3D Helical CT Reconstruction With a Memory Efficient Learned Primal-Dual Architecture

Jevgenija Rudzusika, Buda Bajić, Thomas Koehler, and Ozan Öktem

Abstract—Deep learning based computed tomography (CT) reconstruction has demonstrated outstanding performance on simulated 2D low-dose CT data. This applies in particular to domain adapted neural networks, which incorporate a handcrafted physics model for CT imaging. Empirical evidence shows that employing such architectures reduces the demand for training data and improves upon generalization. However, their training requires large computational resources that quickly become prohibitive in 3D helical CT, which is the most common acquisition geometry used for medical imaging. This paper modifies a domain adapted neural network architecture, the Learned Primal-Dual (LPD), so that it can be trained and applied to reconstruction in this setting. The main challenge is to reduce the GPU memory requirements during the training, while keeping the computational time within practical limits. Furthermore, clinical data also comes with other challenges not accounted for in simulations, like errors in flux measurement, resolution mismatch and, most importantly, the absence of the real ground truth. To the best of our knowledge, this work is the first to apply an unrolled deep learning architecture for reconstruction on full-sized clinical data, like those in the Low dose CT image and projection data set (LDCT).

Index Terms—Computed tomography, deep learning, helical acquisition, primal-dual, clinical data.

I. INTRODUCTION

CLINICAL computed tomography (CT) aims to computationally recover an image representing the interior anatomy of a subject by taking X-ray projections from different directions. Early approaches were inherently 2D in the sense that they aimed to recover a specific 2D cross section of the subject. To get a 3D image one has to repeat this for multiple adjacent 2D cross sections (step and shoot method). A major advance for 3D imaging came in the late 1980s with the introduction of helical scanning [1]. Here, the subject moves continually through the gantry while the x-ray source and corresponding detector is rotating. This corresponds to having an x-ray source on a helical path around the subject. Compared to step and shoot acquisition, helical CT scanning is much faster, e.g. one can scan many organs in a single breath-hold, thus reducing pulmonary motion artefacts. Helical scanning is therefore the de facto standard acquisition method for clinical CT.

First reconstruction methods for helical 3D CT were analytic methods that combine back-projection with a convolution using a suitable (reconstruction) filter. Since the filter in such a method depends on the acquisition geometry, early approaches relied on approximate recovery that combines re-binning to a simplified geometry, e.g., slice-wise 2D, with 2D-filtered back-projection (FBP) or 3D-Feldkamp-Davis-Kress (FDK) type of analytic reconstruction (see surveys [2], [3]). Novel reconstruction filters for FBP that provide theoretically exact recovery from helical CT data where introduced in 2003 [4]. Analytic reconstruction methods have fast reconstruction run-time and work very well in case of densely-sampled noise-free data. However, in the case of highly noisy data, these methods are not optimal, since they do not incorporate statistical information about the noise or the signal.

A fundamentally different approach is to define reconstruction as a solution to an optimization problem, where the goal is to minimize data discrepancy (considering statistical properties of the noise) while incorporating prior assumptions through additional regularizing components. These iterative model based methods have become increasingly popular since 2015 and are nowadays preferred over analytic reconstruction methods in clinical CT [5]. Here the acquisition geometry is explicitly encoded in the forward/back-projection model. However, arrays needed to store projection and image data in helical CT are very large, so the key challenge is to handle the computational burden. Currently, most iterative model based methods are stopped long before iterates have converged.

Recently there has been a surge of interest in data driven deep learning based approaches for CT reconstruction. Much of the development is catalysed by the possibility to significantly improve image quality in low-dose setting without compromising on computational feasibility [1], [6], [7]. A computationally efficient way to use deep learning for image reconstruction is to augment an analytical reconstruction method, such as FBP, [8], [9], [10]. This can be done by learning the convolutional kernel
for filtering the data and/or by applying the neural networks for pre-processing the data and/or for post-processing the reconstruction. Methods that focus solely on the post-processing step has been particularly popular, since they do not require access to the tomographic operators during training. For instance, one can train a convolutional neural network (CNN), like U-Net [11], to represent a mapping between a low dose and a high dose FBP reconstructions. However, in this case, the neural network does not directly access tomographic data and the performance of the method depends on the quality of initial reconstruction. An alternative approach is to construct a domain adapted architecture by unrolling an optimization scheme used in iterative model based methods. In this context, unrolling refers to setting a fixed number of iterations and substituting specific components of the optimization scheme with neural networks. Such unrolled architectures have been shown to outperform post-processing in various settings [12], [13], [14]. Unfortunately, the substantial demand for computational resources during training often constrains the size of problems one can address with deep learning without using graphics processing unit (GPU) super-computing resources and extensive software engineering. In particular, in clinical 3D helical CT the arrays for storing a 3D image and projection data can occupy up to 200 MB and 8 GB, respectively. With input data of this size, hardware limitations, such as, limited GPU memory, pose a major problem.

The overall aim of this paper is to develop a deep learning based reconstruction method that
a) delivers high image quality,
b) can be trained on high-end consumer GPU hardware,
c) and can be applied to the clinical 3D helical CT data.

We start out from the Learned Primal-Dual (LPD) architecture [15] that is a relatively simple unrolled architecture that has shown state-of-the-art performance in 2D CT reconstruction. Our main contributions are the following:

- We show how to overcome technical challenges and scale up the method to be applicable to 3D helical CT. This includes finding the right balance between GPU memory and computational time required for training. As a result, our methods can be trained on high-end consumer GPU hardware such as GeForce RTX 3090 GPU with 24 GB of memory in 3-6 weeks;
- We propose changes in the neural network architecture that are specifically suited for helical geometry and lead to additional performance gains;
- We apply the method to clinical data from the Low dose CT image and projection (LDCT) data set. This includes bridging the gap between the real data and simulations, while dealing with the absence of the actual ground truth. Although we focus on the LPD architecture, the ideas presented in this paper are rather general and can be used to apply other learned reconstruction methods to clinical data. This should be particularly true for other unrolled architectures.

