Universiteit
Leiden
The Netherlands

A computationally efficient reconstruction algorithm for circular cone-beam computed tomography using shallow neural networks

Lagerwerf, M.J.; Pelt, D.M.; Palenstijn, W.J.; Batenburg, K.J.

Citation

Lagerwerf, M. J., Pelt, D. M., Palenstijn, W. J., & Batenburg, K. J. (2020). A computationally efficient reconstruction algorithm for circular cone-beam computed tomography using shallow neural networks. *Journal Of Imaging*, *6*(12). doi:10.3390/jimaging6120135

Version:	Publisher's Version
License:	Creative Commons CC BY 4.0 license
Downloaded from:	https://hdl.handle.net/1887/3304530

Note: To cite this publication please use the final published version (if applicable).

Journal of *Imaging*

MDPI

Article

A Computationally Efficient Reconstruction Algorithm for Circular Cone-Beam Computed Tomography Using Shallow Neural Networks

Marinus J. Lagerwerf [1,*], Daniël M. Pelt [1,2], Willem Jan Palenstijn [1] and Kees Joost Batenburg [1,2]

[1] Centrum Wiskunde & Informatica, Science Park 123, 1098 XG Amsterdam, The Netherlands; d.m.pelt@liacs.leidenuniv.nl (D.M.P.); wjp@cwi.nl (W.J.P.); k.j.batenburg@cwi.nl (K.J.B.)

[2] Leiden Institute of Advanced Computer Science, Universiteit Leiden, 2333 CA Leiden, The Netherlands

* Correspondence: m.j.lagerwerf@cwi.nl

Received: 17 November 2020; Accepted: 4 December 2020; Published: 8 December 2020

Abstract: Circular cone-beam (CCB) Computed Tomography (CT) has become an integral part of industrial quality control, materials science and medical imaging. The need to acquire and process each scan in a short time naturally leads to trade-offs between speed and reconstruction quality, creating a need for fast reconstruction algorithms capable of creating accurate reconstructions from limited data. In this paper, we introduce the Neural Network Feldkamp–Davis–Kress (NN-FDK) algorithm. This algorithm adds a machine learning component to the FDK algorithm to improve its reconstruction accuracy while maintaining its computational efficiency. Moreover, the NN-FDK algorithm is designed such that it has low training data requirements and is fast to train. This ensures that the proposed algorithm can be used to improve image quality in high-throughput CT scanning settings, where FDK is currently used to keep pace with the acquisition speed using readily available computational resources. We compare the NN-FDK algorithm to two standard CT reconstruction algorithms and to two popular deep neural networks trained to remove reconstruction artifacts from the 2D slices of an FDK reconstruction. We show that the NN-FDK reconstruction algorithm is substantially faster in computing a reconstruction than all the tested alternative methods except for the standard FDK algorithm and we show it can compute accurate CCB CT reconstructions in cases of high noise, a low number of projection angles or large cone angles. Moreover, we show that the training time of an NN-FDK network is orders of magnitude lower than the considered deep neural networks, with only a slight reduction in reconstruction accuracy.

Keywords: tomography; circular cone-beam CT; machine learning; neural network; multilayer perceptron; Feldkamp–Davis–Kress (FDK); reconstruction algorithm

1. Introduction

Circular cone-beam (CCB) Computed Tomography (CT) has become an integral part of non-destructive imaging in a broad spectrum of applications, such as industrial quality control [1], materials sciences [2,3] and medical imaging [4,5]. Especially in industrial and medical applications, the scanning time, reconstruction time and the scanning dose are limited resources. Such limitations lead to trade-offs between computation time and scanning time—i.e., number of projections, noise level—on the one hand, and reconstruction accuracy on the other hand. Additionally, CT reconstruction has become a *big data* problem due to the development of readily available high-resolution CT-scanners [6–8]. This stresses the need for computationally efficient reconstruction methods that are applicable to a broad spectrum of high-resolution problems and produce accurate results from data with high noise levels, low numbers of projection angles or large cone angles.

In practice, if computational efficiency is a constraint and especially for high-resolution problems, *direct methods* (e.g., the filtered backprojection (FBP) algorithm [9], the Feldkamp–Davis–Kress (FDK) algorithm [10] and the Katsevich algorithm [11]) are still the common choice of reconstruction method [12]. While *iterative methods* have been shown to be more accurate for noisy and limited data problems [13–18], they have a significantly higher computational cost. Consequently, there have been efforts to improve the accuracy of direct methods by computing data-specific or scanner-specific filters [19–23]. Although these strategies do improve the reconstruction accuracy, they also add significant computational effort or are specific to one modality, e.g., tomosynthesis [24].

An emerging approach for improving direct methods is to use *machine learning* to remove artifacts from the reconstructions. The idea is to use high-quality reconstructions to train a neural network that removes artifacts from low-quality reconstructions using a supervised learning approach. This *post-processing* approach has shown promising results for computed tomography using deep neural networks (DNNs) [25–27]. Deep neural network structures contain a large number of layers, leading to millions of trainable parameters and, therefore, require a large amount of training data [28]. This is problematic in CT imaging, since there is often a limited amount of training data available, e.g., due to scanning time, dose, and business-related concerns. Moreover, for the available data, there are often no reference datasets or annotations available [29]. The large amount of training data and large number of parameters also lead to long training times. While for standard 2D networks, the training time ranges between a couple of hours and a couple of days (see Section 5.1.2), for 3D networks the training time becomes prohibitively long [30] (i.e., weeks). Therefore, to apply post-processing to 3D problems, the reconstruction volume can be considered as a stack of 2D problems [26,31] for which one 2D network is trained and then applied in a *slice-by-slice* fashion to the 3D volume. Although this strategy reduces the training time and the training data constraints, applying a 2D network to all slices can still be computationally intensive due to the number of slices in the 3D volume. A more in-depth discussion on current developments related to machine learning methods in CT imaging is given in Section 2.

In this work, we propose the Neural Network FDK (NN-FDK) reconstruction algorithm. It is a direct reconstruction method that is designed to produce accurate results from noisy data, data with a low number of projection angles, or a large cone angle, but still maintains a similar computational efficiency and scalability as the standard FDK algorithm. Moreover, the algorithm has a fast training procedure, and requires a limited amount of training data.

The NN-FDK algorithm is an adaptation of the standard FDK algorithm using a shallow multilayer perceptron network [32] with one fully connected hidden layer, a low number of trainable parameters and low memory constraints. We will show it is possible to interpret the weights of the first layer of the perceptron network as a set of learned filters for the FDK algorithm. We can then use the FDK algorithm to evaluate the network efficiently for all voxels simultaneously to arrive at an accurate reconstruction for the CCB CT problem.

The NN-FDK algorithm is an extension of the method proposed in [33] for the Filtered Backprojection (FBP) algorithm [9]. The derivation of the approach outlined in [33] relies on the shift-invariance property of the FBP algorithm. We will show that, although the FDK algorithm does not have this shift-invariance property, we can derive a similar method for the FDK algorithm. Moreover, the proposed strategy can be extended to any linear filtered backprojection type reconstruction method.

Using both simulated and experimental data, we compare the proposed method with the standard FDK algorithm, SIRT [34] with a nonnegativity constraint (SIRT$^+$), which is a commonly used iterative algorithm for CT problems, and two 2D deep neural networks (U-net [31] and MSD-net [26]) trained to remove reconstruction artifacts from slices of standard FDK reconstruction. We show that the NN-FDK algorithm is faster to evaluate than all but the standard FDK algorithm and orders of magnitude faster to train than the considered DNNs, with only a slight reduction in reconstruction accuracy compared to the DNNs.

The paper is structured as follows. In Section 3, we give definitions and introduce our method. In Section 4, we introduce the data and the parameters used for the experiments. The experiments and their results are shown and discussed in Section 5. The paper is summarized and concluded in Section 6.

2. Related Work

Using *machine learning* methods is an emerging approach in CT imaging [29]. *Deep learning* methods have shown promising results for many applications within the development of CT reconstruction methods [35]. For the sake of exposition, we split these machine learning approaches into two categories: (i) Improving standard reconstruction methods by replacing components of the reconstruction method with networks specifically trained for the application; and (ii) improving the image quality of reconstructions computed with existing reconstruction methods by training neural networks to perform *post-processing* in order to remove artifacts or reduce noise.

Examples of the first strategy (improving standard reconstruction methods) applied to iterative methods are the learned primal-dual reconstruction algorithm [36,37], variational networks [38,39], plug and play priors [40–42], and learned regularizers [43,44]. These methods achieve promising results in reconstruction accuracy and generalizability. However, their high computational cost limits the applicability if high throughput is required. Examples for this strategy applied to direct methods are the NN-FBP method [33], and also the NN-FDK method introduced in this paper. These methods are designed to improve the image quality of direct methods for data with limitations (e.g., data with noise or a low number of projection angles), while maintaining their computational efficiency.

Examples of the second strategy (learned post-processing) have demonstrated substantial improvements in reconstruction quality for CT imaging [25,28,31,35]. This is aided by the fact that the post-processing problem can be viewed as a classic imaging problem—e.g., denoising, segmentation, inpainting, classification— for which many effective machine learning methods have already been developed [45–47]. Although the general trend is towards deeper networks to make such networks more expressive [48], this can lead to problems with scalability for large 3D image datasets.

The rise in popularity of machine learning in CT is driven by the increased computational possibilities and although these advances are sufficient to handle most 2D problems, scaling towards 3D problems can be problematic, due to memory constraints. This is illustrated in Section 5.1.1, where we plotted the memory constraints for applying a 2D and 3D U-net and MSD-net in terms of gigabytes (GB) of memory as a function of the size of the image. This shows that, in theory, one could apply a 2D MSD-net to images of 7500×7500 pixels (with a 24GB GPU), but in 3D, this limit lies around $400 \times 400 \times 400$ voxels. Considering that CT problems range between $256 \times 256 \times 256$ (small image size) up to $4096 \times 4096 \times 4096$ images, this gives an indication that scalability can become an issue, especially for 3D problems.

When applying machine learning techniques for improving the reconstruction quality in CT, a balance must be struck between image quality, running time, and memory requirements. Here, we propose a method that achieves relatively high accuracy, while also being computationally efficient and scalable.

3. Method

The NN-FDK algorithm is a reconstruction algorithm with a machine learning component, meaning that a number of parameters of the reconstruction algorithm are optimized through *supervised learning* [49]. Similar to the network presented in [33], the *NN-FDK network* is a two layer neural network with a hidden layer and an output layer. We design the network such that it reconstructs one single voxel, but handles all voxels in a similar manner. This means that we only have to train one network for a full reconstruction. We consider the NN-FDK algorithm to have three parts: The *NN-FDK network*, the *NN-FDK reconstruction algorithm* and the *training process*.

We introduce the reconstruction problem, FDK algorithm, a filter approximation method and the definition of a perceptron in Section 3.1. In Section 3.2, we provide the *NN-FDK reconstruction algorithm* and derive from this algorithm the *NN-FDK network*. The input of the network that is needed in the *training process* is a pre-processed version of the input of the reconstruction algorithm. In Section 3.3, we discuss how to compute this pre-processing step for all voxels simultaneously and we introduce the optimization problem and related notation for the training process. Lastly, we summarize and discuss the characteristics of the method in Section 3.4.

3.1. Preliminaries

3.1.1. Reconstruction Problem

In this paper, we focus exclusively on the circular cone-beam (CCB) geometry, where the object rotates with respect to a point source and a planar detector, acquiring 2D cone-beam projections. The reconstruction problem for the CCB geometry can be modeled by a system of linear equations

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

where $\mathbf{x} \in \mathbb{R}^n$ is the vector describing the reconstruction (i.e., every element coincides with a voxel value), $\mathbf{y} \in \mathbb{R}^m$ is the vector describing the measured projection data, and $W \in \mathbb{R}^{m \times n}$ is a discretized version of the *cone-beam transform* or *forward projection*. For the sake of simplicity, we assume that the volume consists of $n = N \times N \times N$ voxels and the detector consists of $N \times N$ pixels. We denote the number of angles by N_a, so we have $m = N_a \times N \times N$.

3.1.2. FDK Algorithm & Filter Approximation

The FDK algorithm, as presented in [10], is a filtered backprojection-type algorithm that solves the CCB reconstruction problem (1) approximately. First, for each projection angle, it applies a *reweighting* step, $r : \mathbb{R}^{N_a \times N \times N} \to \mathbb{R}^{N_a \times N \times N}$, that adapts the cone-beam data such that it approximately behaves as fan-beam data. Second, it applies a *filtering* step, that convolves the data with a one-dimensional *filter* $\mathbf{h}$ in a line-by-line fashion, $(- * -)_{1\mathrm{D}} : \mathbb{R}^{2N} \times \mathbb{R}^{N_a \times N \times N} \to \mathbb{R}^{N_a \times N \times N}$. Last, it applies a *backprojection* step. This step transforms the filtered projection data to the image domain. Using the notation of (1), the FDK algorithm is given by

$$\mathrm{FDK}(\mathbf{y}, \mathbf{h}) = W^T(\mathbf{h} * r(\mathbf{y}))_{1\mathrm{D}}, \tag{2}$$

with W^T the transpose of W. The operator W^T is also known as the *backprojection operator*.

In [22,23,33], exponential binning is used to approximate filters, leading to $N_e \approx \log N$ coefficients to describe a filter. This approximation can be seen as a matrix $E \in \mathbb{R}^{2N \times N_e}$ applied to a coefficient vector $\mathbf{h}_e \in \mathbb{R}^{N_e}$:

$$\mathbf{h} \approx E\mathbf{h}_e. \tag{3}$$

The implementation details of this filter approximation can be found in [23].

3.1.3. Perceptron

In a similar manner as in [32], we define a *perceptron* or *node* $\mathsf{P} : \mathbb{R}^l \to \mathbb{R}$ as a non-linear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ applied to a weighted sum of the input $\eta \in \mathbb{R}^l$ with the weights $\xi \in \mathbb{R}^l$ and a bias $b \in \mathbb{R}$:

$$\mathsf{P}_{\xi,b}(\eta) = \sigma(\eta \cdot \xi - b) \tag{4}$$

In this paper, we will only consider the sigmoid function as an activation function, i.e., $\sigma(t) = 1/(1+e^{-t})$.

A multilayer perceptron is a network structure containing two types of layers with perceptrons, where each perceptron operates on the outputs of the previous layer. These layers are, in order, any number of *hidden layers*, and the *output layer*. Note that the number of hidden layers and number of hidden nodes N_h in these layers can be chosen freely.

3.2. Reconstruction Algorithm & Network Design

We formulate the NN-FDK reconstruction algorithm in a similar fashion as the NN-FBP method in [33]. The NN-FDK reconstruction algorithm consists of N_h individual FDK algorithms executed on the input data y, each using its own (exponentially binned) filter $\mathbf{h}_e^k \in \mathbb{R}^{N_e}$. It combines these N_h volumes into a single reconstruction, using point-wise application of the activation function σ and an output perceptron with parameters $b_o, b_k \in \mathbb{R}$, and $\xi \in \mathbb{R}^{N_h}$.

We use $\theta = (\xi, b_o, \mathbf{h}_e^k, b_k)$ as short-hand for the full set of parameters of the NN-FDK reconstruction algorithm. The full algorithm (Algorithm 1) is then given by the following equation.

$$\text{NN-FDK}_\theta(\mathbf{y}) = \sigma\Big(\sum_{k=1}^{N_h} \xi_k \sigma \Big(\text{FDK}(\mathbf{y}, E\mathbf{h}_e^k) - b_k \Big) - b_o \Big) \tag{5}$$

The FDK algorithm is a bilinear map in the input projection data and the used filter. Therefore, for fixed input projection data $\mathbf{y}$ and an expanded exponentially binned filter $E\mathbf{h}_e$, the FDK algorithm can be written as a linear map $F_\mathbf{y}$ applied to $E\mathbf{h}_e$. The product $F_\mathbf{y}E$ can be considered as a matrix of size $N^3 \times N_e$, and the v-th voxel of the output of the FDK algorithm is given by the inner product of $\mathbf{h}_e$ with $(F_\mathbf{y}E)_{v:}$, the v-th row of the matrix $F_\mathbf{y}E$. This leads to the following:

$$\begin{aligned}
(\text{NN-FDK}_\theta(\mathbf{y}))_v &= \sigma\Big(\sum_{k=1}^{N_h} \xi_k \sigma \Big((F_\mathbf{y}E\mathbf{h}_e^k)_v - b_k \Big) - b_o \Big), && (6)\\
&= \sigma\Big(\sum_{k=1}^{N_h} \xi_k \sigma \Big((F_\mathbf{y}E)_{v:}\mathbf{h}_e^k - b_k \Big) - b_o \Big), && (7)\\
&= \mathsf{P}_{\xi,b_o}\left(\left[\mathsf{P}_{\mathbf{h}_e^k,b_k}((F_\mathbf{y}E)_{v:}) \right]_k \right). && (8)
\end{aligned}$$

Therefore, we define the two-layer perceptron network $\mathsf{N}_\theta : \mathbb{R}^{N_e} \to \mathbb{R}$:

$$\mathsf{N}_\theta(\mathbf{q}) = \mathsf{P}_{\xi,b_0}\left(\left[\mathsf{P}_{\mathbf{h}_e^k,b_k}(\mathbf{q}) \right]_k \right). \tag{9}$$

This is our NN-FDK network, and as we derived above, it has the following relationship with the NN-FDK reconstruction algorithm:

$$\mathsf{N}_\theta((F_\mathbf{y}E)_{v:}) = (\text{NN-FDK}_\theta(\mathbf{y}))_v. \tag{10}$$

This relationship shows that we can *evaluate* the NN-FDK reconstruction algorithm efficiently on full input projection data at once, but also *train* the NN-FDK network efficiently with each *individual* voxel $(\mathbf{x}_{\text{HQ}})_v$ in a high-quality reconstruction yielding a training pair with input $(F_\mathbf{y}E)_{v:}$ and target $(\mathbf{x}_{\text{HQ}})_v$. A schematic representation of the network is given in Figure 1.

Note that we arrive at the same network structure as found in [33] for FBP, using only the properties that the FDK algorithm is a bilinear map in the data and the filter, and that all operations can be applied point-wise. Using this reasoning, we can derive a similar network structure for any FBP-type method satisfying these conditions.

Even though we use the same network structure as [33], the way we compute inputs to the network is different. In [33], the input to the NN-FBP network is explicitly calculated by shifting and adding

projection data for each reconstruction pixel. The FDK algorithm has additional weighting factors and lacks the shift-invariance property, which makes the approach presented in [33] not directly applicable. In the next section, we detail an alternative method to compute the input. The same approach could be applied to the NN-FBP method, similarly simplifying the network input computations.

Algorithm 1 Neural Network Feldkamp–Davis–Kress (NN-FDK) reconstruction algorithm

1: Given a set of parameters, $\theta := \left(\xi, b_o, \mathbf{h}_e^k, b_k\right)$.
2: Compute H_k for all nodes k of the hidden layer:
3: **for** $k = \{1, 2, .., N_h\}$ **do**
4: $\quad \mathrm{H}_k(y) = \sigma\left(\mathrm{FDK}(\mathbf{y}, E\mathbf{h}_e^k) - b_k\right)$
5: **end for**
6: Compute the output of the output layer:
$\mathrm{NN\text{-}FDK}_\theta(\mathbf{y}) = \sigma\left(\sum_{k=1}^{N_h} \xi_k H_k(\mathbf{y}) - b_o\right)$

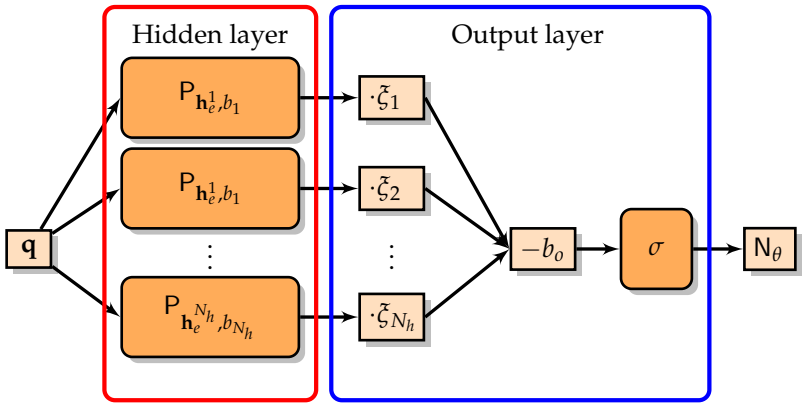

Figure 1. Schematic representation of the NN-FDK network, $\mathsf{N}_\theta : \mathbb{R}^{N_e} \to \mathbb{R}$, with N_h hidden nodes. Note that if we take $q = (F_\mathbf{y}E)_{v:}$ we get $q \cdot \mathbf{h}_e^k = (\mathrm{FDK}(\mathbf{y}, E\mathbf{h}_e^k))_v$ in the perceptrons of the hidden layer and the output of the network is equal to the v-th voxel of the NN-FDK reconstruction algorithm.

3.3. Training Process

3.3.1. Training and Validation Data

We will train our network using supervised learning, where we assume that we have N_{TD} and N_{VD} datasets available for training and validation, respectively. These datasets consist of low-quality tomographic input data and a high-quality reconstruction from which we randomly draw a total of N_{T} training pairs and N_{V} validation pairs. Note that we ensure that every drawn pair is unique and that an equal number of pairs is taken from each dataset. Moreover, to avoid selecting too many training pairs from the background, we only take training pairs from a region of interest (ROI) around the scanned object. This ROI is defined from the high-quality reconstruction as the voxels in the reconstructed object plus a buffer of roughly 0.2 N voxels around it.

Recall from the previous section that given low-quality tomographic data $\mathbf{y}$ and a high-quality reconstruction $\mathbf{x}_{\mathrm{HQ}}$, the matrix $F_\mathbf{y}E$ contains each input vector $Z = \left(F_\mathbf{y}E\right)_{v:} \in \mathbb{R}^{N_e}$ corresponding to the target voxel $O = (\mathbf{x}_{\mathrm{HQ}})_v$. However, due to memory constraints, $F_\mathbf{y}E$ cannot be computed directly as a matrix product. Therefore, we observe that each column of $F_\mathbf{y}E$ is an FDK reconstruction with a specific filter:

$$(F_\mathbf{y}E)_{:j} = F_\mathbf{y}E\mathbf{e}_j = \mathrm{FDK}(\mathbf{y}, E\mathbf{e}_j), \tag{11}$$

with $\mathbf{e}_j \in \mathbb{R}^{N_e}$ the unit vector with all entries equal to zero except for the j-th element.

3.3.2. Learning Problem

The parameters of the NN-FDK network are learned by finding the set of parameters $\theta^\star$ that minimize the *loss function* $\mathcal{L}$ on the training set. We minimize the ℓ^2-distance between the network output and the target voxel for all training pairs in T:

$$\theta^\star = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta, T) = \underset{\theta}{\operatorname{argmin}}\, \tfrac{1}{2} \sum_{j=1}^{N_\mathrm{T}} \left(O_j - \mathsf{N}_\theta(Z_j)\right)^2 . \tag{12}$$

To minimize the loss function, we use a quasi-Newton optimization scheme, the *Levenberg–Marquardt algorithm* (LMA) as proposed in [50,51]. This is a combination of gradient descent and the Gauss-Newton algorithm, improving the stability of Gauss-Newton while retaining its fast convergence and it is specifically designed to minimize a non-linear least squares problem such as (12). Note that the small number of parameters of the proposed network allows us to use such a method. Lastly, to avoid overfitting we check whether every update of the parameters also reduces the loss function on the validation set. We discuss the specifics of this algorithm in Appendix B.

3.4. Method Characteristics & Comparison

To conclude the method section, we compare the characteristics of the NN-FDK algorithm to those of several other methods. These methods are two 2D post-processing DNNs (U-net [31] and MSD-net [28]) applied in a slice-by-slice fashion, the SIRT$^+$ algorithm [34] and the FDK algorithm. We focus our discussion on the goals formulated in Section 1 and show a summary of this comparison in Table 1. The reconstruction accuracy will be discussed in Section 5.

Table 1. Comparison of reconstruction methods with respect to the goals formulated in Section 1. We consider a DNN to be 2D deep convolutional neural network (U-net and MSD-net) applied in slice-by-slice fashion to a standard FDK reconstruction. Reconstruction accuracy is defined as the accuracy of a method when reconstructing low-quality data, e.g., data with high noise or a low number of projection angles.

	Reconstruction		**Training**	
Method	**Time**	**Accuracy**	**Data**	**Time**
NN-FDK	++	?	++	+++
DNN	±	+++	±	- - -
FDK	+++	- -		
SIRT$^+$	- -	+		

Method comparison: Goals.

3.4.1. Computational Efficiency

We approximate the reconstruction time by counting how many times it has to evaluate its most expensive computations. For simplicity, we assume that a backprojection takes approximately the same time as a forward projection, T_BP.

- **FDK:** The FDK algorithm consist of one reweighting, filtering and backprojection step, i.e.,:

$$T_\mathrm{FDK} \approx T_\mathrm{BP}. \tag{13}$$

- **NN-FDK:** The NN-FDK algorithm performs one FDK reconstruction per hidden node N_h. Therefore, the reconstruction time becomes:

$$T_\mathrm{NN\text{-}FDK} \approx N_h T_\mathrm{BP}. \tag{14}$$

- **SIRT$^+$:** The SIRT$^+$ method evaluates a forward and backprojection for each iteration. For N_{iter} iterations, the reconstruction time becomes:

$$T_{\text{SIRT}^+} \approx 2N_{\text{iter}}T_{\text{BP}}. \tag{15}$$

- **DNN:** To evaluate a DNN on a full 3D volume, an FDK reconstruction is performed and a 2D network is applied per slice of the FDK reconstruction.

$$T^{3D}_{\text{DNN}} \approx T_{\text{BP}} + NT^{2D}_{\text{DNN}}, \tag{16}$$

with T^{2D}_{DNN} the time it takes to apply a 2D DNN and T^{3D}_{DNN} the time to do a full DNN reconstruction.

On a modern Nvidia GeForce GTX 1080Ti (11 GB) GPU and with $N = 1024$ and $N_a = 360$, we found in our experiments that $T_{\text{BP}} \approx 10$ s and $T^{2D}_{\text{DNN}} \approx 0.5$ s.

Comparing the reconstruction times, we see that NN-FDK is similar to FDK when the number of nodes N_h is small, which is the case since we will take $N_h = 4$ (see Section 4.3). For DNNs, the computational load of applying a 2D network leads to relatively high reconstruction times compared to the FDK algorithm. Lastly, we note that the number of iterations N_{iter} often lies between the 20 and 200, making SIRT$^+$ several times slower than the (NN-)FDK algorithm.

3.4.2. Number of Trainable Parameters

The number of trainable parameters is closely related to the amount of training data required to train a network [28]. From the definition of the NN-FDK network (5), we can compute the number of trainable parameters $|\theta|$:

$$|\theta| = (N_e + 2)N_h + 1, \tag{17}$$

with $N \gg N_h, N_e > 0$. Taking $N_h = 4$ and $N = 1024$ gives $|\theta| = 61$, which is several orders of magnitude lower than the typical numbers of parameters in a DNN (several tens of thousands to millions).

3.4.3. Training Time

In the training step, a solution to the minimization problem (12) is computed. For the NN-FDK algorithm, this problem has N_{T} samples and $|\theta|$ unknowns. In a similar fashion, we can formulate a least squares problem for training a DNN. Even assuming that we only take the same number of training samples to train the DNNs, this least squares problem is already orders of magnitude larger than that for NN-FDK due to the difference in the number of trainable parameters. Moreover, the LMA (the algorithm used to train NN-FDK) approaches quadratic convergence, which means it will need fewer iterations to converge than a first-order scheme such as ADAM [52], which is often used for training DNNs. Considering these two observations, we expect the training time of the NN-FDK algorithm to be lower than the training time of the DNNs.

4. Experimental Setup

We carried out a range of experiments to assess the performance of the NN-FDK algorithm with respect to the goals formulated in Section 1 compared to several alternative methods. In this section, we introduce the setup of these experiments. We describe the simulated data in Section 4.1 and the experimental data in Section 4.2. In Section 4.3, we discuss the specific network structure for the NN-FDK algorithm and the training parameters used. Finally, we give the quantitative measures we use to compare the reconstruction in Section 4.4.

4.1. Simulated Data

We consider two types of phantom families for the simulated data experiments: the *Fourshape phantom family* and the *Random Defrise phantom family*. Examples are shown in Figure 2a,b, respectively. The Fourshape phantom family contains three random occurrences of each of four types of objects: an ellipse, a rectangle, a Gaussian blob and a Siemens star. For evaluation and visualization of the reconstructions, we fixed one realization that clearly shows at least one of all the four objects and we will refer to this phantom as the *Fourshape test phantom*. The Random Defrise phantom family is a slight adaptation of the phantom introduced in [53], which is a common phantom for assessing the influence of imaging artifacts due to the cone angle. Here, we vary the intensities, orientations and sizes of the disks making sure they do not overlap. Again, we define a test phantom for evaluation and visualization, which is in this case, the standard *Defrise phantom* without alternating intensities (right in Figure 2b). To simulate realistic settings, we scale the phantoms to fit inside a 10 cm cube, and use an attenuation coefficient of $\mu = 0.22\ \mathrm{cm}^{-1}$, approximating that of various common plastics at 40 keV [54]. These phantoms are defined through geometric parameters, and can, therefore, be generated for any desired N. For our experiments, we will take $N = 1024$. Details about how we generate the data are given in Appendix A.1.

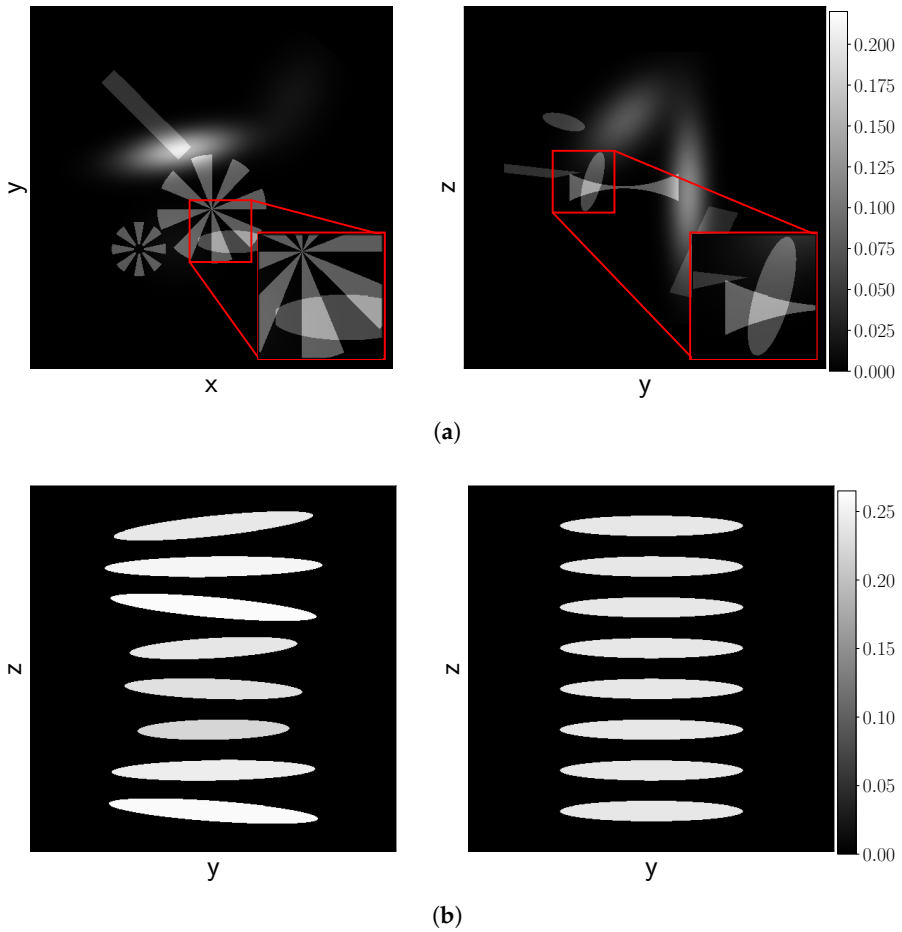

Figure 2. Examples simulated data. (**a**) Slices, (**Left**) $z = 0$, (**Right**) $x = 0$, of the Fourshape test phantom. This phantom is designed such that at least one of all objects can clearly be observed in the slices. (**b**) The $x = 0$ slice for a Random Defrise phantom (**Left**) and the standard Defrise phantom without alternating intensities from [53] (**Right**).

To compute a high-quality reconstruction $\mathbf{x}_{\mathrm{HQ}}$ that can be used as target for training (recall Section 3.3), we consider a simulated dataset with $N_a = 1500$ projection angles, low noise ($I_0 = 2^{20}$ emitted photon count) and cone angle of 0.6 degrees and reconstruct this problem with the standard FDK algorithm using a Hann filter [9].

4.2. Experimental Data

For experimental data, we consider a set of CT scans that were recorded using the custom-built and highly flexible FleX-ray CT scanner, developed by XRE NV and located at CWI [55]. This scanner has a flat panel detector with 972×768 pixels and a physical size of 145.34×114.82 mm. This set of 42 scans was set up to create high-noise reconstruction problems and low-noise reconstruction problems with a low number of projection angles.

We acquired high-dose (low noise) and low-dose (high noise) scans of 21 walnuts. The datasets contain 500 equidistantly spaced projections over a full circle. The distance from the center of rotation to the detector was set to 376 mm and the distance from the source to the center of rotation was set to 463 mm. The scans were performed with a tube voltage of 70 kV. The high-dose scan was collected with a tube power of 45 W and an exposure time of 500 ms per projection. The low-dose scan was collected with a tube power of 20 W and an exposure time of 100 ms per projection. To create a low noise reconstruction problem with a low number of projection angles, we considered the high-dose scan but only took every 16-th projection angle. As high-quality reference reconstructions, we used SIRT$^+$ reconstructions with 300 iterations (SIRT$^+_{300}$) of the high-dose scans with all available projection angles ($N_a = 500$). We will refer to these reconstructions as the *gold standard* reconstruction and we show such a reconstruction in Figure 3. These datasets are available at Zenodo [56].

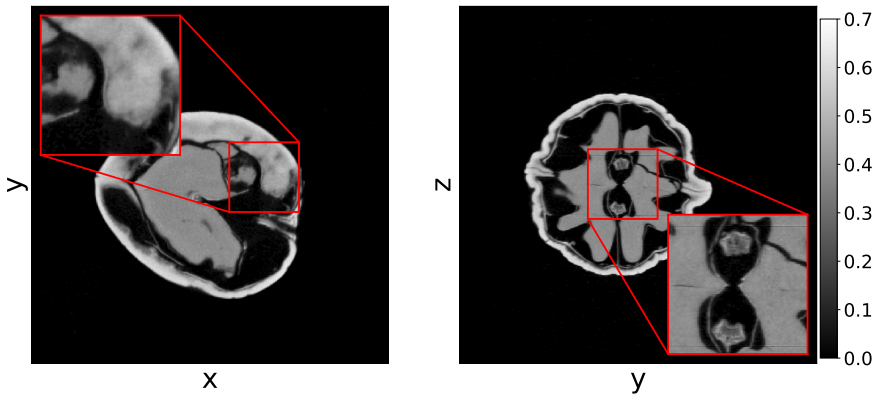

Figure 3. The $z = 0$ (**Left**) and $y = 0$ (**Right**) slice of the gold standard reconstruction of the high-dose dataset of the 21st walnut with full number of projection angles. The projection data are acquired using the FleX-ray scanner located at the CWI [55]. The horizontal line artifacts visible in the right figure are due to imperfections in the projection data and not due to the reconstruction process.

4.3. Parameter Settings NN-FDK

In our initial experiments, we found that taking more FDK-perceptrons improved the accuracy of the networks, at the cost of increasing the training and reconstruction time. More specifically, the reconstruction time scales linearly with the number of perceptrons, whereas the reconstruction accuracy only shows marginal improvements for more than $N_h = 4$ FDK-perceptrons, which is similar to the findings in [33]. Therefore, we fix the number of FDK-perceptrons at $N_h = 4$, and denote the resulting network structure by NN-FDK$_4$.

We found that, similar to the findings in [33], taking $N_{\mathrm{T}} = 10^6$ voxels for training and $N_{\mathrm{V}} = 10^6$ for validation is sufficient for training an NN-FDK network. As test data, we randomly generate 20 datasets for simulated data. For the experimental data, we use all datasets that were not used in the training process.

The network structures and training procedure used for the U-nets and MSD-nets are discussed in Appendix A.2.

4.4. Quantitative Measures

To quantify the accuracy of the reconstructions, we consider two measures, the test set error (TSE) and the structural similarity index (SSIM). These measures compare the reconstructed image $\mathbf{x}_r$ to a high-quality reconstruction $\mathbf{x}_{\text{HQ}}$ on the ROI (as discussed in Section 3.3).

The TSE is the average loss (as defined in (12) in Section 3.3) of the test set, where the test set is all the voxels defined in the ROI of $\mathbf{x}_{\text{HQ}}$:

$$\text{TSE}(\mathbf{x}_r, \mathbf{x}_{\text{HQ}}) = \frac{1}{N_{\text{ROI}}} \mathcal{L}(\mathcal{I}_{\text{ROI}}(\mathbf{x}_{\text{HQ}}), \theta), \tag{18}$$

$$= \frac{1}{2N_{\text{ROI}}} \left\| \mathcal{I}_{\text{ROI}}(\mathbf{x}_{\text{HQ}} - \mathbf{x}_r) \right\|_2^2. \tag{19}$$

with $\mathcal{I}_{\text{ROI}} : \mathbb{R}^{N^3} \to \mathbb{R}^{N^3}$ the masking function for the ROI and N_{ROI} the number of voxels in the ROI.

The SSIM [57] is implemented based on the scikit-image 0.13.1 [58] package, where all the constants are set to default and the filter is uniform, with a width of 19 pixels.

5. Results and Discussion

5.1. Scalability

5.1.1. Memory Scaling

The required memory to store all intermediate images for a forward pass of a 2D or a 3D U-net and MSD-net as a function of the input image size is shown in Figure 4. Considering that CT imaging problems typically range from $256 \times 256 \times 256$ up to $4096 \times 4096 \times 4096$, we conclude from these figures that full 3D networks do not fit into GPU memory for higher resolutions and that even for 2D U-nets, not all resolutions fit into the GPU. As a forward pass of the NN-FDK algorithm requires only one additional reconstruction volume compared to the FDK algorithm, the memory requirements of the NN-FDK algorithm are roughly two-times the memory required by the FDK algorithm. (Technically, a forward pass of the NN-FDK algorithm can be done for every voxel separately; however, for the sake of comparison we assume a forward pass is for a full reconstruction volume.)

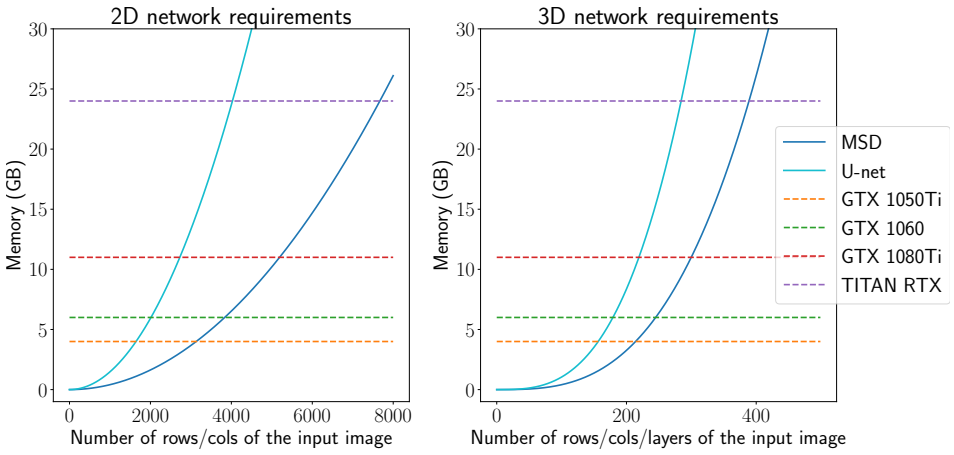

Figure 4. The required memory to store all intermediate images for applying a 2D and 3D U-net and MSD-net as a function of the input image size.

5.1.2. Training Time

In Figure 5, we compare the training processes by plotting the progress of the network training (measured by the TSE) as a function of the number of voxels that the network has seen during training. We see that the NN-FDK has seen 1.1×10^8 voxels when it converges to TSE $= 1.4 \times 10^{-5}$, whereas, MSD-net and U-net have seen 5.1×10^8 voxels and 3.2×10^9 voxels, respectively, at the point they first achieve a similar TSE. Important to note is that both U-net and MSD-net are not yet converged when they match the TSE of NN-FDK, and in general, the DNNs achieve lower TSEs than NN-FDK.

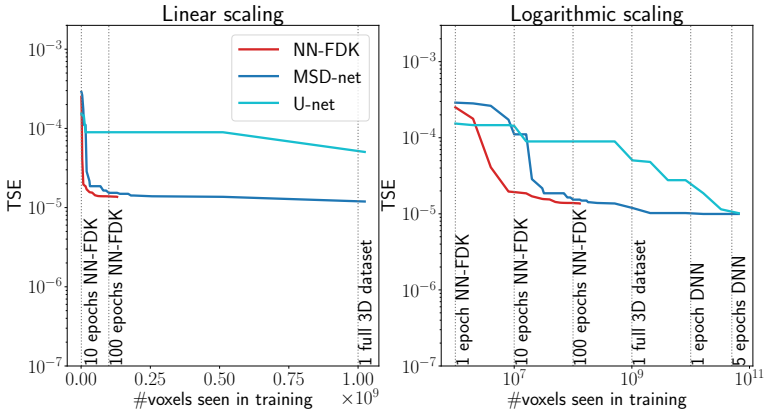

Figure 5. The test set error (TSE) as a function of the number of voxels the training process has seen. We report the lowest TSE until that point. The networks are trained on randomly generated Fourshape phantoms with size $N = 1024$, $N_a = 32$ projection angles and no noise. (**Left**) Linear scaling in the number of voxels ranging from 1 epoch for the NN-FDK (10^6 voxels), to one full 3D dataset (10^9 voxels). (**Right**) Logarithmic scaling in the number of voxels. Ranging from 1 epoch for the NN-FDK network (10^6 voxels) to 5 epochs for a DNN (5×10^{10} voxels).

In Table 2, we show various timings and properties with respect to the training process. These timings are recorded using one Nvidia GeForce GTX 1080Ti with 11GB memory. We define a converged training process as 100 epochs without improvement on the validation set error and the number of epochs to converge as the epoch with the lowest validation set error during a converged training process. From these results, we see that the size of the training problem influences the time per epoch as an NN-FDK epoch is sub-second and the time per epoch for DNNs is in the range of hours.

In practice, we observed that after 2 days of training for the DNNs, any additional training only achieved marginal improvements. Therefore, in the following experiments, we train all DNNs for 2 days with one Nvidia GeForce GTX 1080Ti GPU, unless mentioned otherwise.

Table 2. Timings and properties of the considered training processes. We define a converged training process as 100 epochs without improvement on the validation set error. The number of epochs to converge is, therefore, the epochs computed of such a process minus 100. The training was performed using one Nvidia GeForce GTX 1080Ti GPU (11 GB).

	NN-FDK$_4$	**MSD-net**	**U-net**
Voxels seen in one epoch	1×10^6	1.1×10^{10}	1.1×10^{10}
Time per epoch	0.1336 s	0.95 h	2.36 h
Time to converge	28 s	±10 days	±14 days
Epochs to converge	110	128	42
Epochs in 2 days	-	45	18

Training process.

5.1.3. Reconstruction Time

We measured the average reconstruction times and corresponding standard deviation over 120 reconstructions with resolution $N^3 = 1024^3$ and $N_a = 360$ projection angles. These reconstructions are computed using one Nvidia GeForce GTX 1080Ti with 11 GB memory. The results are shown in Table 3. The subscript 200 for SIRT$^+$ denotes the number of iterations that were used for the reconstruction. We define the reconstruction time as the time it takes to compute the full 3D volume. This means for U-net and MSD-net, an FDK reconstruction needs to be computed and the network needs to be applied $N = 1024$ times to a 2D slice. Although every application can be done within a second (U-net ≈ 0.3 s, MSD-net ≈ 0.7 s) this leads to long reconstruction times.

Table 3. Average and standard deviation of the reconstruction times (in seconds) computed over 120 reconstruction problems with $N = 1024$ and $N_a = 360$ projection angles. These reconstructions are computed using one Nvidia GeForce GTX 1080Ti GPU (11 GB).

FDK	SIRT^+_{200}	NN-FDK_4	U-net	MSD-net
28 ± 8	3225 ± 916	76 ± 8	382 ± 69	809 ± 86

Reconstruction times.

5.2. Reconstruction Accuracy for Simulated Data

For evaluating the reconstruction accuracy using simulated data, we consider 16 cases: 6 different noise levels, 5 different numbers of projection angles and 5 different cone angles. For each case, an NN-FDK, MSD-net and U-net network was trained. For the training process of NN-FDK, we used $N_\text{T} = 10^6$ training voxels and $N_\text{V} = 10^6$ validation voxels from $N_\text{TD} = 10$ and $N_\text{VD} = 5$ datasets, respectively. For U-net and MSD-net, we took the same datasets for training and validation (10 for training and 5 for validation), and used all voxels in these datasets for the training process. The NN-FDK networks were trained till convergence and the DNNs were trained for 48 h. Note that in a few cases, we had to retrain the DNNs because of inconsistent results (i.e., cases with more information achieving a lower reconstruction accuracy), possibly because they got stuck in local minima of the loss function.

In Figure 6, we show the average and standard deviation of the TSE and the SSIM for the considered cases. The subscript HN for the FDK algorithm indicates that the *Hann* filter was used to compute the reconstruction. We observe that U-net and MSD-net achieve the most accurate results and that NN-FDK and SIRT$^+$ closely follow. The FDK algorithm is lowest in all categories. Between NN-FDK and SIRT$^+$, we see that NN-FDK performs best for the noisy reconstruction problems and SIRT$^+$ achieves better results for the reconstruction problems without noise. We visualize the noise for the lowest and highest I_0 in Figure 7 by showing a line profile through the center of the $z = 0$ slice. Here, we see that for the noisiest problems, the amplitude of the noise can be as high as the maximum value of the phantom. In Figure 8, we show 2D slices of reconstructions of the test phantoms for the three types of reconstruction problems. In all cases, we still observe reconstruction artifacts, but comparing these to the baseline FDK reconstructions, the majority is removed or suppressed.

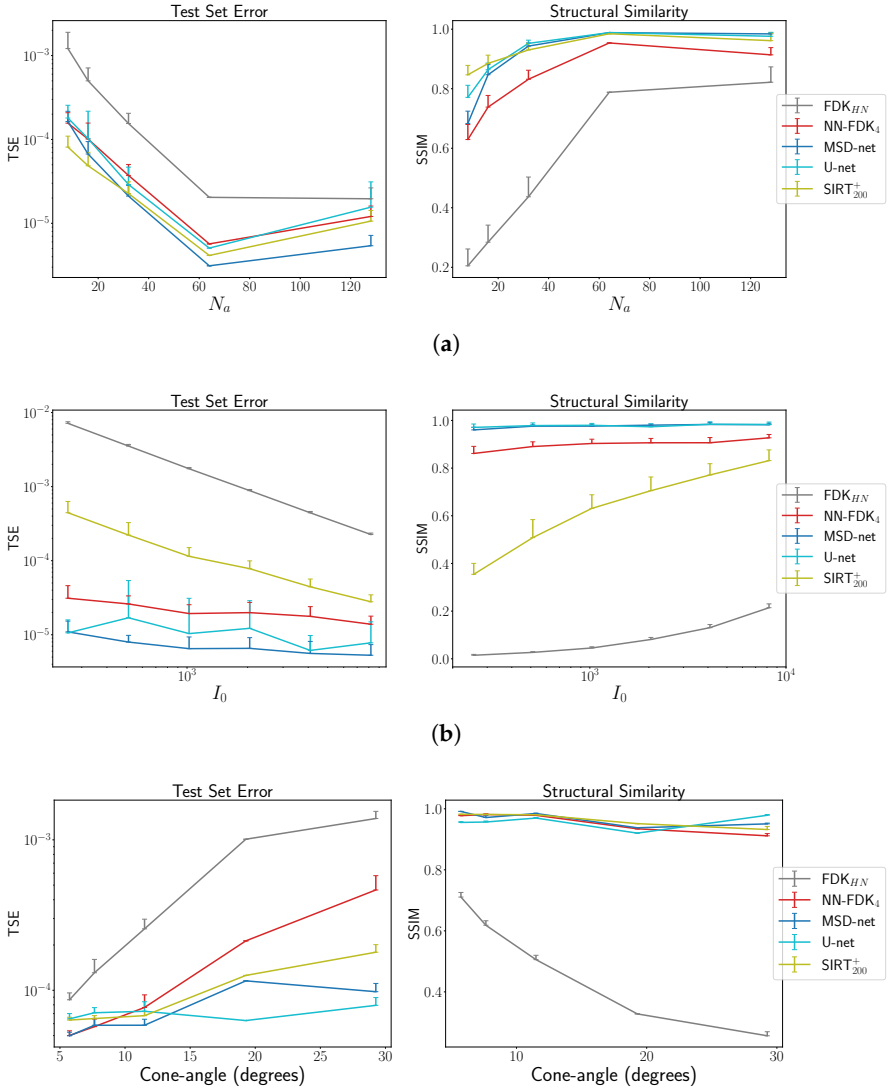

Figure 6. The average and standard deviation of the TSE andstructural similarity index (SSIM). These results are discussed in Section 5.2. For each number of projection angles, the noise level, cone angle and training scenario of one specific network are trained and used to evaluate the 20 reconstruction problems. The NN-FDK reconstruction time is 4-10 times lower than U-net, MSD-net and approximately 40 times lower than SIRT$^+_{200}$. (**a**) The average and standard deviation of the TSE and SSIM as a function of number of projection angles N_a computed over 20 randomly generated phantoms Fourshape family. (**b**) The average and standard deviation of the TSE and SSIM as a function of the emitted photon count I_0 computed over 20 randomly generated phantoms of the Fourshape family. (**c**) The average and standard deviation of the average TSE and SSIM as a function of the cone angle computed over 20 randomly generated phantoms of the Defrise family.

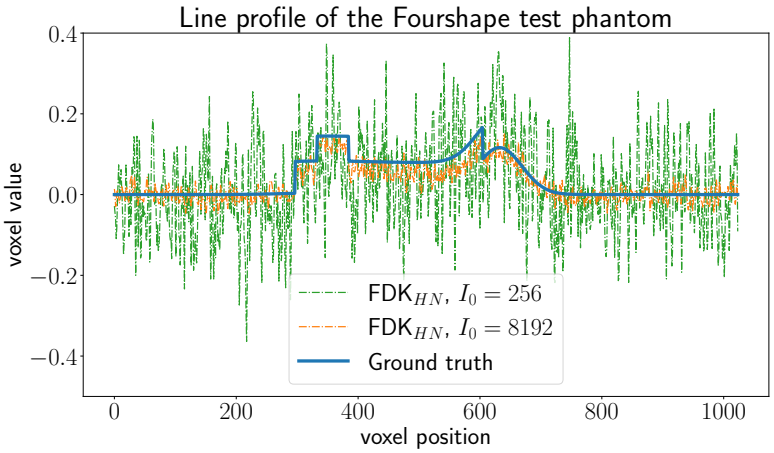

Figure 7. Line profile through the center of the $z = 0$ slice of the Fourshape test phantom. We show the ground truth profile, the profile of the FDK reconstruction with lowest emitted photon count $I_0 = 256$, and the profile of the FDK reconstruction with the highest emitted photon count $I_0 = 8196$.

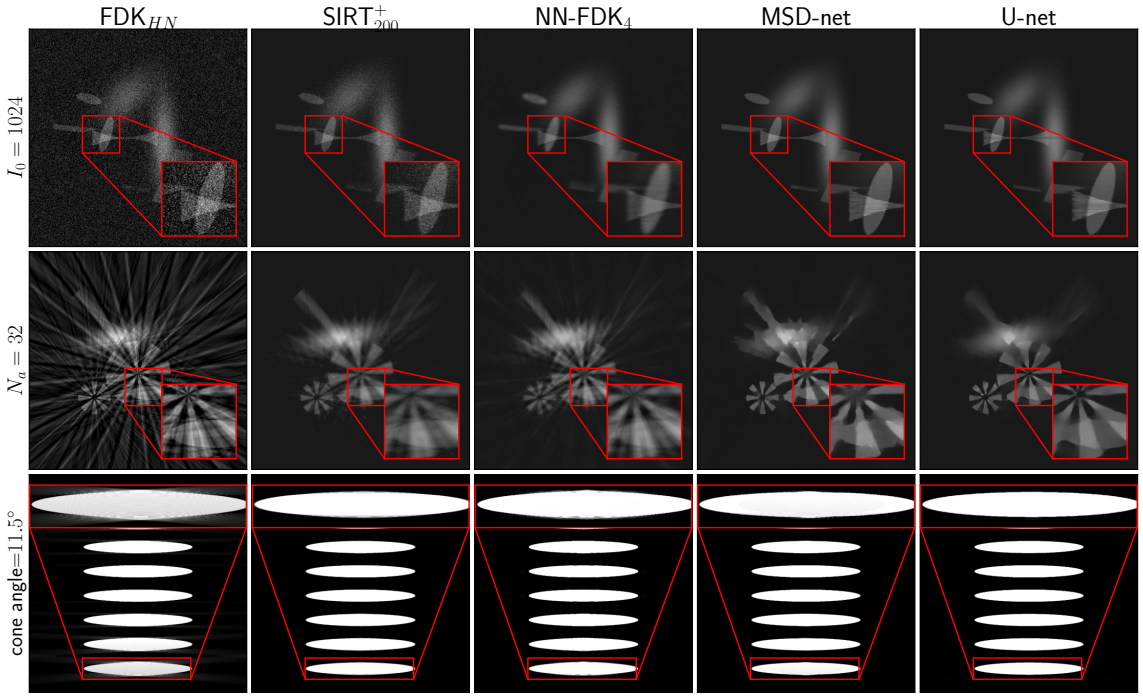

Figure 8. Two-dimensional slices of the reconstructions for the considered reconstruction methods. (Top) Slice $x = 0$ of the Fourshape test phantom reconstruction problem with $N_a = 360$ projection angles and $I_0 = 1024$ emitted photon count. (Middle) Slice $z = 0$ of the Fourshape test phantom reconstruction problem with $N_a = 32$ projection angles. (Bottom) Slice $x = 0$ of the Defrise reconstruction problem with $N_a = 360$ projection angles and a cone angle of 11.5 degrees.

5.3. Reconstruction Accuracy for Experimental Data

In this section, we use the datasets discussed in Section 4.2 to assess the reconstruction accuracy of experimental data. In a similar fashion as for the simulated data, we trained a network for the low-dose reconstruction problem and a network for the high-dose reconstruction problem with $N_a = 32$ projection angles with the notable exception that U-net and MSD-net were trained till convergence. The results are presented in Table 4.

Comparing the results to the simulated data experiments, we see that SIRT^+ performs worse on the experimental data, even with the additional regularization of early stopping. This is most likely due to the high-dose datasets still containing noise, whereas this was completely absent in the simulated data experiments. These differences are illustrated in Figure 9, where 2D slices of the reconstructions for the high-dose reconstruction problem with $N_a = 32$ projection angles are shown.

Table 4. Average and standard deviation of the quantitative measures computed over 6 walnut datasets. The high-dose low projection angle reconstruction problem has $N_a = 32$ projection angles, the low-dose reconstruction problem has $N_a = 500$ projection angles. For the high-dose data, we used 200 iterations of SIRT, and for the low-dose data, we used 20 iterations of SIRT. The NN-FDK reconstruction time is 4-10 times lower than U-net, MSD-net and SIRT^+_{20}, and approximately 40 times lower than SIRT^+_{200}.

	High-Dose, Low Number of Projection Angles		**Low-Dose**	
Method	**TSE**	**SSIM**	**TSE**	**SSIM**
FDK_{HN}	$5.54 \pm 3.43 \times 10^{-3}$	0.224 ± 0.076	$1.40 \pm 0.05 \times 10^{-3}$	0.334 ± 0.104
$\text{SIRT}^+_{200/20}$	$9.94 \pm 0.15 \times 10^{-4}$	0.603 ± 0.087	$1.92 \pm 0.08 \times 10^{-3}$	0.584 ± 0.083
NN-FDK_4	$8.03 \pm 1.39 \times 10^{-4}$	0.946 ± 0.010	$1.14 \pm 0.23 \times 10^{-4}$	0.965 ± 0.012
U-net	$4.10 \pm 1.06 \times 10^{-4}$	0.964 ± 0.009	$1.02 \pm 0.45 \times 10^{-4}$	0.980 ± 0.006
MSD-net	$4.23 \pm 0.97 \times 10^{-4}$	0.964 ± 0.009	$7.82 \pm 2.86 \times 10^{-5}$	0.980 ± 0.007

Experimental data.

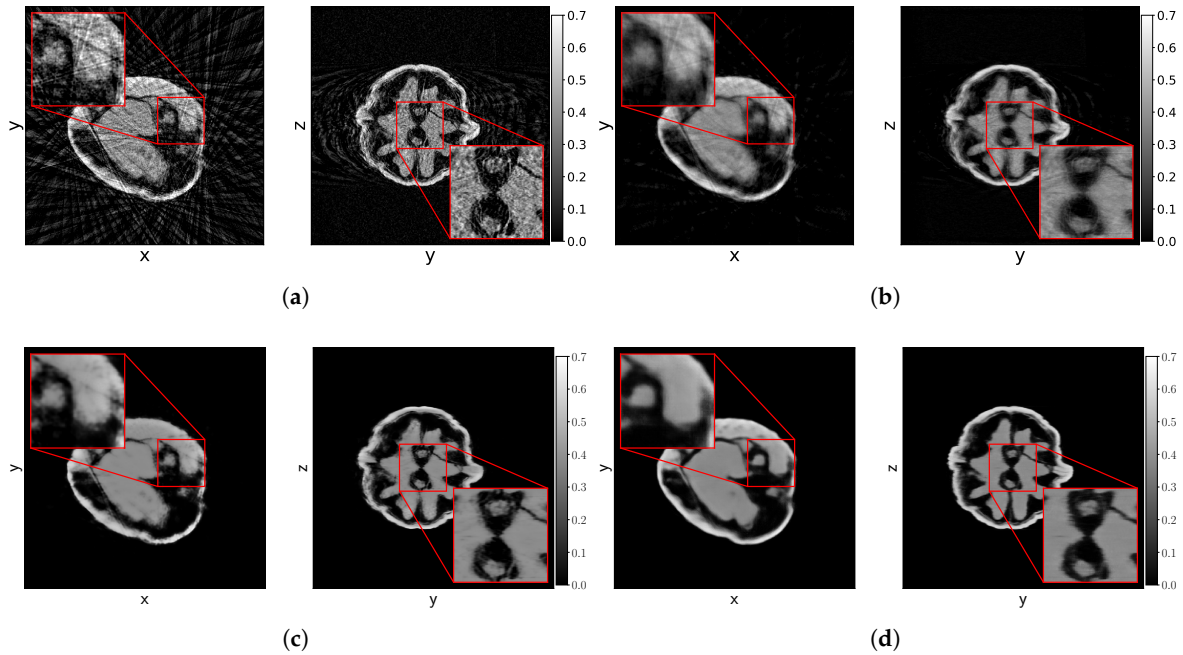

Figure 9. Slices $z = 0$ and $x = 0$ of several reconstruction methods of the high-dose dataset of the 21 st walnut with 32 projection angles. (**a**) FDK_{HN}. (**b**) SIRT^+_{200} reconstruction. (**c**) NN-FDK_4 reconstruction. (**d**) MSD-net reconstruction.

5.4. Segmentation Experiment for Experimental Data

To assess the performance of the different reconstruction approaches in a segmentation task, we focus here on the segmentation of the shell and kernel of walnuts, based on our experimental CT data. The review [59] provides an overview of segmentation problems in walnut imaging, and their relevance. For segmenting the 3D volume after the reconstruction, we used a deterministic segmentation algorithm that combines thresholding, the watershed algorithm and prior knowledge

of the scanned objects. Details of this method are discussed in Appendix A.4. For the reference segmentation, we apply this algorithm to the gold standard reconstruction.

For determining the accuracy of the segmentation of an object—i.e., shell, empty space and kernel of the walnut—we consider three metrics: volume error, mislabeled voxels and the Dice coefficient [60]. We define a segmentation S as a reconstruction volume with value 1 if the voxel is in the object (shell, kernel or empty space) and 0 if outside the object. Furthermore, we define the norm of a segmentation as the sum: $|S| = \sum_i^{N^3} (S)_i$. Using this notation, we can compute the measures in the following manner:

$$V_{\text{err}} = \frac{|S_{\text{rec}}| - |S_{\text{GS}}|}{|S_{\text{GS}}|}, \qquad \text{ML}_{\text{err}} = \frac{|S_{\text{rec}} - S_{\text{GS}}|}{|S_{\text{GS}}|}, \qquad \text{DC} = \frac{2|S_{\text{rec}} \cap S_{\text{GS}}|}{|S_{\text{rec}}| + |S_{\text{GS}}|}, \tag{20}$$

with GS denoting the gold standard reconstruction.

In Table 5, we show the results for computing these metrics on the 6 walnuts not considered in the training process. We observe that MSD-net performs best in segmenting the shell and U-net performs best at segmenting the empty space and kernel and NN-FDK is close to both DNNs and in some cases even better than MSD-net for segmenting the empty space and kernel. Comparing NN-FDK to standard FDK, we observe a significant improvement.

Table 5. The average and standard deviation of the three metrics computed over the 6 low-dose walnut datasets with $N_a = 500$ projection angles. The metrics are computed using (20). The NN-FDK reconstruction time is 4-10 times lower than U-net, MSD-net and approximately 40 times lower than SIRT^+_{200}.

Method	Shell	Empty Space	Kernel
	Volume errors		
FDK_{HN}	0.127 ± 0.078	0.146 ± 0.091	0.128 ± 0.092
SIRT^+_{200}	0.082 ± 0.047	0.104 ± 0.078	0.050 ± 0.074
NN-FDK_4	0.068 ± 0.035	0.045 ± 0.035	0.029 ± 0.032
U-net	0.055 ± 0.019	0.029 ± 0.017	0.012 ± 0.016
MSD-net	0.028 ± 0.010	0.059 ± 0.075	0.035 ± 0.050
	Mislabeled voxels		
FDK_{HN}	0.168 ± 0.087	0.190 ± 0.98	0.144 ± 0.081
SIRT^+_{200}	0.133 ± 0.026	0.182 ± 0.118	0.101 ± 0.048
NN-FDK_4	0.103 ± 0.026	0.087 ± 0.023	0.072 ± 0.018
U-net	0.092 ± 0.028	0.073 ± 0.024	0.059 ± 0.019
MSD-net	0.086 ± 0.038	0.116 ± 0.094	0.061 ± 0.039
	Dice coefficient		
FDK_{HN}	0.922 ± 0.036	0.895 ± 0.061	0.934 ± 0.033
SIRT^+_{200}	0.934 ± 0.016	0.908 ± 0.061	0.947 ± 0.028
NN-FDK_4	0.951 ± 0.012	0.955 ± 0.013	0.964 ± 0.008
U-net	0.955 ± 0.013	0.963 ± 0.012	0.971 ± 0.010
MSD-net	0.957 ± 0.018	0.939 ± 0.055	0.971 ± 0.018

Segmentation errors.

5.5. Data Requirements

To test the influence of the amount of training data on the reconstruction quality, we performed an experiment with three different training scenarios:

- **Scenario 1**. One dataset available. Here, we take the training and validation data from the same dataset.
- **Scenario 2**. Two datasets available. Here, we take the training and validation data from the separate datasets.

- **Scenario 3**. Fifteen datasets available. Again, the training and validation data are picked from separate datasets, but now the training and validation pairs come from several datasets, specifically 10 training datasets ($N_{\mathrm{TD}} = 10$) and 5 validation datasets ($N_{\mathrm{VD}} = 5$). This is the scenario used in the previous experiments.

We fix the number of voxels used for training and validation at $N_{\mathrm{T}} = 10^6$ and $N_{\mathrm{V}} = 10^6$ for all scenarios. For comparison, we trained a U-net and a MSD-net network with the same training scenarios, with the exception that all voxels from the datasets are used. For training scenario 1, the slices are divided into a training and a validation set. More specifically, every fourth slice is used for validation.

We performed this experiment for two simulated data problems, a high noise level (emitted photon count $I_0 = 256$) and a large cone angle (29.3 degrees), and the two experimental data problems. For the sake of brevity, we show only the results for the high-noise simulated data reconstruction problem (Table 6) and the high-noise experimental data reconstruction problem (Table 7). The results for the other reconstruction problems are given in Appendix C. Comparing quantitative measures between the different scenarios, we see that the reconstruction accuracy improves as more data is used for the simulated data experiment, but remains about the same for the experimental data experiment. This can be explained by the variation in the objects used in the reconstruction problems. Recall that the Fourshape phantom family has a large variety in its phantoms, i.e., three instances of four randomly generated objects, and the variety within the walnut datasets is small, i.e., similar shapes, sizes and structures. This indicates that if objects are similar, one training dataset may already be sufficient to train networks that achieve a high reconstruction accuracy.

Note that although the training scenarios for NN-FDK and the DNNs use the same number of datasets, the number of voxels considered for training the NN-FDK network is constant over all three scenarios and is several orders of magnitude lower than the number of voxels considered for training the DNNs. This opens up future possibilities for reducing the training data requirements to only need a high-quality reconstruction of a certain region of interest.

Table 6. Average and standard deviation of the quantitative measures computed over 20 Fourshape phantoms for varying training scenarios. The reconstruction problems have an emitted photon count of $I_0 = 256$ and $N_a = 360$ projection angles. The NN-FDK reconstruction time is 4-10 times lower than U-net and MSD-net.

	TSE		
Method	**1 Dataset**	**2 Datasets**	**15 Datasets**
NN-FDK$_4$	$4.97 \pm 4.68 \times 10^{-5}$	$4.19 \pm 3.60 \times 10^{-5}$	$2.51 \pm 1.14 \times 10^{-5}$
U-net	$1.06 \pm 1.36 \times 10^{-5}$	$2.45 \pm 2.87 \times 10^{-5}$	$8.06 \pm 3.63 \times 10^{-6}$
MSD-net	$1.12 \pm 0.41 \times 10^{-5}$	$1.12 \pm 0.40 \times 10^{-5}$	$7.94 \pm 3.16 \times 10^{-6}$
	SSIM		
NN-FDK$_4$	0.831 ± 0.065	0.844 ± 0.065	0.884 ± 0.030
U-net	0.884 ± 0.075	0.932 ± 0.050	0.979 ± 0.009
MSD-net	0.961 ± 0.013	0.962 ± 0.013	0.974 ± 0.008

Simulated data, high noise.

Table 7. Average and standard deviation of the quantitative measures computed over 6 walnuts for various training scenarios. The datasets are low-dose and have $N_a = 500$ projection angles. The NN-FDK reconstruction time is 4-10 times lower than U-net and MSD-net.

		TSE	
Method	**1 Dataset**	**2 Datasets**	**15 Datasets**
NN-FDK$_4$	$1.16 \pm 0.25 \times 10^{-4}$	$1.23 \pm 0.25 \times 10^{-4}$	$1.14 \pm 0.23 \times 10^{-4}$
U-net	$1.27 \pm 0.38 \times 10^{-4}$	$1.23 \pm 0.35 \times 10^{-4}$	$1.02 \pm 0.45 \times 10^{-4}$
MSD-net	$1.28 \pm 0.41 \times 10^{-4}$	$1.16 \pm 0.35 \times 10^{-4}$	$7.82 \pm 2.86 \times 10^{-5}$
		SSIM	
NN-FDK$_4$	0.973 ± 0.009	0.968 ± 0.011	0.965 ± 0.012
U-net	0.979 ± 0.008	0.978 ± 0.008	0.980 ± 0.006
MSD-net	0.979 ± 0.008	0.979 ± 0.008	0.980 ± 0.007

Experimental data, low-dose.

6. Summary and Conclusions

We have proposed the Neural Network FDK (NN-FDK) algorithm, a reconstruction algorithm for the circular cone-beam (CCB) Computed Tomography (CT) geometry with a machine learning component. The machine learning component of the algorithm is designed to learn a set of FDK filters and to combine the FDK reconstructions done with these filters. This leads to a computationally efficient reconstruction algorithm, since one only needs to compute and combine the FDK reconstructions for this learned set of filters. Due to parametrization of the learned filters, the NN-FDK network has a low number of trainable parameters ($<$100) and can be trained efficiently with the Levenberg–Marquardt algorithm with approximate quadratic convergence rate.

We compared the NN-FDK algorithm to SIRT with a nonnegativity constraint (SIRT$^+$), the standard FDK algorithm and two deep neural networks (DNNs), namely a 2D U-net and a 2D MSD-net applied in a slice-by-slice fashion to a 3D volume. We have shown that the NN-FDK algorithm has the lowest reconstruction time after the standard FDK algorithm. We have also shown that the NN-FDK algorithm achieves a reconstruction accuracy that is similar to that of SIRT$^+$ for simulated data and a higher accuracy than that of SIRT$^+$ for experimental data. The DNNs achieved the highest reconstruction accuracy, but training those networks took between 2 days (1 training and validation dataset) and 2 weeks (15 training and validation datasets), whereas all the NN-FDK networks were trained within 1 minute.

To conclude, the NN-FDK algorithm is a computationally efficient reconstruction algorithm that can reconstruct CCB CT reconstruction problems with high-noise, low projection angles or large cone angles accurately. The training process is efficient and requires a low amount of training data, making it suitable for application to a broad spectrum of large scale (up to $4096 \times 4096 \times 4096$) reconstruction problems. Specifically, the NN-FDK algorithm can be used to improve image quality in high-throughput CT scanning settings, where FDK is currently used to keep pace with the acquisition speed using readily available computational resources.

Author Contributions: Conceptualization, M.J.L., D.M.P., W.J.P. and K.J.B.; methodology, M.J.L. and D.M.P.; software, M.J.L. and W.J.P.; validation, M.J.L.; formal analysis, M.J.L.; investigation, M.J.L.; data curation, M.J.L.; writing—original draft preparation, M.J.L.; writing—review and editing, M.J.L., D.M.P., W.J.P., and K.J.B.; visualization, M.J.L.; supervision, D.M.P., W.J.P., and K.J.B.; funding acquisition, D.M.P. and K.J.B. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge financial support from the Netherlands Organisation for Scientific Research (NWO), project numbers 639.073.506 and 016.Veni.192.235.

Acknowledgments: The authors acknowledge XRE NV for their role in the FleX-ray collaboration. We thank Sophia Bethany Coban for her support in acquiring the experimental data.

Conflicts of Interest: The authors declare no conflict of interest.The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CCB	Circular cone-beam
CT	Computed Tomography
FDK	Feldkamp–Davis–Kress
NN-FDK	Neural Network Feldkamp–Davis–Kress
FBP	Filtered backprojection
DNN	Deep neural network
SIRT	Simultaneous Iterative Reconstruction Technique
MSD-net	Mixed-scale dense network
U-net	U-network
LMA	Levenberg–Marquardt algorithm

Appendix A. Implementation

Appendix A.1. Data Generation

For our simulated data experiments, we take $N = 1024$, which means that reconstructions and reference images are defined on a 1024^3 equidistant voxel grid, and the projection data on a 1024^2 equidistant detector grid per projection angle. However, to avoid using the same operator for reconstructions as for the data generation, we generate the input data at a higher resolution. More specifically, we generate a phantom at $N = 1536$, forward project this phantom to the data space with size $N_a \times 1536^2$ and apply a bilinear interpolation per projection angle to arrive at a 1024^2 detector grid, resulting in input data with the desired resolution $N_a \times 1024^2$. We set the source radius to 10 times the physical size of the phantom, resulting in a cone angle of 5.7 degrees. To generate noise, we compute a noise-free photon count I from clean projection data $\mathbf{y}_c$ and use that to generate a Poisson distributed photon count from which we compute $\mathbf{y}$:

$$I = I_0 e^{-\mathbf{y}_c}, \qquad I_{\text{noise}} \sim \text{Pois}(I), \qquad \mathbf{y} = -\log\left(\frac{I_{\text{noise}}}{I_0}\right), \tag{A1}$$

with I_0 the emitted photon count. Higher I_0 implies a higher dose and, therefore, less noise in the data.

Appendix A.2. Deep Neural Networks

Application strategy: We train 2D DNNs to remove artifacts from 2D slices of an FDK reconstruction. We train one network that handles all slices in the reconstructions.

Training DNNs: We train the DNNs with ADAM [52] and stop training after 48 h of training on an Nvidia GeForce GTX 1080Ti GPU, the network with the lowest validation set error during this training process will be used for the reconstructions.

U-net and MSD-net structures: For U-net, we will take four up and down layers with 3×3 convolutions, 2×2 max-pooling and 2×2 up-convolutions as presented in [31]. For the MSD-nets, we take 100 layers with one input and one output layer and the dilations, as suggested in [28].

Appendix A.3. Code-Base

We implemented the NN-FDK framework using Python 3.6.5 and Numpy 1.14.5 [61]. For the parameter learning, we used the Levenberg–Marquardt algorithm implementation from [33]. The reconstruction algorithm is implemented using ODL [62], the ASTRA-toolbox [63], PyFFTW [64]

and the exponential binning framework for filters from [23]. For performance reasons, the simulated phantoms are generated through C++ using Cython [65].

For the evaluation of U-nets, we took the PyTorch [66] implementation used in [67]. The MSD-nets are implemented using the package published with [26].

All the code related to this paper can be found on Github [68].

Appendix A.4. Segmentation Algorithm

This algorithm consists of several steps:

1. Apply a Gaussian filter to the reconstruction.
2. Compute a histogram of the filtered reconstruction and determine the peaks relating to the background, kernel and shell.
3. Determine the shell and kernel segmentations using a threshold based on the found peaks.
4. Apply the watershed algorithm on the shell segmentation. This gives the total volume inside the walnut.
5. Remove the kernel from the total volume inside the walnut to attain the empty space segmentation.

Further details about this implementation can be found on our Github [68].

Appendix B. Levenberg–Marquardt Algorithm

Given the learning problem (12), the update rule for the Levenberg–Marquardt algorithm (LMA) ([50,51]) is given by:

$$\theta^{i+1} = \theta^i + \mathbf{t}^i, \tag{A2}$$

with $\mathbf{t}^i$ the update vector. This is computed by solving the following equation for $\mathbf{t}^i$

$$\left(J_i^T J_i + \lambda_i I\right) \mathbf{t}^i = -\frac{\partial \mathcal{L}}{\partial \theta}(\theta^i, T) = -J_i^T \sum_{j=1}^{N_T} \left(O_j - \mathsf{N}_\theta(Z_j)\right) \tag{A3}$$

where $\lambda_i > 0$ is the step parameter and J_i the $m \times n$ Jacobian matrix of $\mathsf{N}_{\theta^i}(\mathbf{Z})$ with respect to θ^i, with $\mathbf{Z}$ the vector containing all inputs from the training set T. We can solve (A3) using a Cholesky decomposition. ($J_i^T J_i$ is positive semi-definite and $\lambda_i > 0$; therefore, the left-hand side of (A3) is positive definite.)

To ensure convergence, only updates that improve the training error are accepted, i.e., if the following is true:

$$\mathcal{L}(\theta^i, T) > \mathcal{L}(\theta^i + \mathbf{t}^i, T), \tag{A4}$$

If this is not the case, we change the step parameter λ_i to $a\lambda_i$ with $a > 1$ and compute a new update vector $\mathbf{t}^i$. When an update is accepted, we change the step parameter to $\lambda_{i+1} = \lambda_i / a$.

We use two stopping criteria for the LMA. Firstly, we stop if we cannot find a suitable θ^{i+1}, using several indicators for this:

- The norm of the gradient $\frac{\partial \mathcal{L}}{\partial \theta}(\theta^i)$ is too small
- The step size λ_i is too big
- After N_{up} rejected updates.

The second stopping criterion checks whether the parameters θ^i improve the validation set error. More specifically, we terminate the LMA when the validation set error has not improved for N_{val} iterations.

In Algorithm A1, the LMA is summarized. The random initialization is done with the Nguyen–Widrow initialization method [69]. For our experiments, we take $N_{\text{up}} = 100$, $\lambda_0 = 10^5$, $a = 10$ and $N_{\text{val}} = 100$.

Algorithm A1 Levenberg–Marquardt algorithm

1: Compute random initialization θ^0 using [69]
2: **repeat**
3: Compute $\mathbf{t}^i$ until we accept an update θ^{i+1}.
4: **until** N_{up} updates were rejected **or**
$\mathcal{L}(\theta^i, V)$ did not improve N_{val} times **or**
$\|\frac{\partial \mathcal{L}}{\partial \theta}(\theta^{i+1})\|$ is too small **or** λ_{i+1} is too big.
5: Set $\theta^\star$ equal to the θ^i with the lowest validation error.

Appendix C. Results Data Requirement Experiment

Results for simulated data with a large cone angle are shown in Table A1. Results for experimental data with high-dose adn 32 projection angles are shown in Table A2.

Table A1. Average and standard deviation of the quantitative measures computed over 20 different Defrise phantoms for various training scenarios. The reconstruction problems have a cone angle of 29.2 degrees and $N_a = 360$ projection angles.

		TSE	
Method	**1 Dataset**	**2 Datasets**	**15 Datasets**
NN-FDK$_4$	$6.47 \pm 1.19 \times 10^{-4}$	$4.70 \pm 1.16 \times 10^{-4}$	$4.82 \pm 1.13 \times 10^{-4}$
U-net	$1.04 \pm 0.27 \times 10^{-4}$	$1.02 \pm 0.17 \times 10^{-4}$	$8.23 \pm 0.85 \times 10^{-5}$
MSD-net	$2.44 \pm 1.43 \times 10^{-4}$	$1.53 \pm 0.17 \times 10^{-4}4$	$6.52 \pm 0.43 \times 10^{-5}$
		SSIM	
NN-FDK$_4$	0.825 ± 0.018	0.904 ± 0.011	0.910 ± 0.007
U-net	0.974 ± 0.015	0.971 ± 0.021	0.973 ± 0.010
MSD-net	0.954 ± 0.006	0.937 ± 0.004	0.966 ± 0.002

Simulated data, large cone angle.

Table A2. Average and standard deviation of the quantitative measures computed over the 6 datasets for various training scenarios. These are the high-dose datasets from [56] with $N_a = 32$ projection angles. The best results are highlighted.

		TSE	
Method	**1 Dataset**	**2 Datasets**	**15 Datasets**
NN-FDK$_4$	$8.14 \pm 1.45 \times 10^{-4}$	$8.68 \pm 1.43 \times 10^{-4}$	$8.03 \pm 1.39 \times 10^{-4}$
U-net	$7.56 \pm 1.52 \times 10^{-4}$	$6.85 \pm 1.56 \times 10^{-4}$	$\mathbf{4.10 \pm 1.06 \times 10^{-4}}$
MSD-net	$7.82 \pm 0.41 \times 10^{-4}$	$6.51 \pm 0.35 \times 10^{-4}$	$4.23 \pm 0.97 \times 10^{-4}$
		SSIM	
NN-FDK$_4$	0.950 ± 0.010	0.948 ± 0.010	0.946 ± 0.011
U-net	0.955 ± 0.011	0.930 ± 0.023	$\mathbf{0.964 \pm 0.009}$
MSD-net	0.955 ± 0.010	0.947 ± 0.014	$\mathbf{0.964 \pm 0.009}$

Experimental data, high-dose, 32 projection angles.

References

1. Giudiceandrea, F.; Ursella, E.; Vicario, E. A high speed CT scanner for the sawmill industry. In Proceedings of the 17th International Non Destructive Testing and Evaluation of Wood Symposium, Sopron, Hungary, 14–16 September 2011.
2. Dierick, M.; Van Loo, D.; Masschaele, B.; Van den Bulcke, J.; Van Acker, J.; Cnudde, V.; Van Hoorebeke, L. Recent micro-CT scanner developments at UGCT. *Nucl. Instrum. Methods Phys. Res. Sect. B Beam Interact. Mater. Atoms* **2014**, *324*, 35–40. [CrossRef]
3. Bultreys, T.; Boone, M.A.; Boone, M.N.; De Schryver, T.; Masschaele, B.; Van Hoorebeke, L.; Cnudde, V. Fast laboratory-based micro-computed tomography for pore-scale research: Illustrative experiments and perspectives on the future. *Adv. Water Resour.* **2016**, *95*, 341–351. [CrossRef]
4. Ford, E.; Chang, J.; Mueller, K.; Sidhu, K.; Todor, D.; Mageras, G.; Yorke, E.; Ling, C.; Amols, H. Cone-beam CT with megavoltage beams and an amorphous silicon electronic portal imaging device: Potential for verification of radiotherapy of lung cancer. *Med. Phys.* **2002**, *29*, 2913–2924. [CrossRef]
5. Galicia, J.C.; Kawilarang, J.; Tawil, P.Z. Clinical Endodontic Applications of Cone Beam-Computed Tomography in Modern Dental Practice. *Open J. Stomatol.* **2017**, *7*, 314. [CrossRef]
6. TESCAN. TESCAN UniTOM XL, Modular and Versatile High Resolution 3D X-ray Imaging. Available online: https://www.tescan.com/product/micro-ct-for-materials-science-tescan-unitom-xl/ (accessed on 17 November 2020).
7. TESCAN. TESCAN DynaTOM, High Temporal Resolution 4D X-ray Imaging. Available online: https://www.tescan.com/product/micro-ct-for-materials-science-tescan-dynatom/ (accessed on 17 November 2020).
8. Canon Medical Systems USA, Inc. Aquilon™ Precision, ULTRA High Resolution CT. Available online: https://us.medical.canon/products/computed-tomography/aquilion-precision/ (accessed on 17 November 2020).
9. Natterer, F. *The Mathematics of Computerized Tomography*; SIAM: Philadelphia, PA, USA, 2001. [CrossRef]
10. Feldkamp, L.; Davis, L.; Kress, J. Practical cone-beam algorithm. *JOSA A* **1984**, *1*, 612–619. [CrossRef]
11. Katsevich, A. A general scheme for constructing inversion algorithms for cone beam CT. *Int. J. Math. Math. Sci.* **2003**, *2003*, 1305–1321. [CrossRef]
12. Pan, X.; Sidky, E.Y.; Vannier, M. Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction? *Inverse Probl.* **2009**, *25*, 123009. [CrossRef]
13. Rudin, L.I.; Osher, S.; Fatemi, E. Nonlinear total variation based noise removal algorithms. *Phys. D Nonlinear Phenom.* **1992**, *60*, 259–268. [CrossRef]
14. Bredies, K.; Kunisch, K.; Pock, T. Total generalized variation. *SIAM J. Imaging Sci.* **2010**, *3*, 492–526. [CrossRef]
15. Sidky, E.Y.; Pan, X. Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization. *Phys. Med. Biol.* **2008**, *53*, 4777. [CrossRef]
16. Jia, X.; Lou, Y.; Li, R.; Song, W.Y.; Jiang, S.B. GPU-based fast cone beam CT reconstruction from undersampled and noisy projection data via total variation. *Med. Phys.* **2010**, *37*, 1757–1760. [CrossRef] [PubMed]
17. Niu, S.; Gao, Y.; Bian, Z.; Huang, J.; Chen, W.; Yu, G.; Liang, Z.; Ma, J. Sparse-view X-ray CT reconstruction via total generalized variation regularization. *Phys. Med. Biol.* **2014**, *59*, 2997. [CrossRef] [PubMed]
18. Elbakri, I.A.; Fessler, J.A. Efficient and accurate likelihood for iterative image reconstruction in X-ray computed tomography. In Proceedings of the Medical Imaging 2003: Image Processing, International Society for Optics and Photonics, San Diego, CA, USA, 17–20 February 2003; Volume 5032, pp. 1839–1850.
19. Zeng, G.L. A filtered backprojection algorithm with characteristics of the iterative Landweber algorithm. *Med. Phys.* **2012**, *39*, 603–607. [CrossRef] [PubMed]
20. Nielsen, T.; Hitziger, S.; Grass, M.; Iske, A. Filter calculation for X-ray tomosynthesis reconstruction. *Phys. Med. Biol.* **2012**, *57*, 3915. [CrossRef] [PubMed]
21. Batenburg, K.J.; Plantagie, L. Fast approximation of algebraic reconstruction methods for tomography. *IEEE Trans. Image Process.* **2012**, *21*, 3648–3658. [CrossRef] [PubMed]
22. Pelt, D.M.; Batenburg, K.J. Improving filtered backprojection reconstruction by data-dependent filtering. *IEEE Trans. Image Process.* **2014**, *23*, 4750–4762. [CrossRef] [PubMed]
23. Lagerwerf, M.J.; Palenstijn, W.J.; Kohr, H.; Batenburg, K.J. Automated FDK-filter selection for Cone-beam CT in research environments. *IEEE Trans. Comput. Imaging* **2020**. [CrossRef]

24. Kunze, H.; Haerer, W.; Orman, J.; Mertelmeier, T.; Stierstorfer, K. Filter determination for tomosynthesis aided by iterative reconstruction techniques. In Proceedings of the 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine, Lindau, Germany, 9–13 July 2007; pp. 309–312.
25. Jin, K.H.; McCann, M.T.; Froustey, E.; Unser, M. Deep convolutional neural network for inverse problems in imaging. *IEEE Trans. Image Process.* **2017**, *26*, 4509–4522. [CrossRef]
26. Pelt, D.M.; Batenburg, K.J.; Sethian, J. Improving tomographic reconstruction from limited data using mixed-scale dense convolutional neural networks. *J. Imaging* **2018**, *4*, 128. [CrossRef]
27. Kida, S.; Nakamoto, T.; Nakano, M.; Nawa, K.; Haga, A.; Kotoku, J.; Yamashita, H.; Nakagawa, K. Cone beam computed tomography image quality improvement using a deep convolutional neural network. *Cureus* **2018**, *10*, e2548. [CrossRef] [PubMed]
28. Pelt, D.M.; Sethian, J.A. A mixed-scale dense convolutional neural network for image analysis. *Proc. Natl. Acad. Sci. USA* **2018**, *115*, 254–259. [CrossRef] [PubMed]
29. Wang, G.; Ye, J.C.; Mueller, K.; Fessler, J.A. Image reconstruction is a new frontier of machine learning. *IEEE Trans. Med. Imaging* **2018**, *37*, 1289–1296. [CrossRef] [PubMed]
30. Çiçek, Ö.; Abdulkadir, A.; Lienkamp, S.S.; Brox, T.; Ronneberger, O. 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Athens, Greece, 17–21 October 2016; pp. 424–432.
31. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.
32. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer Science+ Business Media: Berlin/Heidelberg, Germany, 2006.
33. Pelt, D.M.; Batenburg, K.J. Fast tomographic reconstruction from limited data using artificial neural networks. *IEEE Trans. Image Process.* **2013**, *22*, 5238–5251. [CrossRef] [PubMed]
34. Van der Sluis, A.; van der Vorst, H.A. SIRT-and CG-type methods for the iterative solution of sparse linear least-squares problems. *Linear Algebra Appl.* **1990**, *130*, 257–303. [CrossRef]
35. Kang, E.; Min, J.; Ye, J.C. A Deep Convolutional Neural Network Using Directional Wavelets for Low-Dose X-Ray CT Reconstruction. *Med. Phys.* **2017**, *44*, e360–e375. [CrossRef] [PubMed]
36. Adler, J.; Öktem, O. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Probl.* **2017**, *33*, 124007. [CrossRef]
37. Adler, J.; Öktem, O. Learned primal-dual reconstruction. *IEEE Trans. Med. Imaging* **2018**, *37*, 1322–1332. [CrossRef]
38. Kobler, E.; Klatzer, T.; Hammernik, K.; Pock, T. Variational networks: Connecting variational methods and deep learning. In Proceedings of the German Conference on Pattern Recognition, Stuttgart, Germany, 9–12 October 2017; pp. 281–293.
39. Hammernik, K.; Klatzer, T.; Kobler, E.; Recht, M.P.; Sodickson, D.K.; Pock, T.; Knoll, F. Learning a variational network for reconstruction of accelerated MRI data. *Magn. Reson. Med.* **2018**, *79*, 3055–3071. [CrossRef]
40. Venkatakrishnan, S.V.; Bouman, C.A.; Wohlberg, B. Plug-And-Play Priors for Model Based Reconstruction. In Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing, Austin, TX, USA, 3–5 December 2013. [CrossRef]
41. Romano, Y.; Elad, M.; Milanfar, P. The Little Engine That Could: Regularization By Denoising (RED). *SIAM J. Imaging Sci.* **2017**, *10*, 1804–1844. [CrossRef]
42. Reehorst, E.T.; Schniter, P. Regularization By Denoising: Clarifications and New Interpretations. *arXiv* **2018**, arXiv:cs.CV/1806.02296.
43. Lunz, S.; Öktem, O.; Schönlieb, C.B. Adversarial regularizers in inverse problems. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 8507–8516.
44. Mukherjee, S.; Dittmer, S.; Shumaylov, Z.; Lunz, S.; Öktem, O.; Schönlieb, C.B. Learned convex regularizers for inverse problems. *arXiv* **2020**, arXiv:2008.02839.
45. Shelhamer, E.; Long, J.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [CrossRef] [PubMed]

46. Perone, C.S.; Calabrese, E.; Cohen-Adad, J. Spinal Cord Gray Matter Segmentation Using Deep Dilated Convolutions. *Sci. Rep.* **2018**, *8*. [CrossRef] [PubMed]
47. Zhang, K.; Zuo, W.; Chen, Y.; Meng, D.; Zhang, L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Trans. Image Process.* **2017**, *26*, 3142–3155. [CrossRef]
48. Ye, J.C.; Han, Y.; Cha, E. Deep convolutional framelets: A general deep learning framework for inverse problems. *SIAM J. Imaging Sci.* **2018**, *11*, 991–1048. [CrossRef]
49. Anthony, M.; Bartlett, P.L. *Neural Network Learning: Theoretical Foundations*; Cambridge University Press: Cambridge, UK, 2009.
50. Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Q. Appl. Math.* **1944**, *2*, 164–168. [CrossRef]
51. Marquardt, D.W. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Ind. Appl. Math.* **1963**, *11*, 431–441. [CrossRef]
52. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
53. Kudo, H.; Noo, F.; Defrise, M. Cone-beam filtered-backprojection algorithm for truncated helical data. *Phys. Med. Biol.* **1998**, *43*, 2885. [CrossRef]
54. Hubbell, J.H.; Seltzer, S.M. *Tables of X-ray Mass Attenuation Coefficients and Mass Energy-Absorption Coefficients 1 keV to 20 MeV for Elements Z = 1 to 92 and 48 Additional Substances of Dosimetric Interest*; Technical Report; Ionizing Radiation Div., National Institution of Standards and Technology-PL: Gaithersburg, MD, USA, 1995. [CrossRef]
55. Coban, S.B.; Lucka, F.; Palenstijn, W.J.; Van Loo, D.; Batenburg, K.J. Explorative Imaging and Its Implementation at the FleX-ray Laboratory. *J. Imaging* **2020**, *6*, 18. [CrossRef]
56. Lagerwerf, M.J.; Coban, S.B.; Batenburg, K.J. High-resolution cone-beam scan of twenty-one walnuts with two dosage levels. *arXiv* **2020**, arXiv:2010.00421. [CrossRef]
57. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
58. van der Walt, S.; Schönberger, J.L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J.D.; Yager, N.; Gouillart, E.; Yu, T.; The Scikit-Image Contributors. Scikit-image: Image processing in Python. *PeerJ* **2014**, *2*, e453. [CrossRef]
59. Bernard, A.; Hamdy, S.; Le Corre, L.; Dirlewanger, E.; Lheureux, F. 3D characterization of walnut morphological traits using X-ray computed tomography. *Plant Methods* **2020**, *115*, 16. [CrossRef]
60. Dice, L.R. Measures of the amount of ecologic association between species. *Ecology* **1945**, *26*, 297–302. [CrossRef]
61. Walt, S.v.d.; Colbert, S.C.; Varoquaux, G. The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng.* **2011**, *13*, 22–30. [CrossRef]
62. Adler, J.; Kohr, H.; Öktem, O. *ODL 0.6.0*; ODL, Inc.: Zeeland, MI, USA, 2017. [CrossRef]
63. Van Aarle, W.; Palenstijn, W.J.; Cant, J.; Janssens, E.; Bleichrodt, F.; Dabravolski, A.; De Beenhouwer, J.; Batenburg, K.J.; Sijbers, J. Fast and flexible X-ray tomography using the ASTRA toolbox. *Opt. Express* **2016**, *24*, 25129–25147. [CrossRef]
64. Frigo, M.; Johnson, S.G. The design and implementation of FFTW3. *Proc. IEEE* **2005**, *93*, 216–231. [CrossRef]
65. Behnel, S.; Bradshaw, R.; Citro, C.; Dalcin, L.; Seljebotn, D.; Smith, K. Cython: The Best of Both Worlds. *Comput. Sci. Eng.* **2011**, *13*, 31–39. [CrossRef]
66. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.
67. Hendriksen, A.A.; Pelt, D.M.; Palenstijn, W.J.; Coban, S.B.; Batenburg, K.J. On-the-Fly Machine Learning for Improving Image Resolution in Tomography. *Appl. Sci.* **2019**, *9*, 2445. [CrossRef]

68. Lagerwerf, M.J. Neural Network FDK Algorithm. Available online: https://github.com/MJLagerwerf/nn_fdk (accessed on 17 November 2020).
69. Nguyen, D.; Widrow, B. The truck backer-upper: An example of self-learning in neural networks. In *Advanced Neural Computers*; Elsevier: Amsterdam, The Netherlands, 1990; pp. 11–19.

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).