the set of X-rays whose backprojections overlap with the slab. The set is then randomly split into a large set of *input projections* and a small set of *target projections* (Figure 2b). The input projections are fed to our reconstruction pipeline, resulting in a 3D volume. We then compute, for each target projection, a virtual X-ray using the known positions of the radiation source and sensor using a differentiable X-ray simulator (Section 4.3). The final loss function is the mean squared error between the simulated projections and left-out target projections. As all components in the pipeline are differentiable, the gradients of the learnable parameters are computed using standard backpropagation, and used to train the networks with the Adam [30] optimizer.

4.2 Noisy target projections

A subtle point not addressed in the discussion above is that as we train with real X-ray data, we do not have noise-free projections at hand, i.e., the target projections contain all forms of noise inherent to X-ray imaging. Fortunately, this noise fulfills the requirements for the *Noise2Noise* principle [10] to apply: It is zero-mean and uncorrelated between the inputs and outputs, as noise realizations between different X-ray images are independent. Accordingly, the noise "averages out" when the model is trained with the noisy targets and L_2 loss, and given enough data, the model converges to the same optimum as though it were trained with clean targets.

The requirement that the noise in the model inputs must be uncorrelated with training targets is also the reason behind the leave-out strategy of not using target projections as model inputs. If this is not met, the quality of the results deteriorates dramatically, as we demonstrate in Appendix B. While the leave-out strategy leaves gaps in the set of input projections during training, we have found the impact of these gaps to be negligible as long as the number of target projections is kept small. For each training iteration, we use approx. 7000 input projections and 12 target projections.

An important detail to consider is that the noise in the acquired 2D projections is zero-mean in photon intensity, but not in log-attenuation because of the nonlinear transformation. As such, to use noisy training targets, we must compute the L_2 loss in photon intensity space. This, however, has the severe problem that pixels with high photon counts, i.e., low attenuation, have exponentially larger effective weight in the overall loss function than highly attenuated pixels, which is at odds with the practice of viewing the results in log-attenuation space. Therefore, we scale the photon-intensity L_2 loss so that

each pixel's contribution to the overall loss is proportional to what it would have been if the loss were computed in log-attenuation space. The resulting loss function is

$$\mathcal{L}(\hat{X}, \hat{Y}) = \left\| w(\hat{X}) \odot (\hat{X} - \hat{Y}) \right\|_2^2, \quad w(\hat{X}) = 1/\hat{X}, \tag{1}$$

where $\hat{X}$ and $\hat{Y}$ are the simulated projection and the training data projection in photon intensity space, respectively, and $\odot$ denotes element-wise multiplication. The weighting function $w(\hat{X})$ is proportional to the inverse derivative of the exp-transform. Importantly, we treat the gradients of $w(\hat{X})$ as zero to prevent training from attempting to just minimize this weight [10].

We have so far assumed that the same projections are used as both inputs to the reconstruction pipeline as well as training targets. However, we may have paired projections with different dosages available at training time. Such paired data can be obtained by, e.g., acquiring higher-dose projections and adding noise to them that simulates a lower-dose scan. In this situation, we can use the higher-dose projections as $\hat{Y}$ in Equation 1, while still using the lower-dose projections as input.

4.3 Projection simulation, augmentations, and feasible implementation

The projection simulator follows the underlying physical measurement process: For each pixel in the projection, we march a ray through the volume and accumulate the attenuation coefficient. We anti-alias the operations through supersampling and trilinear interpolation, and low-pass filter the target projections to control for ringing. To accelerate the learning of high spatial frequencies, we emphasize them by high-pass filtering the difference images in the loss. Similar to many previous works (e.g., [31]), we also found it practically beneficial to use our custom mipmapping-based backprojection operator for computing the gradient of the ray-marching operation.

Our setting allows for lightweight geometric data augmentations by simply transforming the associated metadata of sensor device positioning in relation to the volume. For each training sample, we randomize the z-axis rotation and add a sub-voxel position jitter to discourage overfitting to orientations or regular grid patterns. We also randomly scale the overall intensity of the projections.

The amount of computation involved in producing a single reconstruction makes a naive implementation of end-to-end training infeasible: For example, the intermediate results required for gradient backpropagation through the thousands of invocations of the 2D network are far too large to be stored in memory, and even if an infinite amount of memory were available, training would be unacceptably slow. To make the training process practical and efficient, we use gradient checkpointing, sparsified backpropagation, and a two-level training loop. For efficiency, we implement our differentiable projection and backprojection operations as custom PyTorch [32] operators using a combination of custom CUDA code and a modified version of the texture lookup function in Nvdiffrast [33]. Please see Appendix C for further details on training, including signal processing and augmentations.

5 Results

Instead of attempting an exhaustive comparison to all major previous reconstruction algorithms, our main goal in evaluation is, in a common architectural setting, to understand the differences between major *families* of feedforward techniques that differ by the types of input data and the data the model is supervised on. We conceptualize the design space through two axes:

1. **Model input:** Does the model take an existing 3D volume as input, or does it operate directly on raw 2D projection data?
2. **Training target:** Is the model trained to match an existing 3D volume, or does the training employ self-supervision based on raw 2D projection data?

Out of the four possible combinations, previous work covers two: taking a reconstructed volume as input and using another reconstruction as a training target (volume-to-volume), and taking projections as input and using a reconstructed volume as training target (projection-to-volume). Our approach (projection-to-projection) is the first to use raw X-rays as both model inputs and training targets without relying on externally-provided reference reconstructions. Compared to the other three choices, we show that this results in significantly improved quality, both quantitative and qualitative.

We evaluate our method on both synthetic and real-world medical chest scans. Synthetic data is included to enable quantitative performance measurement — this is only possible with a known,

Table 1: Quantitative reconstruction quality of different methods using synthetic data.

	Method				Low-dose inputs		Full-dose inputs		Runtime
	Pipeline	Self-sup.	Input	Target	PSNR (dB)	RMSE	PSNR (dB)	RMSE	
A	wFBP [1]		Proj.	–	23.23	0.1379	33.41	0.0427	90s
B	IR-TV [2]		Proj.	–	35.49	0.0336	42.12	0.0157	∼1h
C	RED-CNN [6]*		Vol.†	Vol.†	38.18	0.0247	–	–	106s
D	Our 2D U-Net*		Vol.†	Vol.†	39.17	0.0220	–	–	92s
E	Our 3D U-Net		Vol.†	Vol.†	39.77	0.0206	–	–	93s
F	Our pipeline		Proj.	Vol.†	39.38	0.0215	–	–	27s
G	**Our pipeline**	✓	Proj.	Proj.	**41.11**	**0.0176**	**44.21**	**0.0123**	**27s**
H	Our 3D U-Net	✓	Vol.†	Proj.	39.59	0.0210	–	–	93s

* Processes each z-slice of the volume independently. † Relies on wFBP to obtain the inputs and/or targets.

noise-free volume. Real-world data enables us to qualitatively confirm that our method scales up to the complexity of real subjects and scanner setups. To avoid issues inherent to the domain gap, all methods are trained separately on synthetic and real data. We run the training in parallel using 8 NVIDIA A100 GPUs for 2.5 days (480 GPU hours) for the synthetic dataset, and 8 days (1536 GPU hours) for the real dataset to accommodate for the higher resolution. See Appendix A for details on comparison methods, datasets, and metrics.

To facilitate visual comparison, we slightly blur the 3D reconstructions produced by our method to better match the visual look of the ground-truth and wFBP results using a hand-tuned Lanczos filter. Full versions of result images, including neighboring slices and an interactive HTML viewer, are available as supplemental material. In addition to the results presented below, Appendix B contains further experiments including validation of the self-supervised training setup, target exclusion, and photon-space loss function, as well as additional result images.

5.1 Low-dose inputs

We begin by presenting results using low-dose inputs (10% of full dose) and full-dose targets (where applicable). Table 1 reports the PSNR and RMSE for a representative subset of the methods, computed against noise-free synthetic ground truth volumes, and Figure 4 shows example reconstructions on real data (omitted methods behave consistently with the PSNR measurements). See Appendix B for the corresponding reconstructions using synthetic data.

Baselines. We establish baselines with two traditional methods, wFBP (METHOD A) and total variation regularized iterative reconstruction (IR-TV [2], METHOD B), as well as the machine learning technique RED-CNN ([6], METHOD C). Neither wFBP nor IR-TV make use of training data, except that we tune the parameters of IR-TV to maximize PSNR over the synthetic dataset. As seen in Figure 4, wFBP struggles with low-dose inputs, particularly in the soft tissue windows. While IR-TV achieves a significantly higher PSNR and a noticeably better visual result on real data, both techniques' reconstruction quality is limited in the low-dose regime.

RED-CNN is the simplest point in our design space for learned methods: It processes 2D slices of low-dose wFBP reconstructions and uses 2D slices of full-dose wFBP reconstructions as training targets. Due to the focus on 2D slices, the method cannot exploit 3D structure. Regardless, it improves PSNR by several decibels over the baseline IR-TV. Finally, to control for the effects of the network architecture, we implemented a version of RED-CNN with our 2D U-Net structure (METHOD D), further improving the result by 0.99 dB.

Volume-to-volume and projection-to-volume. We now move on to machine learning techniques that process and produce 3D volumes. To take a minimal step from METHOD D and gauge the potential benefit of providing the model a view to the full volume instead of just slices, we trained a denoiser network based on our 3D U-Net architecture (METHOD E) using full-dose wFBP reconstructions as targets and wFBP low-dose reconstructions as inputs (volume-to-volume). Effectively, this is a 3D version of RED-CNN, similar to [7]. This improves PSNR further by 0.60 dB, confirming that access to 3D structure improves denoising results numerically. As seen in Figure 4, METHOD E achieves a relatively good visual result on real data, with some blur and noise remaining. Appendix B highlights

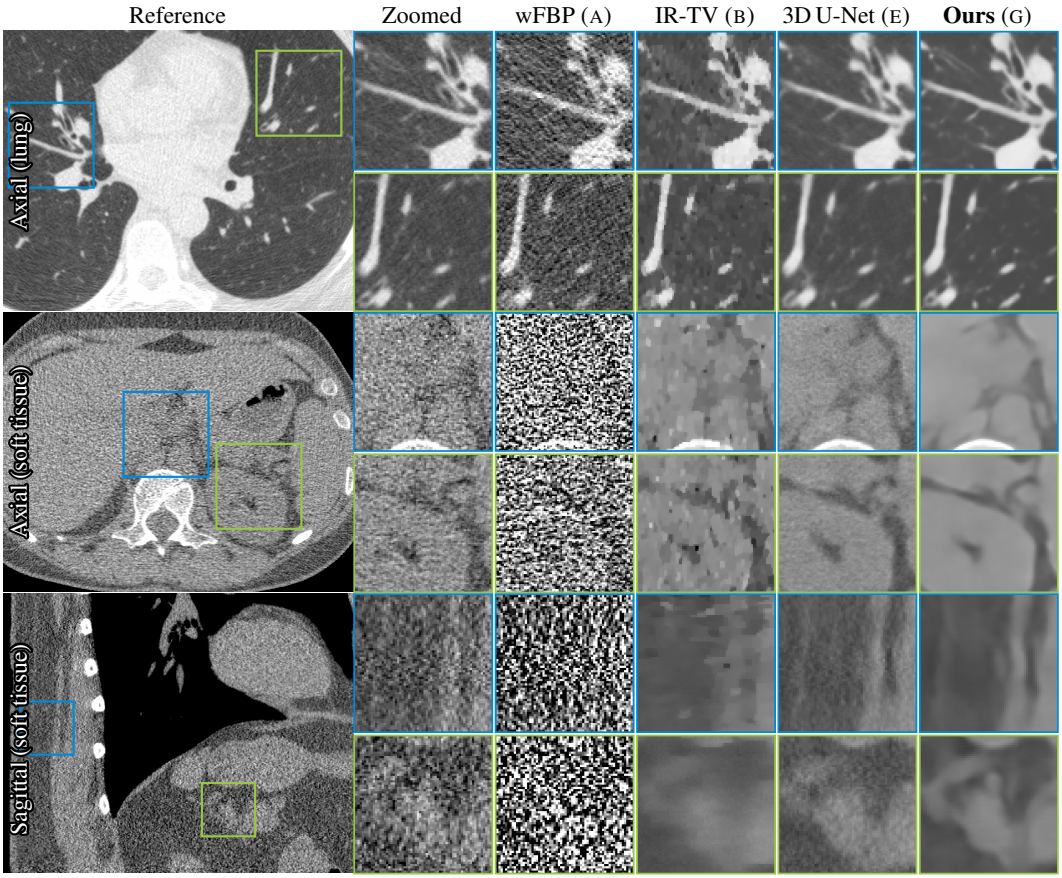


Figure 4: Low-dose reconstructions of real-world data with different methods. For reference, full-dose wFBP (two leftmost columns) is shown in lieu of noise-free ground truth that is not available. Soft tissue display intensity range (window) is set to [−300, 300] Hounsfield units (HU) and lung window to [−1350, 150] HU. Full images and neighboring slices are available in the supplement.

the visual importance of denoising over the full 3D volume instead of processing slice-by-slice, which results in strong artifacts along the non-axial cross-sections.

Next, in METHOD F, we investigate whether additional performance is available by switching to using 2D projections as model inputs instead of volumes reconstructed by wFBP, similar to previous end-to-end approaches [8, 9]. This configuration consists of the entire pipeline depicted in Figure 1 that is trained using full-dose wFBP reconstructions as targets. Interestingly, though the model's inputs are richer than those of METHOD E, quantitative performance decreases noticeably. This suggests that the beginning of our pipeline is, up to and including backprojection, sufficiently different from wFBP to make the task of matching the full-dose wFBP target volume harder to achieve.

Our method: projection-to-projection. Up to this point, the training targets for all methods have been full-dose reconstructions made using wFBP. We now turn to our full method, METHOD G, that uses low-dose projections as inputs and is trained using full-dose projections as targets as described in Section 4.1. As seen in Table 1, this improves PSNR significantly to 41.11 dB, surpassing the best comparison method by 1.34 dB.

Our reconstructions on real data (Figure 4 and supplemental material) exhibit clearly the best visual quality of the comparison methods. As seen in the 2nd column from the right, our main competitor METHOD E suffers from residual noise, particularly in the difficult soft tissue regions. This is because the low-dose projections of our real-world dataset were constructed by adding noise on top of full-dose projections (Appendix A), so that there is a common noise component between inputs and outputs that the 3D network learns to partially preserve. Our method removes the target projections

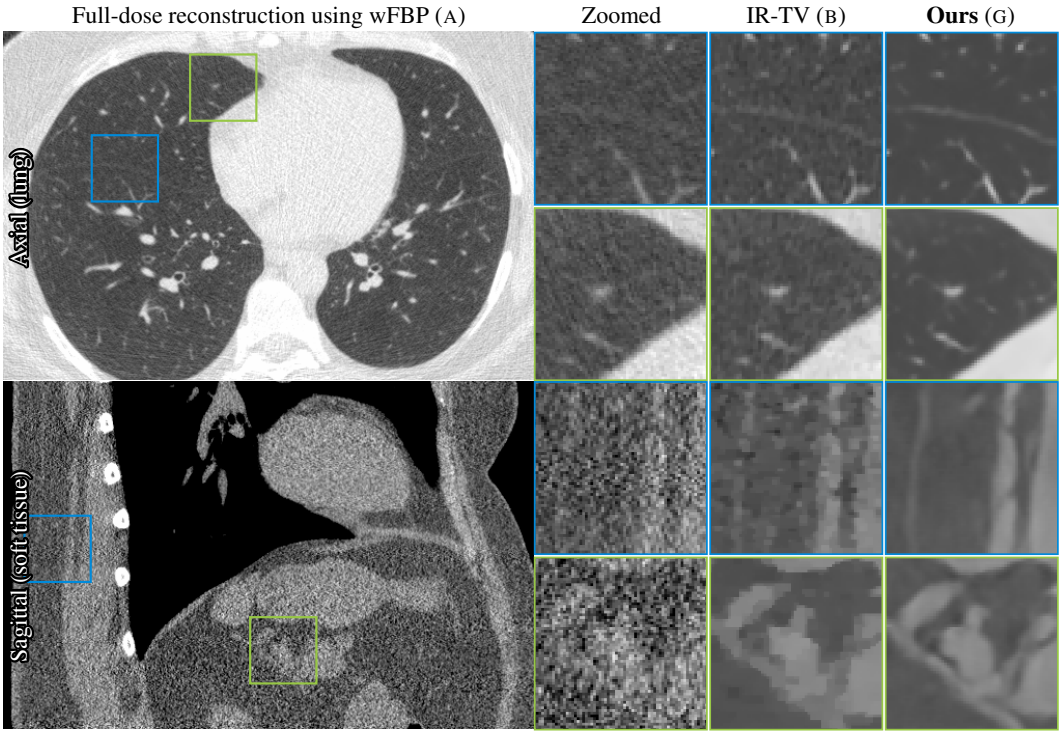


Figure 5: Full-dose reconstructions of real-world data. Soft tissue window is set to [−300, 300] HU, and lung window to [−1350, 150] HU. Our self-supervised loss enables us to train our pipeline with full-dose input data, which is not possible with traditional supervised methods. Compared to FBP and total variation regularized iterative reconstruction, our results contain less noise and do not suffer from IR-TV's blockiness. Full images and neighboring slices are available in the supplement.

from the set of input projections in each training step, avoiding this issue. See Appendix B for a targeted study on correlated noise and target exclusion.

Validation: volume-to-projection. Finally, we verify that the dramatic increase is not due to only the self-supervised loss but an emergent property of the combination of the pipeline and training approach. For this, we constructed the novel METHOD H variant that uses wFBP volumes as inputs and processes them using our 3D U-Net, but trains the 3D U-Net using our self-supervised loss in the projection domain (volume-to-projection). The result, 39.38 dB, is significantly worse than our full approach. This and the previous results indicate that using raw projection inputs and supervising in the projection domain are indeed greatly beneficial when done together, but not in isolation.

5.2 Full-dose inputs

Owing to the projection-domain loss and leave-out strategy, our method can be trained to operate on full-dose scans as well. Table 1 shows that our method achieves the best numerical results compared to wFBP and IR-TV. Other comparison methods are not applicable in this situation, as it is not possible to construct training targets from the same projections that are also used for the inputs without significantly decreasing data efficiency. As shown in Figure 5, our reconstructions are of clearly higher fidelity than those of the baselines.

6 Discussion and future work

We have shown that self-supervised training can be highly beneficial in helical CBCT reconstruction, and believe that the idea of combining projection simulation with end-to-end machine learning could be applied in a range of other tomography setups and other inverse imaging problems.

There are also several specific improvements that could be made in the CBCT case. Most importantly, our training-time model of the imaging setup, i.e., generation of simulated projections from recon-

structed volume, is fairly simplistic. For example, we do not utilize tube current information that determines the exposure for each X-ray and has an effect on the magnitude of noise. We also do not currently attempt to reproduce effects such as beam hardening (selective attenuation of low-energy photons), scattering, and metal artifacts. We believe that using uncorrected raw projection data and simulating these effects during loss computation could lead to further significant improvements, as these artifact-inducing effects are presumably easier to simulate than to remove directly.

Acknowledgments. We thank Timo Aila for discussions and comments on the manuscript; Samuli Siltanen, Alexander Meaney, Antti Korvenoja, Mika Kortesniemi, Juha Järveläinen and Jussi Hirvonen for discussions; David Luebke and Kimmo Kaski for general support; and Tero Kuosmanen, Janne Hellsten and Samuel Klenberg for compute infrastructure support. The project made use of computational resources provided by the Aalto Science-IT project and the Finnish IT Center for Science (CSC). Onni Kosomaa's NVIDIA Research internship notwithstanding, he was funded by the Finnish Center for Artificial Intelligence (FCAI), a part of the Academy of Finland Flagship programme.

References

[1] K. Stierstorfer, A. Rauscher, J. Boese, H. Bruder, S. Schaller, and T. Flohr, "Weighted FBP—A simple approximate 3D FBP algorithm for multislice spiral CT with good dose usage for arbitrary pitch," *Phys. Med. Biol.*, vol. 49, no. 11, pp. 2209–2218, May 2004.

[2] E. Y. Sidky and X. Pan, "Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization," *Phys. Med. Biol.*, vol. 53, no. 17, pp. 4777–807, 2008.

[3] K. Kim, G. El Fakhri, and Q. Li, "Low-dose CT reconstruction using spatially encoded nonlocal penalty," *Med. Phys.*, vol. 44, no. 10, pp. e376–e390, 2017.

[4] A. Katsevich, "Theoretically exact filtered backprojection-type inversion algorithm for spiral CT," *SIAM J. Appl. Math.*, vol. 62, no. 6, pp. 2012–2026, 2002.

[5] G. Wang, J. C. Ye, K. Mueller, and J. A. Fessler, "Image reconstruction is a new frontier of machine learning," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1289–1296, 2018.

[6] 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 Trans. Med. Imag.*, vol. 36, no. 12, pp. 2524–2535, 2017.

[7] A. A. Zamyatin, L. Yu, and D. Rozas, "3D residual convolutional neural network for low dose CT denoising," in *Proc. SPIE 12031, Med. Imag.*, 2022.

[8] T. Würfl, M. Hoffmann, V. Christlein, K. Breininger, Y. Huang, M. Unberath, and A. K. Maier, "Deep learning computed tomography: Learning projection-domain weights from image domain in limited angle problems," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1454–1463, 2018.

[9] J. He, Y. Wang, and J. Ma, "Radon inversion via deep learning," *IEEE Trans. Med. Imag.*, vol. 39, no. 6, pp. 2076–2087, 2020.

[10] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2Noise: Learning image restoration without clean data," in *Proc. ICML*, 2018.

[11] A. A. Hendriksen, D. M. Pelt, and K. J. Batenburg, "Noise2Inverse: Self-supervised deep convolutional denoising for tomography," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 1320–1335, 2020.

[12] D. Wu, K. Kim, and Q. Li, "Low-dose CT reconstruction with Noise2Noise network and testing-time fine-tuning," *Med. Phys.*, vol. 48, no. 12, pp. 7657–7672, 2021.

[13] J. Jing, W. Xia, M. Hou, H. Chen, Y. Liu, J. Zhou, and Y. Zhang, "Training low dose CT denoising network without high quality reference data," *Phys. Med. Biol.*, vol. 67, no. 8, p. 084002, Apr 2022.

[14] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M. K. Kalra, Y. Zhang, L. Sun, and G. Wang, "Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1348–1357, 2018.

[15] J. M. Wolterink, T. Leiner, M. A. Viergever, and I. Išgum, "Generative adversarial networks for noise reduction in low-dose CT," *IEEE Trans. Med. Imag.*, vol. 36, no. 12, pp. 2536–2545, 2017.

[16] J. P. Cohen, M. Luck, and S. Honari, "Distribution matching losses can hallucinate features in medical image translation," in *Proc. MICCAI*, 2018, pp. 529–536.

[17] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. ECCV*, 2020.

[18] M. Nimier-David, Z. Dong, W. Jakob, and A. Kaplanyan, "Material and lighting reconstruction for complex indoor scenes with texture-space differentiable rendering," in *Proc. EGSR (industry track)*, 2021.

[19] Z. Li, L. Wang, X. Huang, C. Pan, and J. Yang, "PhyIR: Physics-based inverse rendering for panoramic indoor images," in *Proc. CVPR*, 2022.

[20] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau, "Single-image SVBRDF capture with a rendering-aware deep network," *ACM Trans. Graph.*, vol. 37, no. 4, 2018.

[21] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, "DiffTaichi: Differentiable programming for physical simulation," in *Proc. ICLR*, 2020.

[22] J. Liang, M. Lin, and V. Koltun, "Differentiable cloth simulation for inverse problems," in *Proc. NeurIPS*, 2019.

[23] H. Chen, J. Gu, O. Gallo, M.-Y. Liu, A. Veeraraghavan, and J. Kautz, "Reblur2Deblur: Deblurring videos via self-supervised learning," in *Proc. ICCP*, 2018.

[24] E. Tseng, A. Mosleh, F. Mannan, K. St-Arnaud, A. Sharma, Y. Peng, A. Braun, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide, "Differentiable compound optics and processing pipeline optimization for end-to-end camera design," *ACM Trans. Graph.*, vol. 40, no. 2, 2021.

[25] C. E. Shannon, "Communication in the presence of noise," *Proc. Inst. Radio Eng.*, vol. 37, no. 1, pp. 10–21, 1949.

[26] L. Williams, "Pyramidal parametrics," *SIGGRAPH Comput. Graph.*, vol. 17, no. 3, pp. 1–11, Jul 1983.

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, 2015, pp. 234–241.

[28] X. Mao, C. Shen, and Y. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proc. NeurIPS*, 2016, pp. 2802–2810.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. ICCV*, 2015, pp. 1026–1034.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[31] G. L. Zeng and G. T. Gullberg, "Unmatched projector/backprojector pairs in an iterative reconstruction algorithm," *IEEE Trans. Med. Imag.*, vol. 19, no. 5, pp. 548–555, 2000.

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, 2019, pp. 8024–8035.

[33] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila, "Modular primitives for high-performance differentiable rendering," *ACM Trans. Graph.*, vol. 39, no. 6, 2020.

[34] T. R. Moen, B. Chen, D. R. Holmes III, X. Duan, Z. Yu, L. Yu, S. Leng, J. G. Fletcher, and C. H. McCollough, "Low-dose CT image and projection dataset," *Med. Phys.*, vol. 48, no. 2, pp. 902–911, 2021.

[35] W. P. Segars, M. Mahesh, T. J. Beck, E. C. Frey, and B. M. W. Tsui, "Realistic CT simulation using the 4D XCAT phantom," *Med. Phys.*, vol. 35, no. 8, pp. 3800–3808, 2008.

[36] J. Hoffman, S. Young, F. Noo, and M. McNitt-Gray, "Technical note: FreeCT_wFBP: A robust, efficient, open-source implementation of weighted filtered backprojection for helical, fan-beam CT," *Med. Phys.*, vol. 43, no. 3, pp. 1411–1420, 2016.

[37] J.-B. Thibault, K. D. Sauer, C. A. Bouman, and J. Hsieh, "A three-dimensional statistical approach to improved image quality for multislice helical CT," *Med. Phys.*, vol. 34, no. 11, pp. 4526–4544, 2007.

[38] H. Kunze, K. Stierstorfer, and W. Härer, "Pre-processing of projections for iterative reconstruction," in *Proc. Fully3D*, 2005.

[39] A. L. Maas, A. Y. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013.

[40] C. Lanczos, *Applied Analysis*. Prentice Hall, 1956.

Appendices

A Evaluation details

A.1 Datasets

The specifications of our two datasets are listed in Table 2.

Real-world data. For the real data experiments we use the Low Dose CT Image and Projection Dataset (LDCT) [34], built at Mayo Clinic, from which we use 47 chest scans captured on Siemens scanners. Each scan contains full-dose projections with various corrections (e.g., for beam hardening, scattering, nonuniformity) applied to them by the scanner manufacturer. In addition, each scan has a corresponding set of simulated low-dose (10% of full dose) projections created by adding noise on top of the full-dose projections. It is important to note that the low-dose and full-dose data are correlated, as they both contain the full-dose noise. Each scan also has a reference 3D reconstruction computed by the scanner manufacturer, but these references are noisy, so we cannot perform numerical comparisons against them.

Synthetic data. We generate a synthetic helical cone-beam dataset using the XCAT CT projection simulator [35]. We simulate full-dose and low-dose scans of each phantom. We chose the scanner parameters to match those of the real-world dataset with the exception that we do not use a flying focal spot. Because the projections in the real-world dataset have beam hardening correction applied, we simulate monochromatic radiation with an energy level of 80 keV to avoid beam hardening effects in the synthetic data as well. Following the real-world setup further, we simulate tube current modulation that attempts to keep the photon Poisson noise roughly consistent throughout the scan. To produce high-quality ground truth volumes, we export the voxel output from XCAT at 16× our target resolution and downscale it using a 16×16×16-voxel box filter, yielding a total of 4096 samples per output voxel.

Our loss function (Equation 1 of the main paper) operates on photon intensities that we obtain directly for the synthetic XCAT projections. The LDCT data contains only log-attenuation values, and we calculate the corresponding per-pixel photon intensities by inverting the log transform. The content in the XCAT dataset occupies a smaller area in the axial plane than the LDCT dataset; we use a reconstruction diameter of 452 mm for XCAT and 600 mm for LDCT.

A.2 Metrics

As reconstruction results are typically viewed in the log-attenuation space, we report root mean squared error (RMSE) and peak signal-to-noise ratio (PSNR) computed from log-attenuation results. We compute both metrics without clipping or quantization, assuming a display window of 2000 Hounsfield Units (HU). This window fully covers the variation in XCAT data. We compute the metrics over the full 3D volume instead of, e.g., averaging over individual 2D slices. In all cases, evaluation is performed using a separate validation set, i.e., a subset of scans that were not shown to the model during training.

A.3 Comparison methods

We compare our reconstructions with three previous methods: wFBP [1], RED-CNN [6], and total variation regularized iterative reconstruction (IR-TV) [2]. We use FreeCT [36] as the wFBP implementation. Our RED-CNN re-implementation follows Chen et al. [6]: It has five 5×5 encoder convolution layers followed by five 5×5 decoder convolution layers, each with 96 output channels, and the inputs of every other encoder layer are summed to the outputs of the corresponding decoder layers. The number of trainable parameters is 668k.

For IR-TV, we use a straightforward implementation that employs the same high-quality projection and backprojection operators as our reconstruction pipeline and performs the optimization using Adam [30]. In addition, we weight the projected rays according to their approximate noise level using a Poisson noise model [37]. The initial reconstruction is computed using wFBP, after which it is optimized iteratively according to a loss function that involves projection consistency and TV regularization terms. Each iteration uses progressively more simulated projections, and the

Table 2: Dataset details.

Dataset details	Real-world	Synthetic
Source	LDCT [34]	XCAT [35]
Training scans	41	13
Validation scans	6	4
Projections per scan	11,000–15,000	9,000–12,000
Projection resolution	736×64	736×64
Spiral pitch	0.9	0.9
Scan range	240–380 mm	210–300 mm
xy voxel grid size	1024×1024	576×576
z voxel grid size (training)	160	128
z voxel grid size (inference)	402–644	272–374
Reconstruction voxel spacing	0.586 mm	0.784 mm
Reconstruction cylinder diameter	600 mm	452 mm

final updates are based on all available projections. This increases efficiency by making the early iterations faster while still being sufficiently accurate. In total, 210 iterative volume updates are performed during reconstruction. As is customary, we low-pass filter the input projections to minimize ringing [38]. The low-pass kernel weights $[0.15, 1.0, 0.15]$ and the TV regularization term strength of $2 \cdot 10^6$ are chosen so that they maximize PSNR on synthetic data.

A.4 Reconstruction speed

In Table 1 of the main paper, we report FreeCT [36] runtime for wFBP. FreeCT is a GPU-accelerated reconstruction library, but our optimizations make our reconstruction pipeline roughly 3× faster. All times were measured on a single NVIDIA RTX 3090 GPU with 24 GB of memory. Timings for methods that use wFBP-reconstructed volumes as input (e.g., RED-CNN) include the initial wFBP execution. The IR-TV execution time should be considered a rough estimate, as we did not aim for an optimal speed vs. quality tradeoff; we simply increased the iteration count until the PSNR leveled off.

B Additional results

B.1 Qualitative results for synthetic data

Figure 6 shows example reconstructions of synthetic data, corresponding to the "Low-dose inputs" column in Table 1 of the main paper. The synthetic dataset is somewhat less challenging than the real-world dataset, especially in terms of fine details such as the pulmonary alveoli. Nevertheless, the differences in visual quality between the different methods are similar to the real-world results shown in Figure 4 of the main paper. Note that METHOD E performs better on synthetic data than real-world data, because the noise in the low-dose input projections is uncorrelated with the full-dose target projections.

Figure 7 further highlights the importance of performing denoising over the full 3D volume. 2D denoisers, such as RED-CNN, construct the 3D volume slice-by-slice, which results in strong artifacts along the non-axial cross-sections. Our 3D network, on the other hand, produces a consistent 3D volume by virtue of operating across all three axes simultaneously.

B.2 Using low-dose projections as training targets

Thus far, we have used full-dose projections as training targets for all learning-based methods to maximize their result quality, as we have paired projections with different dosages available in both of our datasets. In practice, however, it may be desirable to train exclusively using low-dose projections due to the challenges associated with data collection. Figure 8 shows a preliminary experiment, where we trained our method (METHOD G) using synthetic low-dose projections as both inputs and targets, without making any other changes to the training setup. Even though the results are of lower quality than those obtained using full-dose targets (METHOD G in Figure 6), they are still superior compared to the two baselines. Given that low-dose training targets provide less information about the true

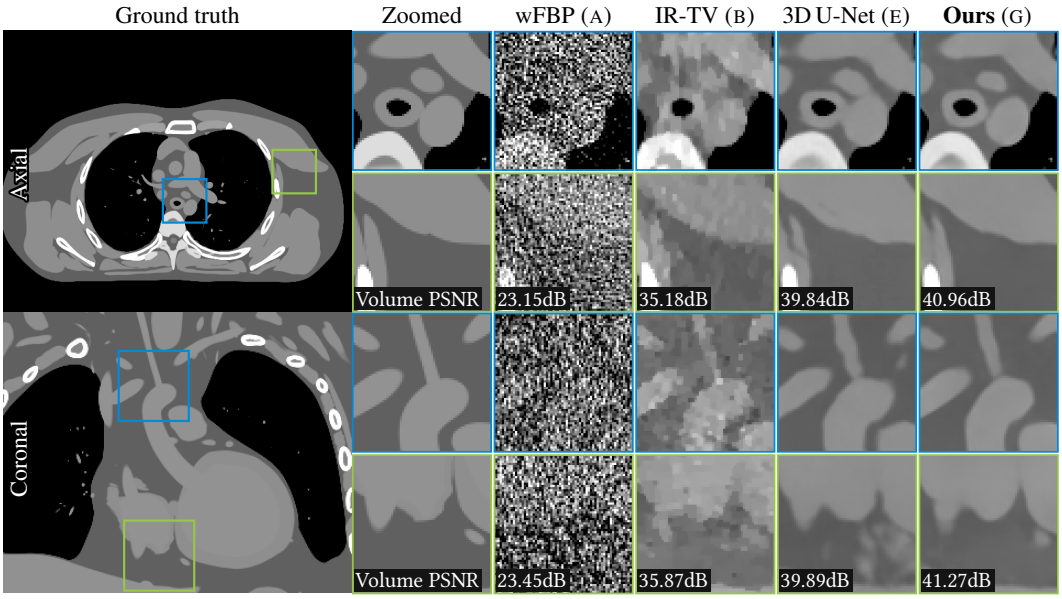


Figure 6: Low-dose reconstructions of synthetic data with different methods. Display window is set to [−400, 400] HU. PSNR values refer to the individual volumes shown, not the entire validation set. Full images and neighboring slices are available in the interactive HTML viewer.

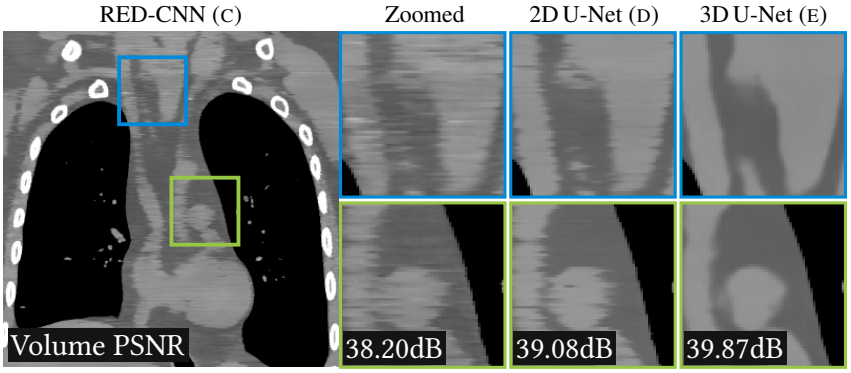


Figure 7: Coronal slices of low-dose reconstructions of synthetic data using supervised 2D and 3D denoisers. As the 2D methods process each axial slice independently, they suffer from inconsistencies in other planes. A 3D denoiser is consistent along all axes.

3D volume than full-dose projections, we suspect that the reconstruction quality could benefit from increasing the amount of training data in this configuration.

B.3 Importance of computing the loss in photon space

To gauge the effect of our photon-space loss (Equation 1 of the main paper), we trained variants of our method using L_2 loss in log-space instead. The log-transformation is nonlinear and therefore skews the mean of the targets, violating the zero-mean noise requirement of Noise2Noise training [10]. When training with full-dose targets, we did not observe significant differences in the numerical results, suggesting that the skew is small compared to the observed photon counts. However, when training with low-dose targets, the photon counts are lower, and the log-transformation skew is relatively larger. With low-dose targets we observed a drop of 3.32 dB compared to our photon-space loss function. Visual inspection confirmed that the results overestimated the attenuation, as the loss function was skewed towards higher log-space attenuation values.

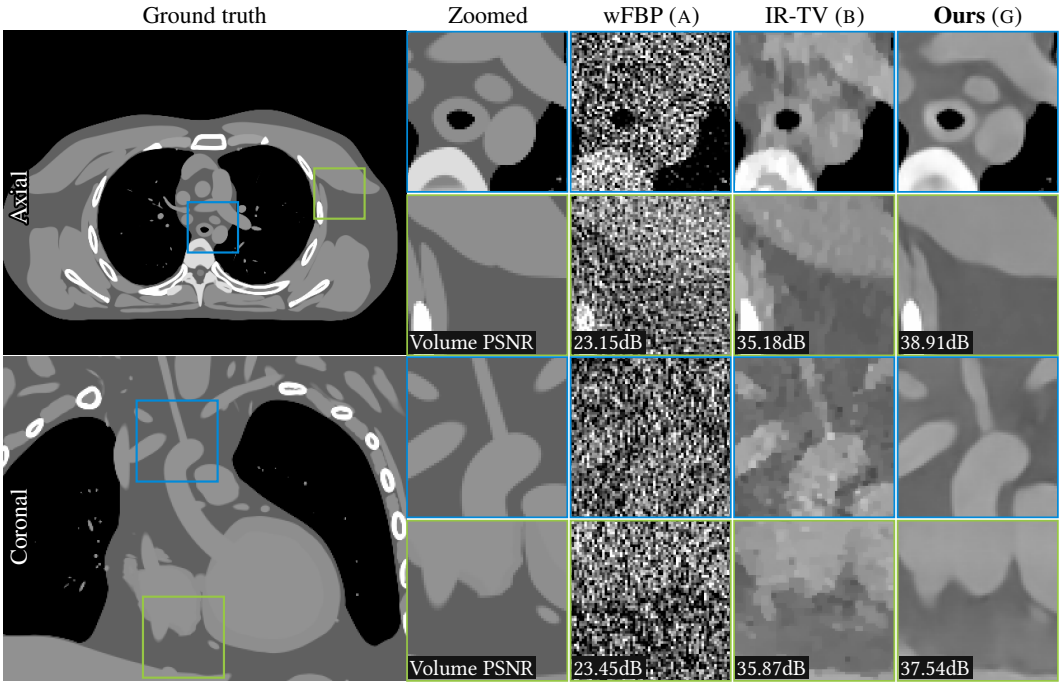
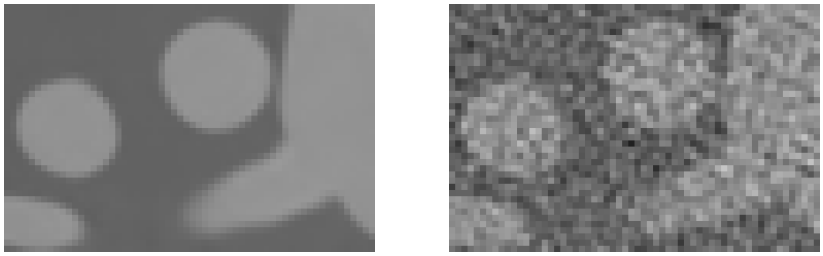


Figure 8: Low-dose reconstructions of synthetic data using our method trained with low-dose targets. Display window is set to [−400, 400] HU. The wFBP and IR-TV results in Figure 6 are duplicated here to facilitate visual comparison.



(a) Target projections correctly excluded from the input set

(b) Target projections incorrectly included in the input set

Figure 9: Example crops of our method trained with target projections excluded (a) and included (b) in the input set, using synthetic full-dose data. If the targets are not excluded, the network input becomes correlated with the targets and the networks learn to pass the noise through instead of removing it. Display window is set to [−400, 400] HU.

B.4 Correlated noise and target exclusion

To highlight the importance of not having correlated corruptions in the inputs and targets (Section 4.2 of the main paper), we performed an experiment where we trained our pipeline without excluding the target projections from the set of input projections. In this case, the input noise is correlated with the target projections and the networks learn to pass the noise on to the reconstructions. This leads to significant degradation in reconstruction quality, as illustrated in Figure 9. Note that the noise present in the real-world reconstructions for the supervised comparison methods (e.g., METHOD E in Figure 4 of the main paper) also stems from this effect: The full-dose noise is present in both the low-dose and full-dose data, so there is no incentive for the networks to remove it.

B.5 Experiments with noise-free inputs and targets

To validate the correctness of our self-supervised training setup, we performed additional experiments with noise-free synthetic data. If both input and target projections are noise-free, our method converges

to a virtually perfect result when using either the supervised or the self-supervised loss. This confirms that our networks are able to correct for artifacts resulting from the volumetric backprojection in a data-driven way, and that the self-supervised loss achieves results on par with the supervised loss when having access to synthetic ground truth reconstructions.

C Model and training details

C.1 Network architectures

The architecture and channel counts of our projection and volume networks are shown in Table 3. All convolution layers use Leaky ReLU [39] activation function with $\alpha = 0.1$, except for the final 1×1 convolution that has linear activation. Size-preserving padding is used at all convolutions. In total, there are 988k and 742k trainable parameters in the projection and volume networks, respectively. We have observed that halving the channel count of the 3D network hurts the numerical results significantly, whereas halving the channel count of the 2D network has a fairly small impact. Hence, it appears that increasing the channel count of the 3D network could improve the results further.

To avoid dependence on the physical units used in the data, we compute the mean and standard deviation of the projection and volume data, and scale and bias the network inputs and outputs so that the networks operate on zero mean and unit variance data on average. In other words, we define $y = f[(x - \mu_x)/\sigma_x] \cdot \sigma_y + \mu_y$, where f is the 2D or 3D network, x and y are the corresponding input and output, and μ and σ are the mean and standard deviation of x and y, computed over the training set.

C.2 Gradient checkpointing

Normally, when training neural networks, intermediate results are stored in memory in the forward pass so that they can be reused in the backward pass for backpropagation. For example, a convolution layer needs to remember its input in order to compute gradients for the convolution weights in the backward pass. *Gradient checkpointing* refers to a technique where the layer inputs are stored only at certain checkpoints. When an input is needed in the backward pass, the necessary forward operations are re-run from the closest checkpoint to regenerate them. As such, additional computation is expended but memory usage is lowered.

In our system, we use checkpointing in both the 2D projection network and the 3D volume network. The 2D projection network is run for all $\sim$7000 input projections when computing a single reconstruction. The intermediate data for each invocation consumes approximately 100MB of memory, and thus storing all the intermediate results would require 700GB of GPU memory in total. To avoid this situation, we store no intermediate results at all in the forward passes of the projection network. Instead, during backpropagation we regenerate the inputs of each layer by running the network once for each input projection in turn. This increases the cost of training the projection network by $\sim$33%: In addition to the usual forward and backward pass (typically twice as costly as a forward pass), an extra forward pass is required in between.

The volume network is run only once per reconstruction, but it operates on a very large 3D tensor representing the full-resolution volume. The intermediate results would again overrun GPU memory, which necessitates gradient checkpointing in the highest-resolution layers where the intermediate data is at its largest. Table 3 indicates the checkpointing blocks that we use for the volume network in the N_{3D} column. Layers in the checkpointing blocks need to be re-run once during backpropagation to regenerate the missing inputs, and the increase in computation workload remains below 33%.

C.3 Sparse backpropagation using a two-level training loop

A naive training loop would yield only a single weight update step per reconstruction. Considering the amount of computation required, this seems wasteful. We implement a two-level training loop that decouples the number of weight updates from the number of reconstructions, at the expense of slightly approximate reconstructions in the forward pass during training.

After choosing a random scan and z range from training dataset in the outer loop, we split the $\sim$7000 input projections and 96 target projections into eight buckets. For each bucket, we compute and store a partial reconstruction volume by running the input projections through the 2D projection network

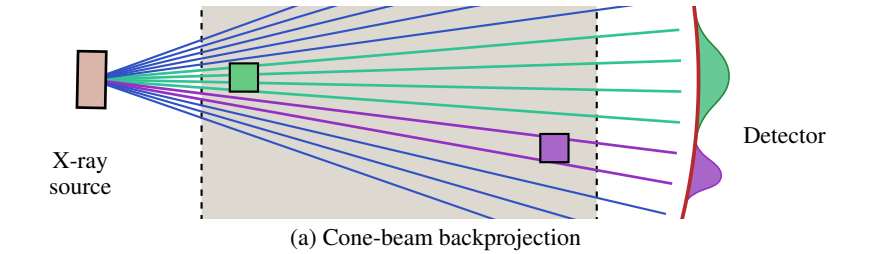
Table 3: Architecture of our neural networks. Columns N_{2D} and N_{3D} show the output channel count of each layer in our projection and volume networks, respectively. Joining arcs in the latter column indicate gradient checkpointing blocks in the volume network, whereas the projection network can be thought of as a single large checkpointing block.

NAME	N_{2D}	N_{3D}	FUNCTION
INPUT	1	1	
ENC_CONV0	48	24	Convolution $3 \times 3 (\times 3)$
ENC_CONV1	48	24	Convolution $3 \times 3 (\times 3)$
POOL1	48	24	Max-pool $2 \times 2 (\times 2)$
ENC_CONV2	48	24	Convolution $3 \times 3 (\times 3)$
POOL2	48	24	Max-pool $2 \times 2 (\times 2)$
ENC_CONV3	48	24	Convolution $3 \times 3 (\times 3)$
POOL3	48	24	Max-pool $2 \times 2 (\times 2)$
ENC_CONV4	48	24	Convolution $3 \times 3 (\times 3)$
POOL4	48	24	Max-pool $2 \times 2 (\times 2)$
ENC_CONV5	48	24	Convolution $3 \times 3 (\times 3)$
POOL5	48	24	Max-pool $2 \times 2 (\times 2)$
ENC_CONV6	48	24	Convolution $3 \times 3 (\times 3)$
UPSAMPLE5	48	24	Upsample $2 \times 2 (\times 2)$
CONCAT5	96	48	Concatenate output of POOL4
DEC_CONV5A	96	48	Convolution $3 \times 3 (\times 3)$
DEC_CONV5B	96	48	Convolution $3 \times 3 (\times 3)$
UPSAMPLE4	96	48	Upsample $2 \times 2 (\times 2)$
CONCAT4	144	72	Concatenate output of POOL3
DEC_CONV4A	96	48	Convolution $3 \times 3 (\times 3)$
DEC_CONV4B	96	48	Convolution $3 \times 3 (\times 3)$
UPSAMPLE3	96	48	Upsample $2 \times 2 (\times 2)$
CONCAT3	144	72	Concatenate output of POOL2
DEC_CONV3A	96	48	Convolution $3 \times 3 (\times 3)$
DEC_CONV3B	96	48	Convolution $3 \times 3 (\times 3)$
UPSAMPLE2	96	48	Upsample $2 \times 2 (\times 2)$
CONCAT2	144	72	Concatenate output of POOL1
DEC_CONV2A	96	48	Convolution $3 \times 3 (\times 3)$
DEC_CONV2B	96	48	Convolution $3 \times 3 (\times 3)$
UPSAMPLE1	96	48	Upsample $2 \times 2 (\times 2)$
CONCAT1	97	49	Concatenate INPUT
DEC_CONV1A	64	32	Convolution $3 \times 3 (\times 3)$
DEC_CONV1B	32	16	Convolution $3 \times 3 (\times 3)$
DEC_CONV1C	1	1	Convolution $1 \times 1 (\times 1)$, linear act.

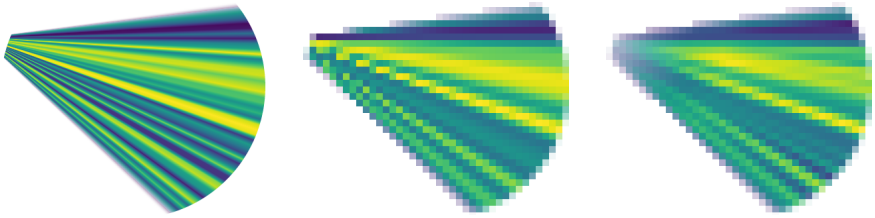
and backprojecting the results into a voxel grid. The 3D volume network is not run at this stage. The resulting partial volumes have the same size as the final reconstruction volume, but each of them only contains the contributions of 1/8$^{\text{th}}$ of the input projections.

Then, in an inner loop, we run a training step for each of the eight buckets. We pass the input projections in the chosen bucket through the 2D processing stage and backprojection, sum the result with the previously calculated partial volumes for the seven other buckets, pass the result through the 3D processing stage, calculate the loss, backpropagate the gradients, and update the weights. The next iteration of the inner loop uses the updated weights of the 2D projection network, ramp filter and 3D volume network. As such, we can perform 8 weight updates at the cost of running the 2D and backprojection stages twice and the 3D stages 8 times, with a slight increase in memory usage. Our scheme represents a form of sparse backpropagation, as only a subset of the input projections are allowed to contribute to the backpropagated gradients in each training iteration.

The partial volumes computed before the inner loop get progressively more out of sync with the most up-to-date projection network weights, which limits the number of useful iterations in the inner loop — we found eight iterations to be optimal in our case. In our experiments, the two-level training loop converged approximately $3\times$ faster than the naive training loop, measured in wall-clock time.



(a) Cone-beam backprojection



(b) Underlying signal (c) Constant prefilter (d) Our variable prefilter

Figure 10: A variable-size prefilter is necessary for high-quality cone-beam backprojection. (a) The green voxel near source intersects a thicker bundle of rays than the purple voxel near detector, and thus requires a wider prefilter in the projection domain to avoid aliasing. Sampling the underlying signal (b) with a constant-size prefilter in the projection domain (c) yields aliasing near source, blurring near detector, or both as in this example. A variable-size prefilter (d) extracts all the frequencies that the sampling grid can represent. 2D illustration, not to scale.

C.4 Training details

We distribute the training to multiple GPUs in a way that requires at least as many GPUs as the batch size. Each reconstruction in a minibatch is assigned to one or more GPUs. The input projections, 2D processing stages, and backprojection are distributed equally among the assigned GPUs in a data-parallel fashion. The 3D processing, on the othe hand, is implemented in a model-parallel fashion by dividing the input and output channels of each 3D convolution into equal-sized chunks and splitting the associated workload among the assigned GPUs.

The network weights were optimized using the Adam optimizer [30] with learning rate $\lambda = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-8}$, and batch size 4. An exponential moving average of the network weights with a decay factor of 0.99 was tracked during training and used for all evaluations. We trained the networks for a total of 40 000 gradient steps, corresponding to 5000 outer loop iterations in the two-level training loop.

C.5 Differentiable backprojection

Figure 10 illustrates the need for variable-size prefilter in the projection domain. As per standard Shannon–Nyquist sampling and reconstruction theory [25], the input signal (here, a 2D X-ray projection) needs to be prefiltered before sampling to remove spatial frequencies too high to be representable by the output signal (here, the voxel grid). With cone-beam geometry, this is non-trivial because the projection diverges and the sensor pixel pitch relative to voxel size changes with distance to the X-ray source. The required prefilter bandwidth is determined by this relative pixel pitch, and therefore it changes across the volume as well. If the prefilter size was kept constant and determined, say, according to the center of the volume, voxels close to the X-ray source would prefilter too little and exhibit aliasing, whereas voxels close to the detector would prefilter too much causing blur. In theory, every 3D voxel would need to sample the 2D projection using a dedicated prefilter kernel in order to account for the voxel's unique footprint on the 2D projection.

While supporting a per-voxel arbitrarily-sized 2D prefilter is infeasible, an excellent approximation can be achieved through a variation of *mipmapping* [26]. In practice, when preparing to backproject a 2D feature map, we compute a number of progressively smoother versions of it using Lanczos-3 [40] prefilters of increasing support (decreasing bandwidth). For each voxel, after the desired prefilter bandwidth has been determined, we reconstruct the backprojected sample via trilinear interpolation

from the two prefiltered projections whose filter sizes best match the desired bandwidth. This is a very close approximation to having a prefilter with arbitrary size and position on the 2D projection. In our current configuration, we use five prefilter bandwidths chosen to cover the range of filter bandwidths required by the backprojection. In contrast to standard mipmapping, we keep the resolution of the prefiltered feature maps unchanged.

C.6 Projection simulation and loss function

The simulated projections are computed by ray marching through the reconstructed voxel grid using trilinear interpolation. The samples are taken at stratified random positions along the ray to avoid structured sampling artifacts. To further prevent aliasing, we simulate the projections at $4\times$ higher resolution and downsample them by 4×4 using a Lanczos-3 filter.

As is common in iterative reconstruction techniques [38], we low-pass filter the target projections before computing the loss to adjust the amount of ringing in the results. We use separable filter kernels $[0.2, 1.0, 0.2]$ and $[0.05, 1.0, 0.05]$ for synthetic and real-world data, respectively.

We found that training convergence can be accelerated considerably by emphasizing high spatial frequencies in the pixelwise error images between simulated projections and target projections before computing the mean squared loss. In practice, we apply a classic (non-learned) ramp filter to the difference images, i.e., the weighted difference inside the norm in Equation 1. Reminiscent of the derivation of the ramp filter in wFBP, this focuses the loss evenly on all frequencies in the volume, whereas low frequencies tend to dominate otherwise. We use this crucial optimization in all of our training runs — without it, the convergence speed of high-frequency details was unbearably slow.

C.7 Augmentations

Three kinds of data augmentation are applied during training to increase the robustness of the model. First, we choose a random rotation around the z axis for the voxel grid, ensuring that the networks learn no preferential orientation of features in the xy plane. Second, we add a random sub-voxel offset for the center of the reconstruction volume to break the alignment of the voxel grid in relation to the scanner geometry. These augmentations are implemented by perturbing the geometry information in training data; they do not involve, e.g., resampling the projection or volume data itself.

Finally, we scale the log-attenuation values in all input and target projections by a random scalar in range $[0.75, 1.25]$ for each individual reconstruction during training. This is a valid transformation, because the backprojection and projection operations are linear in log-domain, and it prevents the networks from learning specific attenuation coefficients seen in the training data and exploiting that information when reconstructing previously unseen data.

In early tests with small datasets, we found that the augmentations improved results noticeably through better generalization power. However, our final training datasets appear to be large enough that the augmentations provide no significant benefit in terms of PSNR or visual quality. We still chose to use them in all our experiments, as we observed no detrimental effects in enabling them.

C.8 Custom CUDA kernels

We developed custom CUDA kernels for fast differentiable backprojection and for the ray-marching projection simulation.

In the backprojection operation, determining the 2D sample point on a projection for a single voxel in the volume requires trigonometry, analytic geometry, intersection computation, etc., resulting in a sequence of several dozen mathematical operations. Implementing each of these using PyTorch native operations is incredibly bandwidth-inefficient: For each primitive operation, the operands are fetched from GPU memory, a trivial computation (e.g., addition) is performed, and the results are stored back in memory. Due to the size of the tensors involved, none of the intermediate results stay in GPU caches, making the performance entirely throttled by memory bandwidth so that only a tiny fraction of the arithmetic power of the GPU is utilized. To overcome the issue, we collect the entire coordinate computation into a single CUDA kernel that also performs the band-limited texture lookup. PyTorch has a just-in-time compiler for fusing simple operations in certain cases, but a custom kernel was the only option for including the mipmapped Nvdiffrast [33] texture lookup as well.

For implementing the ray marching in the projection simulation, the natural approach is to step each ray in a loop and accumulate the log-attenuation in a local CUDA variable. This is not possible using native PyTorch operations. As such, a relatively simple custom kernel offers an easy way to improve performance of this step considerably.

As noted in Section 4.3 of the main paper, these operations can be used to compute the gradient of one another. Even though the correspondence is not exact, this does not matter in practice — experimentally it appears to be sufficient that both operations produce a high-quality, band-limited output.