II. RELATED WORK

Certain features in an architecture of a neural network allow to reduce its memory footprint during training at the cost of additional computations. As an example, invertibility is a property that allows to recompute activations of intermediate layers, while performing back-propagation [16]. Inspired by this idea, [17] modified the architecture of the LPD method [15], so that its unrolled iterations become invertible. As a result, the memory footprint of the method was reduced to the footprint of one invertible block. [14] used invertibility combined with patch-wise evaluation of convolutional operations within the invertible blocks, which allowed to shift the balance between memory and computational time even further. The drawback of enforcing invertability is that it likely results in reduced model capacity so the performance is expected to decrease when compared to the original LPD architecture [17]. In addition, a forward pass through each invertible block has to be done three times, which significantly increases computational time. Another approach proposed in [18] is inspired by neural ordinary differential equations (ODEs) [19]. Here, a solution to a reconstruction problem is represented by a function that evolves over time and the derivative of this function is represented by a neural network. The final solution can be obtained through numerical integration, which results in repeated applications of the network with additive skip-connections. This architecture does not require storing activations of intermediate layers as the gradients can be computed by an ODE solver instead of traditional back-propagation. In this case, the memory footprint of the method is approximately equal to the footprint of the network representing the derivative. So far, all the methods described above have been applied in 3D CT reconstruction with a circular acquisition geometry. However, their application to helical geometry requires additional improvements.

Furthermore, the memory footprint can be reduced by reducing the amount of input data that is processed by a neural network at the same time. Two natural approaches for achieving this are: 1) down-sampling the input data; 2) splitting the input data into parts. The down-sampling strategy has been explored in [20], where the learned gradient-descent is applied in different scales, starting from a coarse reconstruction and then gradually increasing resolution to reconstruct small details. However, since the proposed architecture of a neural network involves a block that acts on the full resolution the memory footprint of this method is still large.

Splitting the tomographic data is a known concept in tomography. In ordered subsets methods [21], [22] only a subset of the data is used to update an image volume during one iteration of iterative reconstruction. This leads to initial acceleration of various optimization methods. Recently, this idea has been used to modify the LPD architecture [23] with the goal of improving computational efficiency of the method. However, Learned Stochastic Primal-Dual (LSPD) method presented in [23], relies on a stochastic choice of the section of the data, while in this work we select the sections subsequently in order to ensure that all the data has been used. The idea of splitting image volumes is partially explored in [24], where images are split into parts (patches) and those are used within greedy training of learned gradient-descent. However, greedy training often results in sub-optimal performance compared to end-to-end training, as it was observed in [17].
Finally, there are a lot of works that have used deep learning for learning the mapping between the low dose and the normal dose reconstructions in the LDCT data set [25]. However, very few works have utilized the clinical projection data from this data set, even fewer without re-binning it to a parallel 2D geometry. One exception is [10], where an FBP-based architecture was trained using an unsupervised loss in the data domain. As a result, the training required neither ground truth reconstructions, nor a reference reconstruction method. As we discuss in Section V, the absence of the ground truth reconstructions is indeed an important issue to address. We find the unsupervised training strategies to be very promising and note that they can be applied to the LPD as well.

III. METHOD

First, we provide the theoretical foundations for deep learning based reconstruction methods. Then we outline the original LPD architecture and describe the proposed modifications. Lastly, we propose different strategies for training and inference using helical CT data.

A. Theoretical Foundations

Learned reconstruction methods use principles from statistical decision theory to learn ‘optimal’ parameters for a parameterized reconstruction method from training data [6, Section 5]. Consider a setting where one has access to supervised training data in the form of i.i.d. samples \((f_1, g_1), \ldots, (f_N, g_N) \in \mathbb{X} \times \mathbb{Y}\) of \((f, g)\) where \(f\) and \(g\) are random variable generating 3D images and helical CT data, respectively, and

\[ g = A(f) + e. \]  

The mapping \(A : \mathbb{X} \to \mathbb{Y}\) (ray transform) models how a 3D image gives rise to helical CT data in the absence of observation errors, and the random variable \(e\) represents observation error.

Our aim is now to learn the ‘best’ reconstruction method from the above supervised training data. Given a pre-defined parametrized family \(\{\mathcal{R}_\theta\}_\theta\) of possible reconstruction methods \(\mathcal{R}_\theta : \mathbb{X} \to \mathbb{X}\) and an image-loss \(L_X : \mathbb{X} \times \mathbb{X} \to \mathbb{R}\), the ‘best’ can be defined as a Bayes’ estimator

\[ \mathcal{R}_\hat{\theta} : \mathbb{X} \to \mathbb{X} \text{ where } \hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} L_X(\mathcal{R}_\theta(g_i), f_i). \]  

The choice of the image-loss governs the type of estimator one seeks to approximate. Choosing squared \(\ell_2\)-loss \(L_X(f, f') := \|f - f'\|_2^2\) means that the learned reconstruction method approximates the posterior mean, i.e.,

\[ \mathcal{R}_\hat{\theta}(g) \approx \mathbb{E}[f \mid g = g] \]  

A key component is the choice of parametrization for the family of reconstruction methods \(\{\mathcal{R}_\theta\}_\theta\). One could select a generic deep neural network (DNN) architecture, consisting of fully connected layers. However, in CT, such architectures easily become prohibitively large due to the size of the input and output data. An alternative approach is to consider \(\mathcal{R}_\theta := C_\theta \circ \mathcal{A}_1\) where \(\mathcal{A}_1 : \mathbb{Y} \to \mathbb{X}\) is an approximate inverse of \(\mathcal{A}\) and \(C_\theta : \mathbb{X} \to \mathbb{X}\) is a post-processing operator that is learned from training data. A particularly popular neural network architecture for the latter is U-Net [11]. A different path to domain adaptation is through unrolling. The idea is to start with some iterative scheme, like one designed to minimize \(f \mapsto \|A(f) - g\|_2^2\). The next step is to truncate this scheme and complement/replace the updates with possibly shallow CNNs. \(\mathcal{R}_\theta\) is then a DNN that consists of blocks representing unrolled iterations, where each iteration includes evaluation of the forward operator \(\mathcal{A}\) and/or its adjoint \(\mathcal{A}^*\) [6, Sec. 4.9.1]. This DNN is domain adapted since the operators \(\mathcal{A}\) and \(\mathcal{A}^*\) encode an explicit physics based-model for CT data acquisition. It is worth noting that unrolling is merely a way to select a DNN architecture for \(\mathcal{R}_\theta\). In particular, training as in (2) will not yield a solution operator for an optimization problem.

B. Architectures

A basic unrolled scheme for CT reconstruction is the LPD network introduced in [15]. In the following subsections, we briefly describe the original method, LPD, and technical steps that are sufficient to use this method in a 3D setting essentially without altering the architecture. We will refer to this method as LPDo. Further we propose improvements of the method specifically suited for helical geometry, LPDh.

1) Original LPD (LPDo): The LPDo architecture is inspired by the iterative scheme in the primal dual hybrid gradient (PDHG) algorithm [26]. More precisely, the LPDo architecture is given in Algorithm 1. The functions \(\Gamma^i\) and \(\Lambda^i\) are CNNs that operate on the dual and the primal space, respectively. For each unrolled iteration \(i\) these networks have the same architecture but different learned parameters. To simplify notation, we suppress the explicit dependence of \(\Gamma^i\) and \(\Lambda^i\) on the learned parameters \(\theta \in \Theta\).

In the original LPD paper [15], \(\Gamma^i\) and \(\Lambda^i\) are convolutional neural networks with two hidden layers and parametric Rectified Linear Unit (ReLU) activation functions. Here, we substitute the 2D convolutional layers by 3D convolutional layers and use simple ReLU activation functions, because of the possibility to

---

Algorithm 1: LPDo [15]†.

1: Choose initial primal and dual variables 
\((f_0, u_0) = \text{init}(g)\), where \((f_0, u_0) \in (\mathbb{X}^{N_p}, \mathbb{Y}^{N_d})\)
2: For \(i = 1, 2, \ldots, M\) do:
3: Dual update: \(u_i = u_{i-1} + \Gamma^i(u_{i-1}, \mathcal{A}(f_{i-1}^{(2)}, y))\)
4: Primal update: \(f_i = f_{i-1} + \Lambda^i(f_{i-1}, \mathcal{A}^* u_{i-1}^{(1)})\)
5: return \(f_M^{(1)}\)

\(M\) is the number of unrolled iterations, \(N_p, N_d\) is the number of memory channels, superscript \((i)\) denotes the \(i\)th channel.

†Note that this LPDo formulation slightly differs from the one given in Algorithm 3 in [15], but it still reflects the actual implementation used in experiments in [15].
evaluate the function in-place without using additional memory. Primal and dual variables \((f_0, u_0)\) are initialized with zeros unless stated otherwise.

As with all deep neural networks, the drawback of LPDo is its large GPU memory footprint during the back-propagation, which is too large for end-to-end training on high-dimensional input data.

a) Check-pointing: Gradient check-pointing is an “off-the-shelf” technique to reduce the memory footprint of a neural network during the training at the cost of computational time [27]. Instead of storing activation values of the whole computational graph in memory, it is possible to save values of selected layers (checkpoints) during the forward pass and recompute the graph part by part from these checkpoints during the back-propagation. In a typical scenario, a part of the graph that has to be recomputed should fit into the GPU memory. Therefore, when the memory is limited there is an incentive to split the graph into smaller parts. Furthermore, the checkpoints can be stored in central processing unit (CPU) memory. Unfortunately, copying the data between devices also takes time. Therefore, it is not beneficial to copy very large arrays. In the case of LPDo, certain balance is achieved when the checkpoints are done between the unrolled iterations, meaning that only inputs and outputs of the neural networks \(\Gamma^i\) and \(\Lambda^i\) are saved, while all intermediate computations are recomputed. The reason is that hidden layers inside the networks have much more channels than primal and dual variables \((f_i, u_i), i = 1, \ldots, M\) (typically 32 channels in the hidden layers and 5 channels for the primal and dual variables).

b) Splitting the layers: Despite a very large reduction in memory requirements achieved with check-pointing, it is still not sufficient for training LPDo on helical CT data on a single GPU of standard capacity. The reason is that the size of a typical helical scan of an abdomen is about 7 GB, while the size of a typical reconstruction is about 200 MB. In LPDo one check-pointed block \((\Gamma^i, \Lambda^i)\) is a shallow neural network applied either in the data domain or in the image domain. If this network has 2 hidden layers with \(n\) channels, then the memory footprint will be at least \(2n + 1\) multiplied by the size of an input (7 GB). Evidently, with such a large amount of input data, even a very shallow network does not fit into the memory. One solution is to de-parallelize a CNN by splitting the input arrays into spatial blocks and applying the network to these blocks sequentially [14]. To ensure that the output values are not affected by the splitting, it is necessary to use a small overlap between blocks. The size of this overlap depends on the field of view of the output neurons and, in case of a 3 layer CNN without strides, it is necessary to add 3 cells to each block from each side.

2) LPD for Helical CT (LPDh): Here, we propose a different approach for “de-parallelization” of a single unrolled LPD iteration. We partition the projection data into non-overlapping sections and identify the sub-volumes in the image domain that correspond to those sections. Inspired from the ordered subsets optimization methods [21], we subsequently apply unrolled LPD iterations to these sub-volumes using only one data section at a time. Further follows a detailed description of the method.

Each element in the tomographic data corresponds to the attenuation of a ray propagating through an object. Mathematically (before discretization and without noise) this can be seen as an integral over a line \(l\) defined in the image domain:

\[
g(l) = A(f)(l) = \int_l f \, ds. \tag{4}
\]

We process only a subset of these line integrals at a time that correspond to a contiguous section of data. These lines pass through the object of interest in a limited region in the image domain. Thus, instead of applying a neural network \(A^i\) to the whole image volume, we apply it only to a sub-volume, which is crossed by the aforementioned lines.

In practice, signals (images) and data measured with 3D helical CT are digitized and represented by arrays. The image space is \(X = \mathbb{R}^{I_w \times I_h \times N_z}\) where \(I_w\) and \(I_h\) are image width and height, respectively, and \(N_z\) is the number of slices along the rotation axis (z-axis). Note that \(N_z\) can be different for different scans, while the other two dimensions are fixed. The data space is \(Y = \mathbb{R}^{N_s \times D_w \times D_h}\) where \(N_s\) is the total number of angles/source positions and \(D_w\) and \(D_h\) correspond to the number of detector columns and detector rows, respectively. As in the image space, the number of source positions along the trajectory \((N_s)\) can be different, while the dimensions of the detector are fixed.

We split the data space \(Y\) into sub-spaces that correspond to the data acquired from the parts of a helical trajectory:

\[
Y = Y^1 \times Y^2 \times \cdots \times Y^{N_s} \times Y^{N_s+1}. \tag{5}
\]

Here, the \(j\)th data subspace is \(Y^j := \mathbb{R}^{N_s \times D_w \times D_h}\), where number of angles in one part equals to \(N_s\) for \(j = 1, 2, \ldots, N_s\). For simplicity fix \(N_s\) and discard the data corresponding to \(Y^{N_s+1}\), i.e. the data observed from the source at the end of the trajectory. Thus, \(N_s\) is the number of complete subspaces of the same dimension. We split the (truncated) data accordingly:

\[
g = (g^1, g^2, \ldots, g^{N_s}). \tag{6}
\]

Next, for each section of data, we consider a corresponding region in the image domain, from which this data has been obtained. Thus, from one image volume \(f\) we extract \(N_s\) overlapping sub-volumes

\[
(f^1, f^2, \ldots, f^{N_s}) \quad \text{where} \quad f^j \in X^j \quad \text{for} \quad j = 1, \ldots, N_s. \tag{7}
\]

Each image subspace is \(X^j = \mathbb{R}^{I_w \times I_h \times N_t}\), where \(N_t\) is the minimal thickness of image sub-volume obtained by adding the physical detector height and the largest pitch. In this case, image subspace \(X^j\) contains all the line segments necessary to compute the data in subspace \(Y^j\). Note that the image sub-volumes \(f^j\) overlap, but the data sections \(g^j\) do not (see visualization in Fig. 1).

We now define the restriction of the forward operator \(A : X \rightarrow Y\) to \(X^j\) as \(A^j : X^j \rightarrow Y^j\), i.e., it is an intersection of rows and columns from \(A\) corresponding to the \(j\)th data section and the \(j\)th sub-volume, respectively. Next, we define projection operators

\[
P_{X^j} : X \rightarrow X^j, \quad \text{s.t.} \quad P_{X^j}(f) = f^j, \tag{8}
\]

\[
P_{Y^j} : Y \rightarrow Y^j, \quad \text{s.t.} \quad P_{Y^j}(g) = g^j. \tag{9}
\]
the dual variable are $N_d = 1$ instead of $5$ and we use 16 channels instead of 32 in the hidden layers of the dual mapping $\Gamma$.  

C. LPDh with gradients (LPDhgrad)  

When the number of training iterations is limited, the method can benefit from a better initialization. First of all, we initialize the primal variables using an approximate FBP reconstruction method implemented in Operator Discretisation Library (ODL). The method is approximate in a sense that the convolutional filter is designed for parallel geometry, therefore it does not properly account for the cone beam, for the pitch of helical trajectory and for the Flying Focal Spot (FFS). However, we handle the redundant data using the Tam-Danielson window [28]. Secondly, we add more structure to primal and dual updates by using functions of the form  

$$ \Gamma^i(u, a, g) = a - g + \alpha_i \bar{\Gamma}^i(u, a, g), $$  

$$ \Lambda^i(f, a) = -\beta_i a + \gamma_i \bar{\Lambda}^i(f, a), \quad (11) $$  

where $\bar{\Gamma}^i, \bar{\Lambda}^i$ are neural networks and $\alpha_i, \beta_i, \gamma_i$ are learnable step size parameters. In addition, we remove a skip connection in the dual update  

$$ u_k = \bar{\mathcal{P}}_{Y^j_i} \left( \Gamma^i \left( \bar{u}_{k-1}, \bar{A}^i \left( f^{(1)}_{k-1} \right) \right), \bar{g} \right), \quad (12) $$  

so that, when the step size variables $\alpha_i, \gamma_i$ are set to $0$, the method is equivalent to the gradient descent:  

$$ f_k = f_{k-1} + \bar{\mathcal{P}}_{X^i_j} \left( -\beta_i (\bar{A}^i)^* \left( A^i \left( f^{(1)}_{k-1} \right) - \bar{g} \right) \right) $$  

which serves as a good starting point for the training process.

D. LPDh with U-Nets (LPDhunet)  

Since the most time consuming operation during the training is evaluation of the tomographic operators, we reduce the number of unrolled iterations in LPDh from 10 to 5. To compensate for the loss in model capacity, we use U-Nets [11] with 32 base channels for the networks in the primal domain $\bar{\Lambda}^i, i = 1, \ldots, 5$. Since there are two times less unrolled iterations in LPDhunet and LPDhgradunet, training takes less time. We use this opportunity to double the number of training iterations for LPDhgradunet (training $\times 2$), so that the total training time closer to LPDh.

E. Training and Testing Schemes  

Even though the above steps are sufficient to process clinical projection data of any size, the training on full-sized data would take months, if not years. In order to facilitate faster training, we train against smaller samples of the projection data and corresponding 3D reconstructions. Specifically, we train the CNNs $\Gamma^i$ and $\Lambda^i$ in an end-to-end manner using data corresponding to $k$ consecutive sections. The loss for the training pair $(f_i, g_i)$ is computed as  

$$ \mathcal{L}_X \left( \mathcal{R}_\theta \left( \mathcal{P}_{Y^j_i} (g_i) \right), \mathcal{P}_{X^j_i} (f_i) \right), \quad (14) $$  

where $\mathcal{L}_X$ is the standard $\ell_2$-norm and $q$ is a random number from 1 till $N_q - k$. The projection operator defined in the data

\begin{algorithm}  
\caption{LPDh.}  
\begin{algorithmic}[1]  
\State Choose initial primal and dual variables $(f_0, u_0) = \text{init}(g)$, where $(f_0, u_0) \in (X^{N_p}, Y^{N_d})$  
\For {$i = 1, 2, \ldots, M$}  
\For {$j = 1, 2, \ldots, N_s$}  
\State $k = iN_s + j$  
\State Select section: $\bar{u}_{k-1} = \mathcal{P}_{X^j_i} (u_{k-1}), \bar{g} = \mathcal{P}_{Y^j_i} (g)$  
\State Select sub-volume: $f_{k-1} = \mathcal{P}_{X^j_i} (f_{k-1})$  
\State Dual: $u_k = u_{k-1} + \bar{\mathcal{P}}_{Y^j_i} \left( \Gamma^i (\bar{u}_{k-1}, \bar{A}^i (f_{k-1}^{(1)}), \bar{g}) \right)$  
\State Primal: $f_k = f_{k-1} + \bar{\mathcal{P}}_{X^j_i} \left( \Lambda^i (f_{k-1} (\bar{A}^i)^* (u_{k}^{(1)})) \right)$  
\EndFor  
\EndFor  
\State \textbf{return} $f_{M_{N_s}}^{(1)}$  
\end{algorithmic}  
\end{algorithm}  

$M$ is the number of unrolled iterations, $N_s$ is the number of data sections/image sub-volumes, $N_p, N_d$ is the number of memory channels, superscript (i) denotes the $i$th channel.

Lastly, we define zero padding operators  

$$ \bar{\mathcal{P}}_{X^j_i} : X^j \rightarrow X, \quad \text{and} \quad \bar{\mathcal{P}}_{Y^j_i} : Y^j \rightarrow Y. \quad (10) $$  

The proposed LPDh method is now outlined in Algorithm 2. The networks $\Gamma^i$ and $\Lambda^i$ have the same architecture but different weights for $i = 1, 2, \ldots, M$. However, the same weights are used within the inner loop $j = 1, 2, \ldots, N_s$. This implies that the method can be applied with different numbers of data sections/line subsets $N_s$, which means that it can be applied to any length of the helical trajectory.

An advantage of LPDh from the optimization perspective is that the updated part of the image serves as input to the next sub-iteration. In contrast, the original LPDo uses all voxels in the image domain. Looking from the deep learning perspective at this approach, we can say that the connectivity between spatially distant neurons within the neural network is increased without increasing the number of network parameters. We empirically compare LPDo and LPDh in Section IV-C. We chose the section size to be equal to a single half-turn of a helical trajectory, so that back-propagation through one unrolled sub-iteration can be done on a GPU with 24 GB of memory. We use slightly smaller CNNs compared to [15]: The number of memory channels for the dual are set to 0, the method is LPDo.

\begin{figure}[h]  
\centering  
\includegraphics[width=0.8\textwidth]{figure1.png}  
\caption{Each sub-iteration of LPD for helical CT (LPDh) is applied to a section of helical data (left) and a part of image volume (right). One image sub-volume corresponds to the data obtained from the source positions on the helix marked with the same color. Note that image sub-volumes overlap, while data is partitioned without overlaps.}
\end{figure}
domain selects \( k \) subsequent data sections, i.e. \( \mathcal{P}_{Y^{(q+q+k)}} : Y \rightarrow Y^{q+q+k} \) where
\[
\mathcal{P}_{Y^{(q+q+k)}}(g) = (g^q, \ldots, g^{q+k-1}) \quad (15)
\]
The projection operator defined in the image domain selects the union of \( k \) image sub-volumes corresponding to the data, i.e. \( \mathcal{P}_{X^{(q+q+k)}} : X \rightarrow X^{q+q+k} \) where
\[
\mathcal{P}_{X^{(q+q+k)}}(f) = f^{q+j-1} \quad \text{for } j = 1, \ldots, k. \quad (16)
\]

a) Simple evaluation: During the evaluation one can simply apply Algorithm 2 using a different number of sections \( k \), i.e. using all the sections. We hypothesize that LPDh can generalize to much larger input data than it was trained for if the number of sections during the training is sufficiently large. This assumption is checked empirically in Section IV-C, where we investigate how the size of the input data (number of sections \( k \)) during the training influences the performance.

b) Sliding window: An alternative strategy is to evaluate the method on the same number of sections as it was trained for. Then the whole sequence of data sections can be processed in a “sliding window” manner. As a result, we obtain independent reconstructions for overlapping sub-volumes. We combine the obtained reconstructions by weighted averaging, where a weight for each slice in a sub-volume depends on the distance of this slice to the center of the sub-volume. The further the slice is from the center, the smaller its weight is. More precisely, let us denote the center along the rotation axis of the reconstructed sub-volume with index \( q \) by \( z_q^0 \) and its thickness by \( z_q^q \). Then, the reconstruction \( \hat{f}_z \) of the 2D slice \( f_z \) is obtained from the independently reconstructed sub-volumes \( \hat{f}_z = \mathcal{R}_\theta(\mathcal{P}_{Y^{(q+q+k)}}(g)) \) for \( q = 1, \ldots, N_s - k \) as follows:
\[
\hat{f}_z = \sum_{q=1}^{N_s} w_{z,q} \hat{f}_z
\]
\[
\text{where } w_{z,q} = \begin{cases} 
1 - \frac{2}{z_l^q} |z - z_q^q|, & \text{if } |z - z_q^q| \leq z_l^q/2 \\
0, & \text{otherwise.} 
\end{cases} \quad (17)
\]

IV. EXPERIMENTS AND RESULTS

First, we use simplified training data to select hyper-parameters and to compare the different approaches described in the previous section in a so-called ablation study. Then, we train the LPDh using realistic training data and perform a quantitative comparison to baseline methods. Finally, we evaluate the trained network on the (pre-processed) real clinical data.

A. Data

We simulate experimental data using LDCT data set [25] provided by the Cancer Imaging Archive. Furthermore, we evaluate the proposed methods on the real projection data from this data set. The data set contains helical CT scans of 299 patients involving head, thorax, and abdomen acquired using scanners from Siemens Healthcare and GE Healthcare. Due to differences in scanning protocols for different vendors and different body parts, we chose to focus only on scans of the abdomen (47 in total) acquired using a Siemens system. Of these, 42 are used for training, one for validation (‘L134’), and four for testing (‘L072’, ‘L019’, ‘L116’, ‘L150’). For each case, one has full dose and reduced dose (25%) projection data along with corresponding reference 3D reconstructions. The average size of the projection data in the test set is around \((N_o, D_u, D_h) = (6000, 736, 64)\). The average size of the image volumes is \(I_w = (512, 512, 190)\).

a) Simulated data for training: We use the real acquisition geometries and noise statistics available in the LDCT data set to generate low-dose projection data from the corresponding full dose 3D reconstructions. Specifically, we use cone beam helical geometries with cylindrical detectors that include z-FFS and quarter-pixel detector shift. FFS refers to periodically shifting the source so that the rays observed from two subsequent source positions interlace [29]. This allows to double the sampling rate along a detector axis without the need to physically reduce the size of detector elements. In particular, z-FFS improves the sampling along the detector axis parallel to the rotation axis. Detector shift, on the other hand, is introduced for the opposing rays to interlace, which improves the sampling along the detector axis perpendicular to the rotation axis [29]. Furthermore, clinical data contains varying pitch and signal-to-noise ratio. All the methods are trained to handle this implicitly.

Specifically, the simulation process involves the following steps. First, the input 3D reconstruction \( f \), which is given in the Hounsfield units is re-scaled back to attenuation values using \( \mu_0 := 0.0192 \text{ mm}^{-1} \) for the X-ray attenuation of water. Next, noise-free tomographic data \( g = A(f) \) is simulated using the ray transform with the original acquisition geometry provided together with the data. Then, the data is corrupted by Poisson noise
\[
H_{\text{noisy}} = \text{Poisson} \left( H_0 e^{-A(f)} \right), \quad (19)
\]
where \( H_0 \) is the number of photons per pixel available with the data. Finally, the data is linearized so that \( g \approx A(f) \) where
\[
g = - \ln \left( \frac{1}{H_0} H_{\text{noisy}} \right). \quad (20)
\]

The full dose reconstructions included in the data set (our ground truth) are slightly truncated from both ends along the rotation axis comparing to the region that was illuminated to generate the projection data. This is done, because reconstruction in those region is only partial, due to the lack of data from all angles. Therefore, when we simulate the data, we discard source positions in the beginning and at the end of a scan, leaving only those source positions, whose rays cross the ground truth volume and do not cross missing regions.

b) Simplified geometries for the ablation study: In order to perform the ablation experiments in a shorter time frame, we simplify the original acquisition geometries included in the data set by using a flat detector and not using FFS (we disregard every second angle).

c) Real data: Even though we use the real acquisition geometries and noise statistics as in the LDCT data set in our simulations, there is still a certain mismatch between the simulated
data and the real data caused by objects outside the scans field of view, errors in flux measurements and resolution mismatch. We pre-process the real data to address these issues.

First of all, the LDCT reference reconstructions, that we use as phantoms to generate the training data, do not cover objects that are outside the scanner’s field of view, such as cables, blankets, and the table. Therefore, our method is not trained to reconstruct those objects. Furthermore, visual examination of the real data shows that for some source angles there appears to be an additive error, which is almost constant along the first detector axis (detector width). This likely comes from normalizing data with incorrect gain estimates [30].

To address both of these issues we seek our own reference reconstruction using the full dose data. Since we don’t have access to other methods we construct a new reference reconstruction by solving the optimization problem:

$$\min_{f, c} L_Y(A(f), g + c),$$  \hspace{1cm} (21)

where \(c \in Y\) is a gain correction for each source position and detector row (constant across the columns) and the data discrepancy \(L_Y : Y \times Y \rightarrow \mathbb{R}\) is a weighted \(l_2\) norm [31]:

$$L_Y(A(f), g) = \sum_{i=1}^{|Y|} w_i \| A(f)_i - g_i \|^2 \ \text{where} \ w_i := e^{-g_i}.$$  \hspace{1cm} (22)

We solve (21) using Nesterov’s accelerated gradient descent [32] for 500 iterations starting from a LDCT full dose reference. We use the obtained reconstruction to correct the real data as follows

$$g_{c_{orr}} = g_{real} - A(M(\hat{f})) + \hat{c}$$  \hspace{1cm} (23)

where \(M : X \rightarrow X\) is an operator that masks the central part of the reconstruction, so that only the truncated regions are projected, and \(\hat{f}, \hat{c}\) are obtained by minimizing (21) as described above. We hypothesize that the corrections (23) could be (as well) performed using a full (not truncated) weighted FBP reconstruction for \(\hat{f}\) and \(g - A(\hat{f})\) averaged along the first detector axis for \(\hat{c}\). Alternatively, the training could be done on full (not-truncated) phantoms.

Lastly, the measured data has much higher resolution in the dimension along the rotation axis than the reference reconstructions. Minimizing data discrepancy (21) for image volumes \(f \in X\) that are too coarse results in typical “undershooting” artifacts near structures with high attenuation such as bones [33], [34]. To avoid this, we smooth the data along this dimension by applying a convolution with a normalized triangle filter with half width equal twice the size of reconstructed voxels.

### B. Implementation Details

The proposed method is implemented in Python with specific C++/CUDA based libraries for computationally demanding tasks. An example of the latter is CT-related components, which are implemented in the ODL [35] with ASTRA [36] as back-end for computing forward/back-projections. The neural network layers and training are implemented using PyTorch [37].

Training is performed with the Adam optimizer [38] using the \(l_2\)-distance to the ground-truth described in (14). The initial learning rate is set to \(5 \cdot 10^{-4}\) with cosine annealing. Since the training samples become larger with the increased number of sections, we define the total number of training iterations to be \(2 \cdot 10^5/(10 + 5 \cdot N_s)\), where \(10 + 5 \cdot N_s\) is a rough estimate of the number of slices in a target reconstruction for a given number of sections \(N_s\). However, we use \(10^5\) iterations by default, when training with the real geometries. The total training time depends on the GPU. In particular, we used GeForce RTX 3090 and A40 in the experiments. In the case of LPDh, the difference is not major, the training takes four and three weeks, respectively. For LPDh_grad with U-Nets (LPDhgradunet) training on A40 for \(2 \cdot 10^5\) iterations takes three weeks, which is almost two times faster than on RTX 3090.

Theoretically, the size of image sub-spaces \(X^j\) depends on the pitch. In practice, the pitch can vary across different scans and also within each scan (we observe values within the range 17.18 mm to 34.5 mm in our training and testing set). The method must therefore be robust with respect to these variations, assuming they are not too big. For simplicity, we set \(N_i = 16\), which is a suitable value for most of the scans.

Quantitative performance of all methods is measured by calculating the peak signal to noise ratio (PSNR) and the structural similarity index (SSIM) between the ground truth image and its corresponding reconstruction. However, since the data is truncated at the beginning and the end of acquisition geometry, we discard first 8 and last 8 reconstructed 2D slices, since reconstruction in those regions is only partial.

- **a) Huber regularization:** We compare our approach to a variational model with Huber regularizer, i.e., reconstruction obtained from \(R_{\theta} : Y \rightarrow X\) that is defined as

$$R_{\theta}(g) \in \arg\min_f \left\{ L_Y(A(f), g) + \lambda S_{\theta}(f) \right\}.$$  \hspace{1cm} (24)

Here, the data discrepancy is given as in (22) and the regularizer \(S_{\theta} : X \rightarrow \mathbb{R}\) is smooth relaxation of the non-smooth \(l_1\) norm that makes up the total variation (TV) regularizer [39]:

$$S_{\theta}(f) := \sum_{i=1}^{2 |X|} h_\theta(|\nabla f_i|) \ \text{where} \ h_\theta(t) := \begin{cases} t^2/(2\theta), & t \leq \theta \\ t - \theta/2, & t > \theta. \end{cases}$$  \hspace{1cm} (25)

In our experiments, we set the regularization parameter and the Huber parameter to \(\lambda = 0.15\) and \(\theta = 0.0012\), respectively. The optimization in (24) is solved using Nesterov’s accelerated gradient descent [32] that is stopped after 200 iterations. We initialize the optimization with an approximate FBP algorithm implemented in ODL. We chose the hyper-parameters and the number of iterations that lead to optimal performance on the validation data. In particular, early stopping does not only improve upon computational feasibility, it has also been shown to provide an additional regularising effect as outlined in [40].

- **b) U-Net post-processing:** To obtain an initial reconstruction, we mask redundant data using the Tam-Danielson window [28] and apply approximate FBP implemented in ODL. The method is approximate in a sense that the convolutional filter is designed for parallel geometry, therefore it does not properly account for
the cone beam, for the pitch of helical trajectory and for the FFS. To compensate for this imprecision we apply 10 accelerated gradient descent iterations to minimize (24) as in the Huber regularization described above. For the post-processing step we use a U-Net architecture described in [11]. Due to GPU memory limitations (24 GB) we train and evaluate the network on patches. We investigate two alternatives. First, we train the U-Net on patches of size $128^3$ and evaluate the network in a single forward pass on the full image. Alternatively, we use patches of size $512 \times 512 \times 16$ during training and evaluation. We obtain a full image volume by applying the network a sliding window manner along the last axis (shifting by 8 slices). The slices are weighted as in (17) and (18). Finally, we investigate if the method has reached its full potential by increasing the model capacity of the U-Net (96 base channels instead of 64). This, however, requires larger GPU memory. The training procedure is the same as for LPD methods, however, the number of training iterations is much higher $5 \times 10^4$. The training takes around 2 days.

### C. Ablation Study

In this section we evaluate the performance of LPDo and LPDh trained with different number of sections. Since training of the neural networks on a single GPU takes weeks, we use data simulated using simplified geometries (as described in Section IV-A). The time it takes to evaluate the method depends almost linearly on the number of evaluations of tomographic operators, which is listed along with quantitative performance results in terms of PSNR and SSIM in Table I.

First of all, both LPDo and LPDh perform poorly, if trained only using 2 subsequent sections. In this cases, it is much more beneficial to apply the method in a sliding window manner. However, as the number of sections during the training increases, the generalization to an unlimited number of sections (25–72 in our test cases) improves greatly. If 6 sections are used, there is almost no difference in performance between the sliding window testing strategy and the regular forward pass, suggesting that almost perfect generalization is achieved in this case. Ultimately, we can see that LPDh with 6 sections provides very good performance without an increase in estimation time (only 10 operator evaluations). Still, the best performing approach is LPDh trained with 4 sections and tested in sliding window manner. We evaluate the latter in the next sections using realistic simulated data and the real data.

### D. Results

Here we evaluate LPDh methods trained on 4 sections, since that showed the best performance in Section IV-C. All methods are trained using simulated data with geometries and noise levels from the LDCT data set. Table II lists quantitative performance results in terms of PSNR and SSIM, the number of operator evaluations, which closely correlates with execution time, and the number of learned parameters.

First, we can see that the performance of U-Net post-processing can vary greatly depending on the training and evaluation strategy. In particular, applying the network in a sliding window manner has significantly boosted the performance. On the other hand, increasing the model capacity has had minimal impact.

Next, we can see that U-Net post-processing can outperform the simplest version of LPDh. We hypothesise that the main reason is the limited number of training iterations in the case of LPDh. In contrast, the post-processing does not involve evaluation of tomographic operators during training, hence, the training is fast and the number of training iterations does not have to be limited.

To address the above issue we introduced LPDhgrad, which has more structured updates compared to LPDh and should be easier to train. Interestingly, the difference between the performance of LPDhgrad and LPDh is quite small. However, this additional structure becomes important when larger networks (U-Nets) are used for the primal updates, i.e. LPDhgradunet outperforms LPDhunet by a large margin.

Finally, increasing the number of training iterations greatly benefits LPDhgradunet, which supports the hypothesis that limited training hinders the performance of the unrolled methods evaluated in this work. Nevertheless, our final version of LPDh, LPDhgradunet, outperforms the baseline methods in terms of PSNR by a considerable margin.

Next, we perform visual comparison of the proposed methods applied to the real data (corrected as described in Section IV-A) in Fig. 2. We present two baseline reconstructions: the low dose
reference included in the data set and the iterative reconstruction with Huber regularization. Those are visually very similar since the latter was initialized with the former. We also show two full dose reconstructions: one that is included in the data set and our own reference obtained from the full dose data by addressing problem (21). Finally, Fig. 3. shows sagittal and coronal cross sections of 3D reconstructions obtained with LPDh.

V. LIMITATIONS

We do not present a quantitative comparison of reconstructions obtained from the real data. The reason is that the LDCT full dose reference does not represent the actual “ground truth” with sufficient accuracy. Fig. 2 shows that the iterative reconstruction computed in a time-unconstrained setting seems to provide a superior recovery of small details. On the other hand, our reference reconstruction is likely to contain more noise than the LDCT full dose reference. We are therefore hesitant to use it as the ground truth.

Yet another issue that undermines quantitative comparison is that full dose and low dose reconstructions obtained with the same method will have additional similarities that result from implicit/explicit prior information imposed by the method, let’s call it method’s bias. Thus, the low dose FBP reconstruction is closer in PSNR to the full dose FBP reconstruction, while iterative reconstructions, Huber regularization and LPDh, are closer to the iterative full dose reconstruction (our full dose reference).

Furthermore, the above issues should be taken into consideration when training the method in a supervised manner with the real data. We tried to train LPDh using the LDCT full dose references, but the performance was similar (slightly worse) to U-Net post-processing of the low dose reference. Nevertheless, we have seen in Fig. 2 that the full dose reference does not contain all the details that LPDh is able to reconstruct after being trained on simulated data. Moreover, the quantitative comparison confirms that LPDh reconstructions are closer to our own reference than both LDCT reference reconstructions.
One option would be to train LPDh using full dose iterative reconstructions as proxies for ground truth. However, obtaining such reconstructions for the whole data set requires a lot of computational resources and the quality of reconstruction would still be questioned.

To summarize, it is clear that there are differences between full dose reconstructions obtained with different methods. Furthermore, none of the methods can perfectly recover the actual “ground truth”. Obtaining a good proxy for the ground truth using full dose data in a time-unconstrained setting requires a separate investigation, which should be ideally performed by the maintainers of LDCT data set. An alternative approach is to avoid defining the ground truth and to use an unsupervised training loss, for instance, as in [10], [41], [42]. Since unsupervised training is typically more computationally demanding than supervised, it is probably beneficial to do supervised pre-training on simulated data first and then fine-tune the neural network on the real data using an unsupervised loss.

The final limitation is the lack of freely available software suitable for processing clinical CT data, such as LDCT. For instance, the open source software library FreeCT [43] that implements the weighted Filtered Back Projection (wFBP) algorithm from [44] currently does not support FFS. As a consequence, we used iterative reconstruction with Huber regularizer as the baseline method, even though the method is not practical due to its execution time.

VI. CONCLUSION

This work addresses the challenges of applying unrolled deep learning-based reconstruction methods, such as LPD, to clinical 3D helical CT data. Specifically, we demonstrate how to train the method on a single GPU within a reasonable time frame and propose architectural changes that further enhance its performance on helical CT data. Although we observe benefits from unrolling within the architecture of a neural network, we find that in the case of fully sampled clinical CT data the performance gains are harder to attain. Overall, the effectiveness of such architectures should be evaluated based on the ill-posedness of the problem and the available hardware resources for training.

REFERENCES

[1] P. J. L. Rivierea and C. R. Crawford, “From EMI to AI: A brief history of commercial CT reconstruction algorithms,” J. Med. Imag., vol. 8, no. 5, pp. 39–51, 2021.
[2] C. Bonitus and T. Köhler, “Reconstruction algorithms for computed tomography,” Adv. Imag. Electron Phys., vol. 151, pp. 1–63, 2009.
[3] J. Hsieh, B. Nett, Z. Yu, K. Sauer, J.-B. Thibault, and C. A. Bouman, “Recent advances in CT image reconstruction,”Curr. Radiol. Rep., vol. 1, pp. 39–51, 2013.
[4] A. Katsevich, “A general scheme for constructing inversion algorithms for cone beam CT,” Int. J. Math. Math. Sci., vol. 2003, no. 21, pp. 1305–1321, 2003.
[5] A. Mileto, L. S. Guimarães, C. H. McCollough, J. G. Fletcher, and L. Yu, “State of the art in abdominal CT: The limits of iterative reconstruction algorithms,” Radiology, vol. 293, no. 3, pp. 491–503, 2019.
[6] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, “Solving inverse problems using data-driven models,” Acta Numerica, vol. 28, pp. 1–174, 2019.
[7] C. M. McLeavy et al., “The future of CT: Deep learning reconstruction,” Clin. Radiol., vol. 76, no. 6, pp. 407–415, 2021.
[8] M. J. Lagerwerf, D. M. Pelt, W. J. Palenstijn, and K. J. Batenburg, “A computationally efficient reconstruction algorithm for circular cone-beam computed tomography using shallow neural networks,” J. Imag., vol. 6, no. 12, 2020, Art. no. 135.
[9] F. Wagner et al., “On the benefit of dual-domain denoising in a self-supervised low-dose CT setting,” in Proc. IEEE 20th Int. Symp. Biomed. Imag., 2023, pp. 1–5.
[10] O. Kosomaa, S. Laine, T. Karras, M. Aittala, and J. Lehtinen, “Simulator-based self-supervision for learned 3D tomography reconstruction,” 2022, arXiv:2212.07431.
[11] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, “Deep convolutional neural network for inverse problems in imaging,” IEEE Trans. Image Process., vol. 26, no. 9, pp. 4509–4522, Sep. 2017.
[12] J. Leuschnert et al., “Quantitative comparison of deep learning-based image reconstruction methods for low-dose and sparse-angle CT applications,” J. Imag., vol. 7, no. 3, pp. 44, 2021.
[13] M. Genzel, I. Gühring, J. Macdonald, and M. Müez, “Near-exact recovery for tomographic inverse problems via deep learning,” in Proc. Int. Conf. Mach. Learn., 2022, pp. 7368–7381.
[14] N. Moriaikov, J. Sonke, and J. Teuwen, “End-to-end memory-efficient reconstruction for cone beam CT,” Med. Phys., vol. 50, no. 12, pp. 7579–7593, 2023.
[15] J. Adler and O. Öktem, “Learned primal-dual reconstruction,” IEEE Trans. Med. Imag., vol. 37, no. 6, pp. 1322–1332, Jun. 2018.

[16] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, “The reversible residual network: Backpropagation without storing activations,” in Proc. Adv. Neural Inf. Process. Syst., Curran Associates, Inc., 2017, vol. 30. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/9be311e56811a9ad8150a60844bb94c-Paper.pdf

[17] J. Ruidzusika, B. Bajić, O. Öktem, C.-B. Schönlieb, and C. Ettmann, “Invertible learned primal-dual,” in Proc. NeurIPS 2021 Workshop Deep Learn. Inverse Problems, 2021.

[18] M. Thies, F. Wagner, M. Gu, L. Folle, L. Felsner, and A. Maier, “Learned cone-beam CT reconstruction using neural ordinary differential equations,” in Proc. 7th Int. Conf. Image Formation X-Ray Comput. Tomogr., 2022, vol. 12304, pp. 48–54.

[19] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural ordinary differential equations,” in Proc. Adv. Neural Inf. Process. Syst., S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett Eds. Curran Associates, Inc., 2018, vol. 31. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfe6d0692a24c8686939b9-Paper.pdf

[20] A. Hauptmann, J. Adler, S. R. Arridge, and Ö. Öktem, “Multi-scale learned iterative reconstruction,” IEEE Trans. Comput. Imag., vol. 6, pp. 845–856, 2020.

[21] H. M. Hudson and R. S. Larkin, “Accelerated image reconstruction using ordered subsets of projection data,” IEEE Trans. Med. Imag., vol. 13, no. 4, pp. 601–609, Dec. 1994.

[22] H. Erdogan and J. A. Fessler, “Ordered subsets algorithms for transmission tomography,” Phys. Med. Biol., vol. 44, no. 11, 1999, