Noise2Filter: fast, self-supervised learning and real-time reconstruction for 3D computed tomography

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 015012 https://doi.org/10.1088/2632-2153/abbd4d

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
30 June 2020

REVISED
15 September 2020

ACCEPTED FOR PUBLICATION
1 October 2020

PUBLISHED
1 December 2020

PAPER

Noise2Filter: fast, self-supervised learning and real-time reconstruction for 3D computed tomography

Marinus J Lagerwerf[1,3], **Allard A Hendriksen**[1,3], **Jan-Willem Buurlage**[1] and **K Joost Batenburg**[1,2]

[1] Computational imaging group, Centrum Wiskunde & Informatica, Amsterdam 1098 XG, The Netherlands
[2] Leiden Institute of Advanced Computer Science, Leiden Universiteit, 2333 CA Leiden, The Netherlands
[3] Equal contribution

E-mail: m.j.lagerwerf@cwi.nl

Keywords: computed tomography, reconstruction algorithm, real-time, machine learning, self-supervised learning, filtered backprojection, denoising

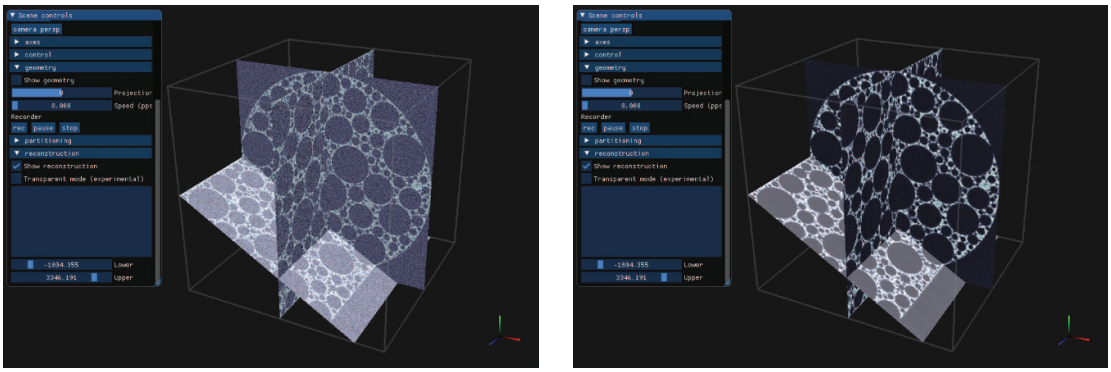
Abstract

At x-ray beamlines of synchrotron light sources, the achievable time-resolution for 3D tomographic imaging of the interior of an object has been reduced to a fraction of a second, enabling rapidly changing structures to be examined. The associated data acquisition rates require sizable computational resources for reconstruction. Therefore, full 3D reconstruction of the object is usually performed after the scan has completed. Quasi-3D reconstruction—where several interactive 2D slices are computed instead of a 3D volume—has been shown to be significantly more efficient, and can enable the real-time reconstruction and visualization of the interior. However, quasi-3D reconstruction relies on filtered backprojection type algorithms, which are typically sensitive to measurement noise. To overcome this issue, we propose Noise2Filter, a learned filter method that can be trained using only the measured data, and does not require any additional training data. This method combines quasi-3D reconstruction, learned filters, and self-supervised learning to derive a tomographic reconstruction method that can be trained in under a minute and evaluated in real-time. We show limited loss of accuracy compared to training with additional training data, and improved accuracy compared to standard filter-based methods.

1. Introduction

Computed tomography is a non-destructive imaging technique with applications in biology [1], energy research [2], materials science [3], and many other fields [4]. In a tomographic scan, a rotating object is positioned between a source emitting penetrating radiation and a detector that captures the projections of the object. Tomographic reconstruction algorithms compute a 3D image of the interior of the object from its projections. Besides extensive use in medical and laboratory settings, tomography is routinely used at synchrotron facilities, where advances in the last decade have enabled time-resolved imaging of the interior structure of a rapidly changing object [1–3]. So far, reconstruction algorithms are typically operated offline, enabling visualization of the object only after a scan has completed.

Recent advances in tomographic reconstruction enable real-time interrogation of the reconstructed volume during the scanning process using a quasi-3D reconstruction protocol [5, 6]. In this framework, arbitrarily oriented slices are selected for reconstruction and can be interactively rotated and translated, after which they are reconstructed and visualized virtually instantaneously. This creates the illusion of having access to the full reconstructed 3D volume, but at a fraction of the computational cost. The quasi-3D reconstruction protocol has been implemented in the RECAST3D software package. The information gained from this quasi-3D visualization can be used to directly steer the tomographic experiment, for instance, by adjusting an external parameter—such as temperature—in response to changes in the interior of the object. In addition, the object can be re-positioned, or other acquisition parameters can be adjusted to facilitate the best possible reconstruction [7].



(a) FBP reconstruction (b) Noise2Filter reconstruction

Figure 1. Real-time reconstructions using FBP and Noise2Filter of a high-noise acquisition using the RECAST3D software package. The highlighted slice is currently being moved.

Real-time 3D reconstruction is computationally demanding and data sizes are substantial—data acquisition rates of 7.7GB per second are not uncommon [6]. To attain real-time visualization, the quasi-3D reconstruction protocol is essentially limited to filtered backprojection type methods, since it exploits the locality of backprojection to obtain fast reconstructions. Filtered backprojection (FBP) methods are sensitive to measurement noise, leading to errors in the reconstructed slices [8]. Therefore, application of these methods in the quasi-3D reconstruction protocol is not well-suited to high-noise acquisitions [2, 9], as illustrated in figure 1(a).

In this paper, we combine a learning-based filtered reconstruction method with a self-supervised training strategy to obtain Noise2Filter, a denoising FBP-type reconstruction algorithm that can be applied in a quasi-3D reconstruction protocol. This algorithm is designed to be both fast to train and fast to evaluate. Moreover, no additional training data is required other than the measured projection data.

For dynamic scans, our method enables a possible use case where a static scan is performed—with the exact same acquisition rates as the dynamic scan—permitting the Noise2Filter method to be trained immediately. After training for tens of seconds, real-time visualization of the dynamic experiment can ensue, as illustrated in figure 1(b). In addition, we note that Noise2Filter can be used as a stand-alone reconstruction method.

The first component of our method is the Neural Network filtered backprojection (NN-FBP) method [10]. This method learns a set of filters, along with additional weights, and then forms the reconstructed image as a non-linear function of the individual FBP reconstructions, resulting in higher image quality than standard FBP. However, its application requires the availability of ground truth or high-quality reconstructed images.

This limitation can be overcome using the second component of our method, Noise2Inverse [11], which is a recent machine learning method designed to train denoising convolutional neural networks (CNNs) in inverse problems in imaging. To train a denoising CNN, the method splits the measured projection data to obtain multiple statistically independent reconstructed slices, which are presented to the network during training, without requiring additional high-quality data.

Our main contribution is that we show how to combine the NN-FBP method with the Noise2Inverse training strategy. In addition, we demonstrate that NN-FBP training can be substantially accelerated as compared to previous methods [10]. We evaluate our method on both simulated and experimental datasets, comparing to both conventional filter-based methods and supervised NN-FBP. Finally, we demonstrate that the method can be used in a quasi-3D reconstruction protocol, and exhibit its potential use for dynamic control of tomographic experiments.

The paper is structured as follows. In the next section, we introduce the tomographic reconstruction problem and the filtered backprojection algorithm. In addition, we introduce quasi-3D reconstruction, NN-FBP, and Noise2Inverse. These methods are combined in section 3, where we describe the Noise2Filter method. In sections 4 and 5, we describe experiments to analyze the reconstruction accuracy of Noise2Filter on real and simulated CT datasets. Moreover, we study the hyper-parameters of the proposed method. We discuss these results in section 6.

2. Preliminaries

2.1. Reconstruction problem

In parallel-beam tomography, an unknown object rotates with respect to a planar detector and a parallel source beam. Projections are acquired at a finite number N_a of rotation angles, yielding 2D images defined on an $N \times N$ pixel grid. The reconstruction problem can be modeled by a system of linear equations

$$W\mathbf{x} = \mathbf{y}, \tag{1}$$

where the vector $\mathbf{x} \in \mathbb{R}^n$ denotes the unknown object, $\mathbf{y} \in \mathbb{R}^m$ describes the measured projection data, and $W = (w_{ij})$ is an $m \times n$ matrix where w_{ij} denotes the contribution of object voxel j to detector pixel i. For the sake of simplicity we assume that the volume consists of $n = N \times N \times N$ voxels, and the projection dataset contains $m = N_a \times N \times N$ pixels.

2.2. Filtered backprojection methods

We consider the filtered backprojection (FBP) method for parallel beam tomography [12]. The FBP algorithm is a two step algorithm. First, the data $\mathbf{y} \in \mathbb{R}^m$ is convolved over the width of the detector with a one-dimensional filter $\mathbf{h} \in \mathbb{R}^{N_f}$. Next, the *backprojection* $W^T : \mathbb{R}^m \to \mathbb{R}^n$ is applied to compute a reconstruction $\mathbf{x}_{\text{FBP}} \in \mathbb{R}^n$. Expressing the FBP algorithm in terms of $\mathbf{h}$, $\mathbf{y}$ and W yields

$$\text{FBP}(\mathbf{y}, \mathbf{h}) = W^T(\mathbf{y} * \mathbf{h}) = \mathbf{x}_{\text{FBP}}. \tag{2}$$

Observation 1 (FBP is two-step) *The FBP algorithm consists of a* filtering step *and a* backprojection step, *and both can be computed separately. That is, the filtering can be performed in advance, and the backprojection can occur on demand. This technique will be used throughout the paper.*

We observe that the FBP algorithm can be described by a linear operator when fixing either $\mathbf{y}$ or $\mathbf{h}$. This will be exploited in the discussion of learned filter methods in section 2.4.

2.3. Quasi-3D reconstruction

A property shared by filtered-backprojection type algorithms is that they are *local*, in the sense that each voxel of the reconstructed volume can be computed directly from the filtered data by backprojecting onto only that voxel [5]. Therefore, if one is interested in a subset of the reconstructed volume, much of the computational cost of a full 3D reconstruction can be avoided. Specifically, if the reconstructed subset is a rectangular box or a slice, efficient backprojection algorithms such as those implemented in the ASTRA toolbox [13] can be used. This reduces the computational cost of the backprojection step by an order of N.

Observation 2 (Locality) *The backprojection operator is* local. *Computing the backprojection for a single voxel or a subset of voxels is therefore substantially faster than computing the backprojection for all voxels.*

This methodology has been implemented in the RECAST3D software package [5], which exposes a limited number of arbitrarily oriented 2D slices. These slices are interactive and can be manipulated by the technician of the tomographic experiment. This technique for real-time visualization has been successfully applied in practice to acquisitions in micro-CT systems [14], synchrotron tomography [6], and electron tomography [7].

2.4. NN-FBP reconstruction algorithm

The NN-FBP algorithm learns a set of suitable filters and a set of weights, and then forms a non-linear model that combines the individual FBP reconstructions. The algorithm may be considered as a multi-layer perceptron [15] that operates pointwise on a collection of suitable reconstructions. A schematic representation of the NN-FBP algorithm is given figure 2, a mathematical description is given below.

To obtain these reconstructions, we first make some general observations: a filter $\mathbf{h}$ can be seen as a vector in $\mathbb{R}^{N_f}$, and the FBP method is linear in the filter when fixing the measured projection data $\mathbf{y}$. Therefore, an FBP reconstruction can be expressed as a linear combination in the basis of the filter. Let $\mathbf{e}_1, \ldots, \mathbf{e}_{N_f}$ be any basis for the space of filters $\mathbb{R}^{N_f}$, such as the standard basis. Define the reconstruction of $\mathbf{y}$ filtered by a basis element $\mathbf{e}_i$ as

$$\mathbf{x}_{\mathbf{e}_i} := W^T(\mathbf{y} * \mathbf{e}_i). \tag{3}$$

Then we can write the FBP reconstruction as a linear combination of these reconstructions

$$\mathbf{x}_{\text{FBP}}(\mathbf{y}, \mathbf{h}) = \sum_{i=1}^{N_f} \mathbf{h}_i \mathbf{x}_{\mathbf{e}_i} = \sum_{i=1}^{N_f} W^T(\mathbf{y} * \mathbf{h}_i \mathbf{e}_i) = W^T(\mathbf{y} * \mathbf{h}), \tag{4}$$

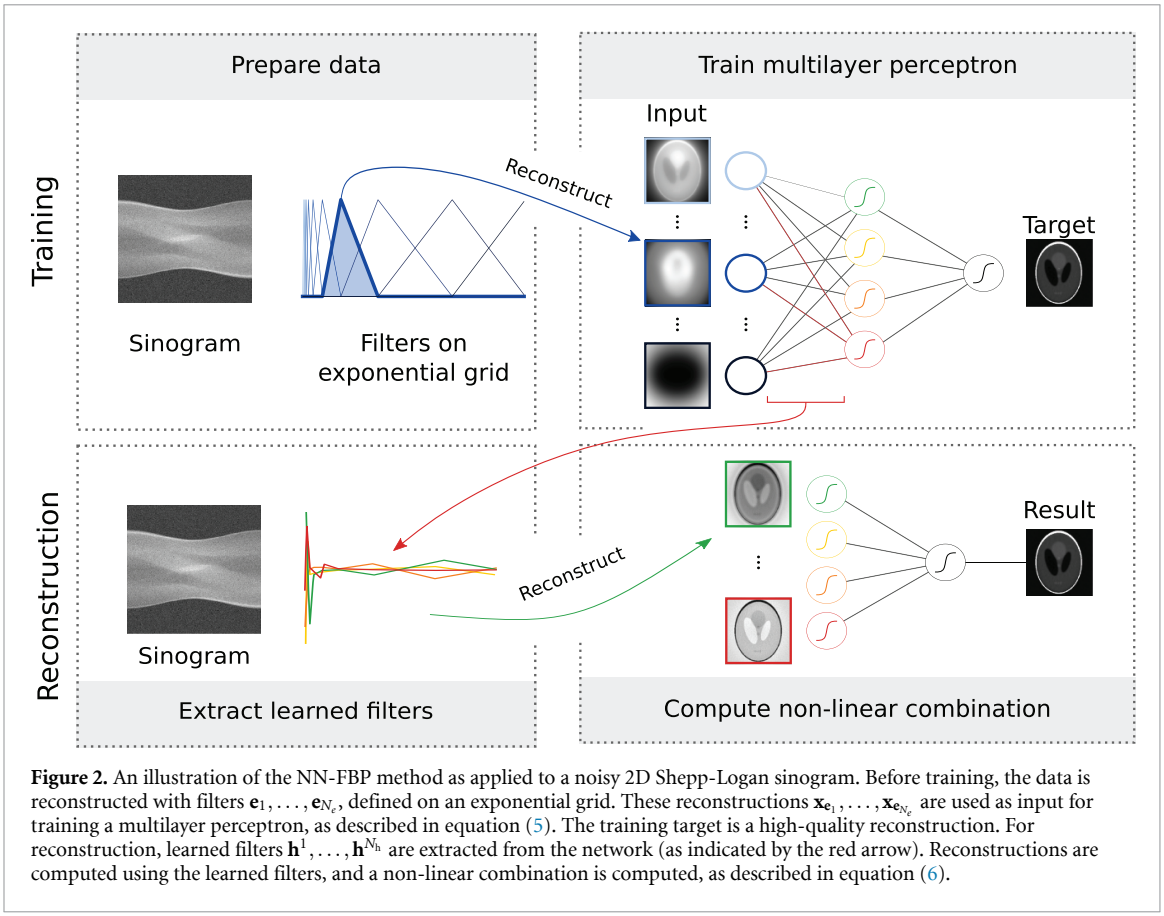


Figure 2. An illustration of the NN-FBP method as applied to a noisy 2D Shepp-Logan sinogram. Before training, the data is reconstructed with filters $\mathbf{e}_1, \ldots, \mathbf{e}_{N_e}$, defined on an exponential grid. These reconstructions $\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_e}}$ are used as input for training a multilayer perceptron, as described in equation (5). The training target is a high-quality reconstruction. For reconstruction, learned filters $\mathbf{h}^1, \ldots, \mathbf{h}^{N_h}$ are extracted from the network (as indicated by the red arrow). Reconstructions are computed using the learned filters, and a non-linear combination is computed, as described in equation (6).

where $\mathbf{h}_i$ denotes the coordinate of the ith basis element $\mathbf{e}_i$.

Given a set of N_h filters $\mathbf{h}^1, \ldots, \mathbf{h}^{N_h}$, we can define a multi-layer perceptron (MLP) with one hidden layer as a function of the reconstructions $\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_f}}$

$$\mathrm{MLP}_\theta(\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_f}}) = \sigma\left(\sum_{k=1}^{N_h} \mathbf{a}_k \sigma\left(\sum_{i=1}^{N_f} \mathbf{h}_i^k \mathbf{x}_{\mathbf{e}_i} - \mathbf{b}_k\right) - \mathbf{b}_0\right), \tag{5}$$

where σ is a non-linear activation function, such as the sigmoid. The multi-layer perceptron has free parameters $\theta = (\mathbf{a}, \mathbf{b}, \mathbf{h}^1, \ldots, \mathbf{h}^{N_h})$. Plugging equation (4) into equation (5), we obtain the NN-FBP reconstruction algorithm

$$\mathrm{NN-FBP}_\theta(\mathbf{y}) = \sigma\left(\sum_{k=1}^{N_h} \mathbf{a}_k \sigma\left(\mathrm{FBP}(\mathbf{y}, \mathbf{h}^k) - \mathbf{b}_k\right) - \mathbf{b}_0\right), \tag{6}$$

which is amenable to fast, parallel computation because it is a non-linear combination of FBP reconstructions.

Observation 3 (pointwise) *Note that the multi-layer perceptron operates* point-wise *on the voxels of the reconstructed volumes. Therefore, a* single *voxel can be computed without having to reconstruct other voxels. This observation connects to the observation of locality on Page 5, and will return several times in this paper.*

Supervised training [15] is used to determine the free parameters of the MLP defined in equation (5). The goal is to approximate a suitable *target* reconstruction $\mathbf{x}_{\mathrm{Target}}$ by minimizing

$$\left\| \mathrm{MLP}_\theta(\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_f}}) - \mathbf{x}_{\mathrm{Target}} \right\|_2^2, \tag{7}$$

i.e. the mean square error with respect to the target reconstruction.

The size of the training problem in equation (7) is related to the number of reconstructed volumes $\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_f}}$ and the size of these reconstructions, which suggests two techniques that may be used to accelerate training. First, to reduce the number of reconstructions, the filter is expressed on an exponentially binned grid, which grows logarithmically in the width of the filter. Since the filter width is proportional to

the number of pixels in each detector row, we have $N_e = O(\log N_t) = O(\log N)$. This technique yields suitable filter approximations, as observed in [10, 16]. Second, training may be accelerated by sampling *a subset of voxels* on which to minimize equation (7), rather than the full volume. Subsampling is possible because NN-FBP operates pointwise, as noted in Observation 3.

To summarize, we can split the NN-FBP algorithm into three parts, namely: (1) *data preparation*, where the input training data $\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_e}}$ is computed, (2) *network training*, where the weights $\theta^\star$ for the network are determined using a supervised learning approach, and (3) the *reconstruction algorithm*, which is summarized in Algorithm 1.

We use the same network architecture as proposed in [10]. The hyperparameters used in this paper are discussed in section 4.2.

Algorithm 1 NN-FBP reconstruction algorithm

1: Given projections $\mathbf{y}$ and a set of parameters $\theta^* := \left(\mathbf{a}, \mathbf{b}, \mathbf{h}^1, \ldots, \mathbf{h}^{N_h}\right)$.
2: Compute the FBP reconstruction using the learned filters:
3: **for** $k = \{1, 2, .., N_h\}$ **do**
4: $\mathbf{x}_{\mathbf{h}^k} = \mathrm{FBP}(\mathbf{y}, \mathbf{h}^k)$
5: **end for**
6: Compute a non-linear combination of these reconstructions:
$\mathrm{NN-FBP}_{\theta^\star}(\mathbf{y}) = \sigma\left(\sum_{k=1}^{N_h} \mathbf{a}_k \sigma(\mathbf{x}_{\mathbf{h}^k} - \mathbf{b}_k) - \mathbf{b}_0\right)$

2.5. Noise2Inverse training

Noise2inverse is a technique to train a convolutional neural network (CNN) to denoise reconstructed images in a self-supervised manner [11]. This means that no additional training data is required beyond the acquired noisy measurements. The key idea is change the training strategy by splitting the projection dataset into subsets, computing sub-reconstructions with these subsets and train a neural network mapping one sub-reconstruction to another.

First, the projection data is split into N_s sub-datasets such that projection images from successive angles are placed in different sub-datasets $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{N_\mathrm{s}}$. The network is trained to predict the reconstruction from one subset using the reconstruction of the other subsets. Training therefore aims to find the parameter θ^* that minimizes

$$\sum_{j=1}^{N_\mathrm{s}} \left\| \mathrm{CNN}_\theta(\mathrm{FBP}(\mathbf{y}_j)) - \mathrm{FBP}(\mathbf{y}_{l\neq j}) \right\|_2^2, \tag{8}$$

where $\mathrm{FBP}(\mathbf{y}_j)$ denotes the reconstruction from one subset of the data, and $\mathrm{FBP}(\mathbf{y}_{l\neq j})$ denotes the FBP reconstruction of the remaining subsets. We observe that the FBP reconstruction of a projection dataset is the mean of the FBP reconstruction of each projection image individually, which enables us to obtain

$$\mathrm{FBP}(\mathbf{y}_{l\neq j}) = \frac{1}{N_\mathrm{s} - 1} \sum_{l\neq j} \mathrm{FBP}(\mathbf{y}_l). \tag{9}$$

Now the *original training data* can be denoised by applying the trained network to each subreconstruction individually and averaging to obtain

$$\mathbf{x}_{\mathrm{N2I}} = \frac{1}{N_\mathrm{s}} \sum_{i=1}^{N_\mathrm{s}} \mathrm{CNN}_{\theta^\star}(\mathrm{FBP}(\mathbf{y}_i)). \tag{10}$$

In the previous discussion, we have assumed that the target images are reconstructed from more subsets than the input images. As in [11], we call this the *1:X* strategy. A reverse *X:1* training strategy is also possible. Here, the target is a single subreconstruction and the input is reconstructed from the remaining sub-datasets.

Note that convolutional neural networks take into account the surrounding structure of a voxel, typically a 2D slice, and thus do not operate pointwise. Therefore, these networks are an example where Observation 3 does not apply.

3. Noise2Filter method

Our proposed method combines the three ideas introduced in the previous section. The NN-FBP method is trained on a single projection dataset using the Noise2Inverse training strategy. This enables fast

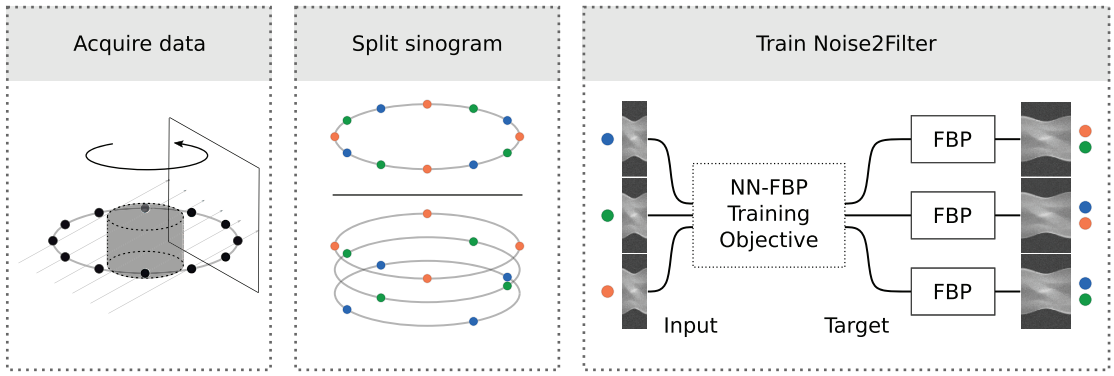


Figure 3. An illustration of the training of the Noise2Filter method. Data is acquired using a 3D parallel beam geometry. For each detector row, the sinogram is split in three sub-datasets such that acquisitions from successive projection angles belong to different sub-datasets. Each sub-dataset is used as input for NN-FBP training; the remaining sub-datasets are used in the target FBP reconstruction. This illustration depicts the *1:X* strategy. In the *X:1* strategy, the input is computed from the majority of the data, and the target from the minority, rather than vice versa.

reconstruction of arbitrarily oriented slices using the NN-FBP reconstruction algorithm in a quasi-3D reconstruction protocol.

Training The training procedure for the Noise2Filter method is similar to the NN-FBP procedure described in [10], with two notable exceptions. First, instead of minimizing the supervised training objective in equation (7), Noise2Filter minimizes a self-supervised training objective similar to equation (8). Second, training voxels are sampled from a subset of the reconstructed volume, rather than the full volume.

As in Noise2Inverse, the projection data $\mathbf{y}$ is split into N_s subdatasets with FBP reconstructions $\mathbf{x}_{\mathrm{FBP},j}, j = 1,\ldots,N_\mathrm{s}$. For each subdataset $\mathbf{y}_j$, we denote with $\mathbf{x}_{j,\mathbf{e}_i}$ a reconstruction filtered with basis element $\mathbf{e}_i$.

Training aims to minimize the difference between the MLP output of a subset of projection data and the FBP reconstruction of the remaining data. For the *1:X* training strategy, the MLP operates on a single subset of the data and the target is reconstructed from the remaining subsets. For the *X:1* training strategy, on the other hand, the target is reconstructed from a single subdataset, and the MLP operates on the remaining subsets. The self-supervised training objective thus becomes:

$$\sum_{j=1}^{N_\mathrm{s}} \left\| \mathrm{MLP}_\theta(\mathbf{x}_{j,\mathbf{e}_1},\ldots,\mathbf{x}_{j,\mathbf{e}_{N_e}}) - \mathbf{x}_{\mathrm{FBP},l\neq j} \right\|_2^2, \quad (\mathrm{X:1strategy}) \tag{11}$$

$$\sum_{j=1}^{N_\mathrm{s}} \left\| \mathrm{MLP}_\theta(\mathbf{x}_{l\neq j,\mathbf{e}_1},\ldots,\mathbf{x}_{l\neq j,\mathbf{e}_{N_e}}) - \mathbf{x}_{\mathrm{FBP},j} \right\|_2^2, \quad (1:\mathrm{Xstrategy}) \tag{12}$$

with

$$\mathbf{x}_{\mathrm{FBP},l\neq j} = \frac{1}{N_\mathrm{s}-1}\sum_{l\neq j}\mathbf{x}_{\mathrm{FBP},l}, \quad \mathbf{x}_{l\neq j,\mathbf{e}_i} = \frac{1}{N_\mathrm{s}-1}\sum_{l\neq j}\mathbf{x}_{l,\mathbf{e}_i}. \tag{13}$$

A schematic summary of the *1:X* training strategy is given in figure 3.

The second difference is related to the voxels that are considered for the training. Like NN-FBP, we minimize the training objective on a random sample of N_T voxels. We have $N_\mathrm{T} \ll N^3$, and increasing the sample size in response to increasing object size has been observed to yield diminishing returns. Unlike NN-FBP, training voxels are sampled only from the reconstructions of the axial, frontal, and longitudinal *ortho-slices*, rather than the full volume. This choice substantially reduces the computational effort of the data preparation step, as shown below.

Data preparation We discuss the *1:X* strategy; similar statements hold true for the *X:1* strategy.

The data preparation step is the most computationally expensive part of the method. In this step, an input reconstruction $\mathbf{x}_{l\neq j,\mathbf{e}_i}$ is computed for each subdataset $\mathbf{y}_j$ and each basis element $\mathbf{e}_i$. In addition, a target reconstruction $\mathbf{x}_{\mathrm{FBP},j}$ is computed for each subdataset, resulting in a total of $N_\mathrm{s}(N_e+1)$ reconstructions. These reconstructions are computed on the ortho-slices instead of the full volume. Due to locality—see Observation 2—the computational cost of the data preparation is therefore reduced by an order of N.

Note that the computational cost of the FBP algorithm scales linearly in the number of projection angles, therefore the computational cost of this step is equal to $3(N_e+1)$ FBP reconstructions of a 2D slice. Splitting the projection data thus has no adverse effect on the performance.

Reconstruction The reconstruction algorithm is almost identical to the NN-FBP reconstruction algorithm described in Algorithm 1. Whereas the aim of NN-FBP is to reconstruct the full volume, we aim only to reconstruct slices on demand. Therefore, reconstruction can be substantially accelerated.

We make use of Observation 1 that the FBP algorithm can be split in a filtering and backprojection step. First, the acquired projection data is filtered with the learned filters and cached. Then, a single slice can be reconstructed using Algorithm 1, which can occur in real-time due to the locality of the backprojection (Observation 2) and the pointwise nature of the multi-layer perceptron (Observation 3). Therefore, the reconstruction can be integrated in the quasi-3D reconstruction protocol, computing reconstructions of arbitrarily oriented slices in real time.

We note that the reconstruction step deviates slightly from the Noise2Inverse reconstruction described in equation (10). Rather than averaging separate reconstructions of each subset of the projection data, Noise2Filter computes a reconstruction using the learned filters directly from all data. In the context of self-supervised learning, this technique has been observed to yield improved results [17].

Noise2Filter summary

The Noise2Filter method consists of three steps. A summary of these steps, and specifically the computations performed, is given below:

1. **Data preparation** Compute the input and target training pairs from the measured projection data $\mathbf{y}$. Specifically, split the measured projection data in N_s equal sub-datasets and compute the following for the ortho-slices:

$$\mathrm{FBP}(\mathbf{y}_i, \mathbf{h}) \text{for } i = 1, \ldots, N_\mathrm{s} \tag{14}$$

$$\mathrm{FBP}(\mathbf{y}_i, \mathbf{e}_j) \text{for } i = 1, \ldots, N_\mathrm{s},\ j = 1, \ldots, N_e. \tag{15}$$

   The computational effort of this step is equal to $3(N_e+1)$ FBP reconstructions of a 2D slice.
2. **Training** Obtain a random sample of N_T voxels on the ortho-slices for inclusion in the training set. Compute the optimal parameters $\theta^\star$ that minimizes the training objective with respect to the sampled voxels. Note that the training time depends on the size of the training set, which may be fixed independent of the object size.
3. **Reconstruction** Using the computed parameters $\theta^\star$, compute an NN-FBP reconstruction for the desired 2D slices. Recall from equation (6) that the computational cost of an NN-FBP reconstruction is equivalent to N_h FBP reconstructions.

The network architecture used for the Noise2Filter method is the same as the architecture used for the NN-FBP method and the considered hyperparameters are discussed in section 4.2.

4. Experimental setup

In this section we discuss the setup of the experiments. Specifically, we describe the data used in the experiments, the implementation of NN-FBP and Noise2Filter, and the measures used to quantify these comparisons.

4.1. Simulated data

A phantom was generated by removing 100,000 randomly-placed non-overlapping balls from a foam cylinder. The `foam_ct_phantom` package [9] was used to generate analytic projection images with $2\times$ supersampling, were each pixel's value is averaged over four equally-spaced rays through the pixel. The result contains 1024 equally-spaced projection images with 512×768 pixels.

In each experiment, the simulated projection images were corrupted with Poisson noise of various levels of intensity, by altering the incident photon count I_0 per pixel. Specifically, we compute the mean measured photon I_mean count for an incident photon count I_0 from the analytic projection images $\mathbf{y}_\mathrm{analytic}$:

$$I_\mathrm{mean} = I_0 e^{-y_\mathrm{analytic}}. \tag{16}$$

Given the mean measured photon count, we draw from a Poisson distribution the measured photon count I with respect to I_0 and compute the corresponding noisy projection data $\mathbf{y}$:

$$I \sim \text{Pois}(I_{\text{mean}}) \qquad \mathbf{y} = -\log\left(\frac{I}{I_0}\right). \tag{17}$$

The average absorption of the sample was 10%. Reconstructions without Poisson noise and with Poisson noise ($I_0 = 1000$) are shown in figure 5.

4.2. NN-FBP and Noise2Filter

Noise2Filter and NN-FBP benefit from a shared implementation. Therefore, most almost all implementation details are the same. As in the original NN-FBP implementation [10], the number of learned filters is set to $N_h = 4$, the non-linear activation function is the sigmoid, the exponential binning parameter is set to 2, but the filters are piece-wise linear—rather than piece-wise constant—as proposed in [16]. Moreover, changes have been made to the shared implementation in order to accelerate data preparation, training, and reconstruction.

In the data preparation step, reconstructions are computed of the ortho-slices rather than the full volume. These reconstructions are performed using the RECAST3D software package [5].

Some changes have been made to the training procedure. As in the original implementation, the training objective is minimized using the Levenberg-Marquadt algorithm (LMA), which requires that the data samples are split into a training set and a validation set. Compared to the original implementation, however, the number of training samples is reduced from 10^6 to $5 \cdot 10^4$, and training is stopped after the validation set error has not improved for 10 epochs (originally 100 epochs were used). The effect of this reduction is discussed in section 5.2. In addition, the original CPU implementation of the training process is accelerated by performing computations on the graphics processing unit (GPU) using PyTorch [18].

Final reconstructions are computed using the RECAST3D software package [5].

NN-FBP The free parameters for the NN-FBP method are trained and tested on *separate* tomographic datasets. The training dataset consists of paired noisy and noiseless reconstructions. Supervised training minimizes the training objective in equation (7).

Noise2Filter The Noise2Filter parameters are optimized using self-supervised training on the noisy test dataset, rather than on a separate training dataset. No noiseless reconstructions are necessary for training. Depending on the training strategy (*X:1* or *1:X*), training minimizes either equation (11) or (12).

4.3. Quantitative measures

Reconstruction accuracy is quantified using the *Peak Signal-to-Noise Ratio* (PSNR) and the *Structural Similarity* (SSIM) index [19] metrics. Both metrics were computed with respect to the noiseless reconstructed images and using a data range that was determined by the minimum and maximum intensity of the noiseless reconstructed images. If not otherwise mentioned, the reported metrics are the average of the metric as computed on the three ortho-slices.

5. Experiments and results

We performed several experiments to evaluate the Noise2Filter method. We provide a short summary below.

Reconstruction accuracy We compare Noise2Filter to supervised NN-FBP training and several standard FBP improvement strategies in terms of reconstruction accuracy.

Hyperparameter analysis Implementation choices in the design of the Noise2Filter method are analyzed, including the number of training samples, training strategy (*X:1* or *1:X*), and number of splits.

Timing An analysis of data preparation, training, and reconstruction speed is given.

Experimental data The method is applied to experimental data, including a showcase that illustrates the potential for use in dynamic control.

5.1. Reconstruction accuracy comparison

In this section, we assess the reconstruction accuracy of the Noise2Filter method. We compare to other filter-based reconstruction techniques in terms of reconstruction accuracy. Specifically, we compare to a baseline FBP reconstruction (with a Ram-Lak filter) and FBP with standard noise reduction techniques—Gaussian filtering (FBP_G) and frequency scaling (FBP_{sc}). These two methods are discussed in more detail in appendix A. In addition, we compare to the NN-FBP, which is trained on a separate training dataset with ground truth images.

The comparison is performed on the simulated foam dataset with varying levels of Poisson noise. The incident photon count I_0 was varied between 1000 and 32, 000 in powers of two.

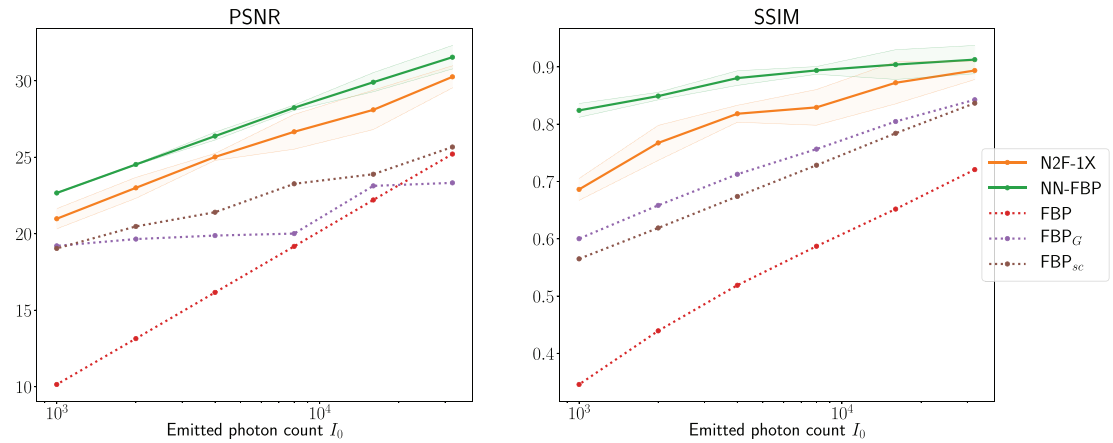


Figure 4. Reconstruction accuracy comparison of Noise2Filter (N2F-1X), NN-FBP, and FBP with Gaussian filtering, frequency scaling, and default filter. For varying noise levels, the average (line) and standard deviation (shaded region) over 20 trials of the PSNR and SSIM are reported.

For each of the methods, parameter selection was performed as follows. For Noise2Filter, training was performed on the noisy test set. For NN-FBP, training was performed on a separate training dataset. For both methods, training was repeated 20 times to obtain statistics for the PSNR and SSIM\@. For Gaussian filtering and frequency scaling, the parameters maximizing the SSIM on the test set were determined using a linear grid search.

The Noise2Filter method with the *1:X* training strategy and 3 splits is used. We find that this yields consistent results at various noise levels.

The quantitative measures for the ortho-slices are shown in figure 4. For all noise levels, the Noise2Filter metrics are higher than FBP with frequency scaling or Gaussian filtering. The NN-FBP method attains the best metrics, although the difference with Noise2Filter decreases as the noise level decreases. The difference in reconstruction accuracy is illustrated in figure 5, where the ground truth phantom, reconstructions, and residuals for all considered methods are shown for the incident photon count $I_0 = 1000$. Notice that NN-FBP and Noise2Filter remove the noise in the voids, unlike the FBP methods.

5.2. Hyper parameter analysis

We consider three hyper parameters for the N2F method: the number of samples considered for training, the training strategy *X:1* or *1:X* and the number of splits N_s for the measured projection data.

First, we analyzed the reconstruction accuracy as a function of N_T, the number of training samples used in the training process. Here, the number of validation samples is fixed to 10% of the number of training samples. Noise was applied to the projection dataset equivalent to $I_0 = 1000$. The results for this experiment are shown in figure 6. We observe that increasing the number of voxels yields virtually no increase in PSNR or SSIM beyond $N_T = 5 \cdot 10^4$ voxels.

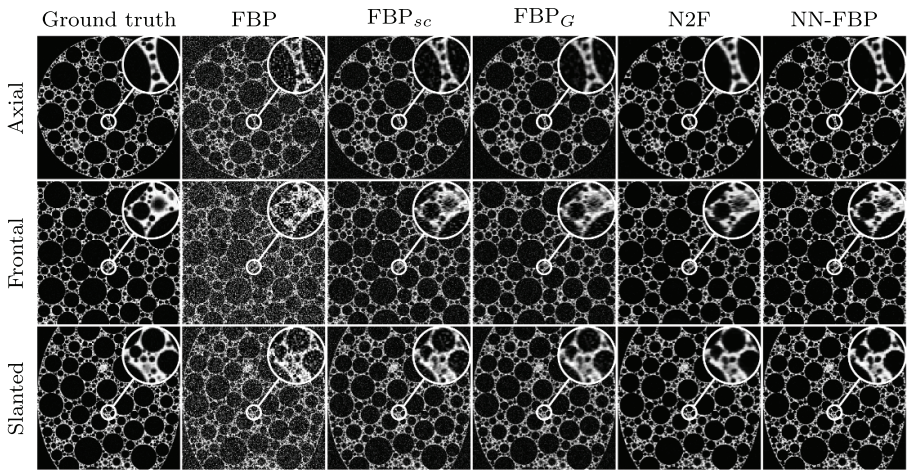
Second, we compare the training strategies and the number of splits on the simulated foam dataset for two noise levels, $I_0 = 1000$ and $I_0 = 8000$. For various values of the number of splits, 20 networks were trained and used to reconstruct the projection data. The average and standard deviation of the PSNR and SSIM are shown in figure 7. For both noise levels we observe that the *1:X* strategy with 3 splits obtains the best SSIM and close to the best PSNR.

5.3. Timing comparison

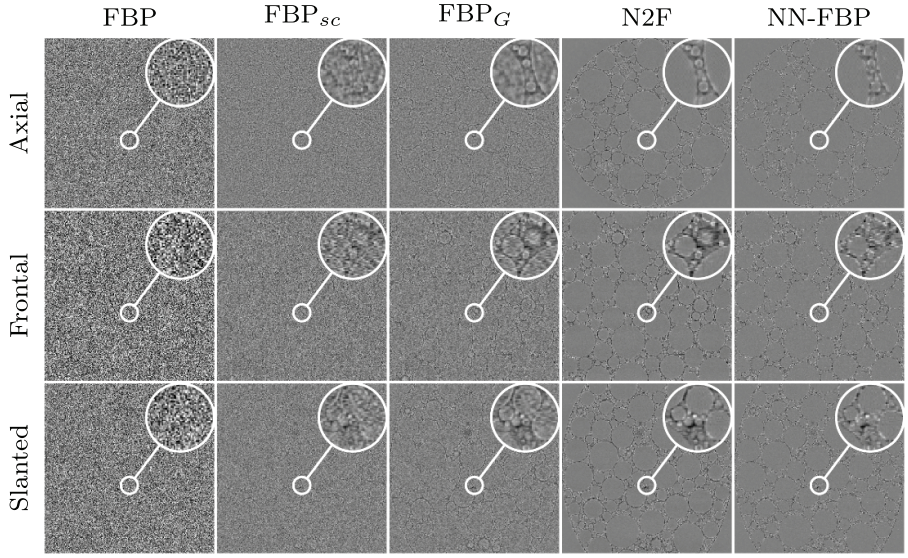
We give timings for the data preparation, training, and reconstruction step of the Noise2Filter method. The computations were performed on a server with 375 GB of RAM and made use of a single Nvidia GeForce RTX 2080 Ti GPU (Nvidia, Santa Clara, CA, USA).

We computed the mean and standard deviation of the training time and number of epochs over 50 trials, resulting in a training time of 5.45 ± 4.21 s and a number of epochs of 58.21 ± 34.73.

In table 1 we report the reconstruction times of one 2D slice using the RECAST3D framework for standard FBP and the Noise2Filter method. We see that Noise2Filter is roughly 4 times slower than standard FBP, which is expected considering that we use $N_h = 4$ learned filters.



(a) Reconstructions



(b) Residuals

Figure 5. Reconstructions and residuals of the FBP algorithm, FBP with frequency scaling (FBP$_{sc}$, $sc = 0.4$), FBP with Gaussian filtering (FBP$_G$, $\sigma = 1.5$), Noise2Filter (N2F), and NN-FBP on a simulated foam phantom with photon count $I_0 = 1000$. Results are shown on an axial, frontal, and 45° slanted slice. The insets are zoomed by a factor of four.

Table 1. Benchmark results for the data preparation (DP) and reconstruction steps. FBP and Noise2Filter (N2F) reconstructions are performed on a single slice from filtered projection data. Due to memory constraints, some reconstructions were not performed, as indicated by a —.

Data size				Duration (seconds)		
# voxels	# pixels	# angles	N_e	DP	FBP	N2F
128^3	128×192	256	10	0.34	0.003	0.009
256^3	256×384	512	11	1.34	0.006	0.024
512^3	512×768	1024	12	6.08	0.030	0.114
1024^3	1024×1536	2048	13	44.00	—	—

5.4. TomoBank dynamic dataset

We consider two experiments with an experimental dynamic tomographic dataset, consisting of 60 scans at consecutive time steps. First, we train Noise2Filter on the data from the first time step and use the trained reconstruction method to compute reconstructions for later time steps. This experiment aims to reveal the ability of Noise2Filter to generalize over dynamics in time. Second, we consider determining the correct center of rotation using Noise2Filter.

The experimental data is taken from the public TomoBank repository [4] and was acquired at the TOMCAT beamline at the Swiss Light Source (Paul Scherrer Institut, Switzerland). In this experiment,

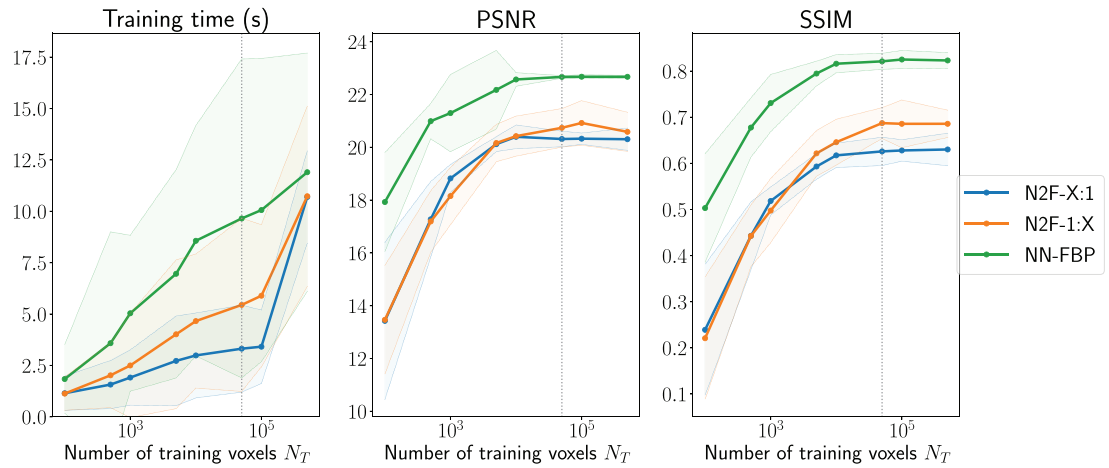


Figure 6. Training time and reconstruction accuracy for varying amounts of training voxels N_T. The mean (line) and standard deviation (shaded region) over 50 trials are reported. For both NN-FBP and Noise2Filter, increasing N_T yields diminishing returns in terms of PSNR and SSIM beyond $N_\mathrm{T} = 5 \cdot 10^4$, as indicated by the dashed line.

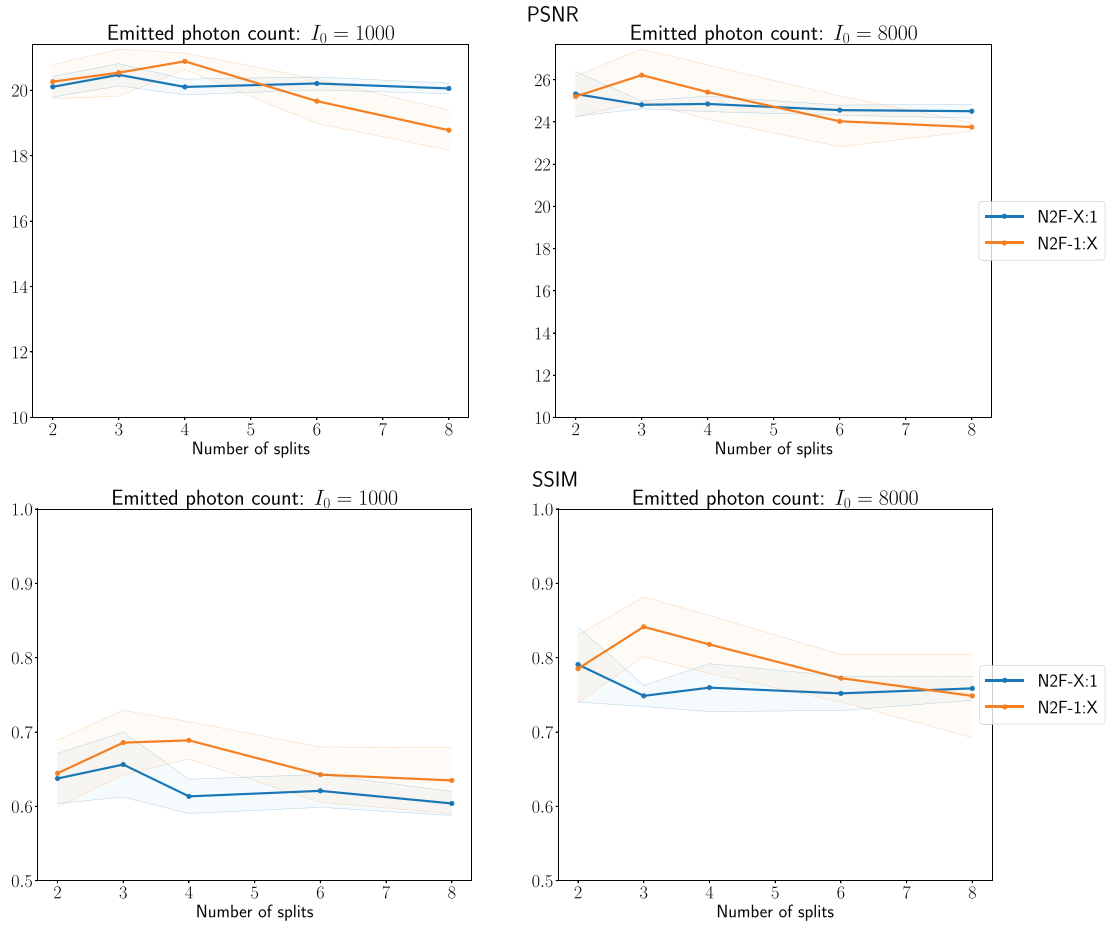


Figure 7. A comparison of Noise2Filter reconstruction accuracy for varying number of splits N_s and training strategies X:1 and 1:X. Mean (line) and standard deviation (shaded region) over 20 trials of the PSNR and SSIM are plotted for noise levels $I_0 = 1000$, and $I_0 = 8000$.

sub-second x-ray tomographic microscopy was used to investigate liquid water dynamics in a fuel cell during operation. The experiment took less than 6 seconds, during which 60 scans were acquired. A scan consists of 301 projections taken by a detector with 1100 × 1440 detector pixels. Without loss of generality we have set the pixel size to 1, which means the linear attenuation coefficient – *i.e.* the intensity of the

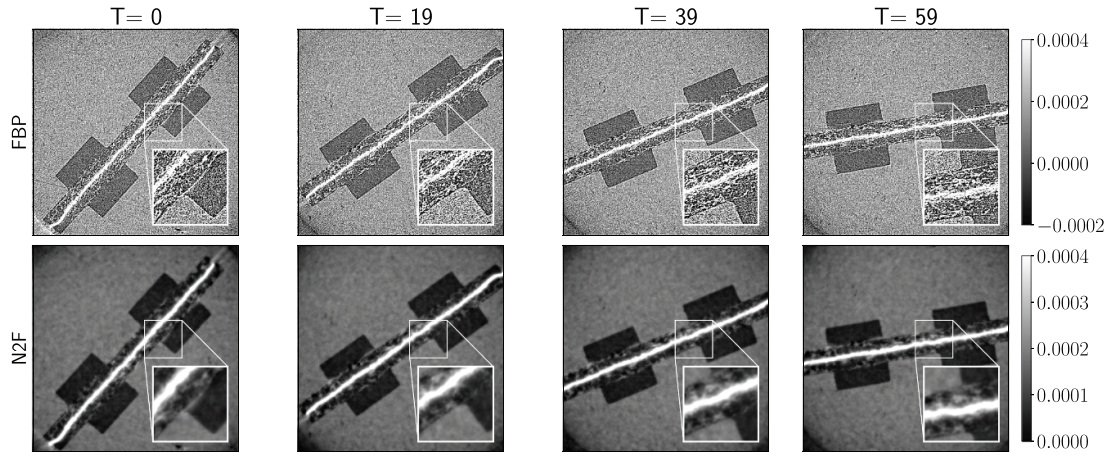


Figure 8. Reconstruction of the fuel cell at various time steps using FBP and Noise2Filter (N2F). The Noise2Filter method was trained on the first time step and also used to reconstruct later time steps. The insets are zoomed by a factor two.

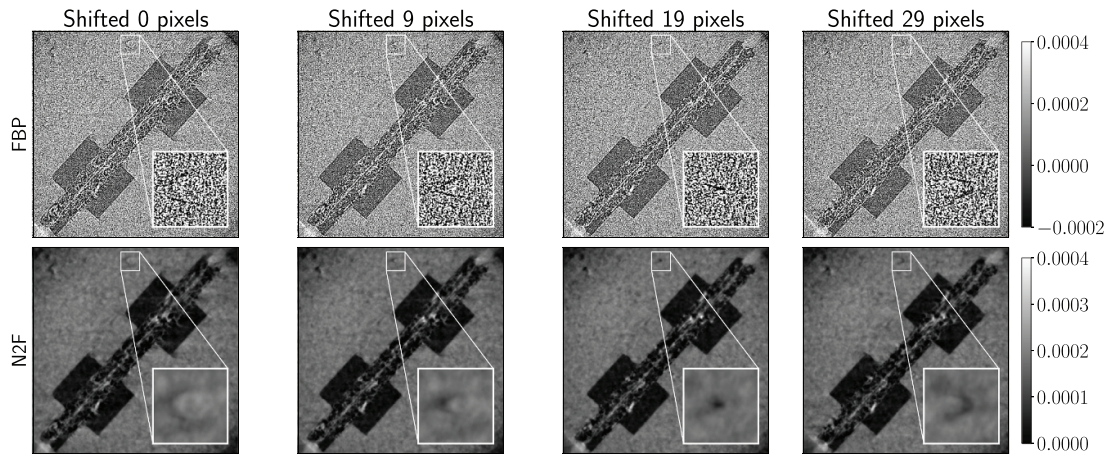


Figure 9. Reconstructions of a fuel cell at various centers of rotation using FBP and Noise2Filter (N2F). In the inset, a center of rotation artifact is highlighted, which disappears at a shift of 19 pixels. The distance between the detector pixels, or pixel pitch, for this dataset is 2.75 μm. The insets are zoomed by a factor four.

reconstructions—is expressed in attenuation per pixel. Note that there is no reference reconstruction available for these experiments. Therefore, the analysis of these experiments is purely qualitative.

First, we train a Noise2Filter network at the first time step $T = 0$ and use this network to evaluate all further time steps. Figure 8 shows the results for this strategy for $T = 0, 19, 39, 59$ and the FBP reconstructions at these time steps. There is no visible deterioration of the reconstruction accuracy over time, indicating that the trained network generalizes over the whole experiment.

Second, we consider determining the correct center of rotation. In the presence of noise, determining the correct center of rotation for a dataset can be difficult and is often performed after acquiring the measured projection data. Using the tools developed in [7], the center of rotation can be adapted interactively in real-time. In figure 9 we show Noise2Filter and FBP reconstructions with shifted centers of rotation at the first time step. We note that no retraining was performed for Noise2Filter: the network parameters were determined once using a shift of 0 pixels. In the FBP reconstructions, the center of rotation artifacts (half moons) are difficult to discern. In the Noise2Filter reconstruction, however, these artifacts are both clearly visible, and visibly disappear at a shift of 19 pixels, which coincides with the reported center of rotation in [4].

6. Conclusion and outlook

We have introduced Noise2Filter, a machine learning method for denoising filter-based reconstruction that does not require any additional training data beyond the acquired measurements. We show that this self-supervised method improves reconstruction accuracy compared to standard filter-based methods, and

has limited loss of accuracy compared to its supervised counterpart (NN-FBP). The method exhibits sub-minute training times and reconstruction times in the order of hundred milliseconds, which demonstrates the potential for use in quasi-3D reconstruction for real-time visualization of tomographic experiments. In addition, we demonstrate that visual calibration of the center of rotation is possible, which illustrates the potential of our method for use in the dynamic control of tomographic experiments where noise is a challenge.

This method enables operators of dynamic experiments to directly adjust for external parameters—such as temperature—in response to changes in the measured object, even with high acquisition noise. Moreover, it can be used in high-throughput real-time quality control applications, where a fast scanning protocol leads to data with high acquisition noise.

Acknowledgments

The authors acknowledge financial support from the Dutch Research Council (NWO), project number 639.073.506. The authors have made use of the following additional software packages to compute and visualize the experiments: Matplotlib and scikit-image [20, 21].

Data availability statement

No new data were created or analysed in this study. https://tomobank.readthedocs.io/en/latest/source/data/docs.data.dynamic.html#fuel-cell-data.

Appendix A. Standard FBP improvement strategies

In addition to standard FBP and NN-FBP, the Noise2Filter method is compared to two commonly used strategies to improve the reconstruction accuracy of the FBP algorithm for noisy data [22].

A.1. Gaussian filtering

In this strategy the standard filter $\mathbf{h}$ in the FBP algorithm is convolved with a Gaussian filter $G_\sigma \in \mathbb{R}^{N_f}$ to smooth the noise in the reconstructions, with σ the standard deviation of the Gaussian. The elements j of the filter G_σ are defined as follows:

$$(G_\sigma)_j = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(j-N_f/2)^2}{2\sigma^2}}, \tag{A1}$$

resulting in the smoothed reconstruction $\mathrm{FBP}_G(\mathbf{y}, \mathbf{h}, \sigma) = W^T(\mathbf{y} * (\mathbf{h} * G_\sigma))$.

A.2. Frequency scaling

This strategy removes the higher frequencies from the FBP reconstruction. This is done by setting the frequencies above a threshold f_{sc} in Fourier domain of the filter $\mathbf{h}$ equal to zero and using this filter in the standard FBP algorithm, obtaining $\mathrm{FBP}_{sc}(\mathbf{y}, \mathbf{h}_{sc}) = W^T(\mathbf{y} * \mathbf{h}_{sc})$.

For these strategies we optimized the choice of variable by computing reconstructions with a range of variables and taking the reconstruction with the highest SSIM.

ORCID iDs

Marinus J Lagerwerf https://orcid.org/0000-0003-1916-4665
Jan-Willem Buurlage https://orcid.org/0000-0001-5008-7173

References

[1] dos Santos Rolo T, Ershov A, van de Kamp T and Baumbach T 2014 In vivo X-ray cine-tomography for tracking morphological dynamics *Proc. Natl. Acad. Sci.* **111** 3921–6
[2] Xu H, Bührer M, Marone F, Schmidt T J, Büchi F N and Eller J 2020 Optimal image denoising for in situ X-ray tomographic microscopy of liquid water in gas diffusion layers of polymer electrolyte fuel cells *J. Electrochem. Soc.* **167** 104505
[3] García-Moreno F, Kamm P H, Neu T R and Banhart J 2018 Time-resolved in situ tomography for the analysis of evolving metal-foam granulates *J. Synchrotron Radiat.* **25** 1505–8
[4] De Carlo F *et al* 2018 TomoBank: a tomographic data repository for computational X-Ray science *Meas. Sci. Technol.* **29** 034004
[5] Buurlage J-W, Kohr H, Palenstijn W J and Batenburg K J 2018 Real-time quasi-3D tomographic reconstruction *Meas. Sci. Technol.* **29** 064005

[6] Buurlage J-W, Marone F, Pelt D M, Palenstijn W J, Stampanoni M, Batenburg K J and Schlepütz C M 2019 Real-time reconstruction and visualisation towards dynamic feedback control during time-resolved tomography experiments at TOMCAT *Sci. Rep.* **9** 1–11
[7] Vanrompay H, Buurlage J-W, Pelt D M, Kumar V, Zhuo X, Liz-Marzán L M, Bals S and Batenburg K J 2020 Real-time reconstruction of arbitrary slices for quantitative and in situ 3D characterization of nanoparticles *Part. Part. Syst. Char.* **37** 2000073
[8] Buzug T M 2008 *Computed Tomography: From Photon Statistics to Modern Cone-Beam CT* (Berlin: Springer)
[9] Pelt D M, Batenburg K J and Sethian J A 2018 Improving tomographic reconstruction from limited data using mixed-scale dense convolutional neural networks *J. Imaging* **4** 128
[10] Pelt D M and Batenburg K J 2013 Fast tomographic reconstruction from limited data using artificial neural networks *IEEE Trans. Image Process.* **22** 5238–51
[11] Hendriksen A A, Pelt D M and Batenburg K J 2020 Noise2Inverse: Self-supervised deep convolutional denoising for tomography *IEEE Trans. Comput. Imaging* **6** 1320–35
[12] Natterer F and Wübbeling F 2001 *Mathematical Methods in Image Reconstruction* (Society for Industrial and Applied Mathematics)
[13] Van Aarle W, Palenstijn W J, Cant J, Janssens E, Bleichrodt F, Dabravolski A, De Beenhouwer J, Batenburg K J and Sijbers J 2016 Fast and flexible X-ray tomography using the ASTRA toolbox *Opt. Express* **24** 25129–47
[14] Coban S B, Lucka F, Palenstijn W J, Van Loo D and Batenburg K J 2020 Explorative imaging and its implementation at the FleX-ray laboratory *J. Imaging* **6** 18
[15] Hastie T, Tibshirani R and Friedman J 2009 *The Elements of Statistical Learning* (Berlin: Springer Series in Statistics
[16] Lagerwerf M J, Palenstijn W J, Kohr H and Batenburg K J 2020 Automated FDK-filter selection for cone-beam CT in research environments *IEEE Trans. Comput. Imaging* Early access **6** 739–48
[17] Batson J and Royer L 2019 Noise2Self: blind denoising by self-supervision *Proc. of the 36th Int. Conf. on Machine Learning* eds Chaudhuri K and Salakhutdinov R *Long Beach, California, USA* 09–15 Jun 2019 vol 97 pp 524–33
[18] Paszke A *et al* 2017 Automatic differentiation in PyTorch *NIPS-W*
[19] Wang Z, Bovik A, Sheikh H and Simoncelli E 2004 Image quality assessment: from error visibility to structural similarity *IEEE Trans. Image Process.* **13** 600–12
[20] Hunter J D 2007 Matplotlib: a 2D graphics environment *Comput. Sci. Eng.* **9** 90–5
[21] van der Walt S, Schönberger J L, Nunez-Iglesias J, Boulogne F, Warner J D, Yager N, Gouillart E and Yu T 2014 Scikit-image: image processing in Python *PeerJ* **2** e453
[22] Russo P 2017 *Handbook of X-ray Imaging: Physics and Technology* (Boca Raton, FL: CRC press)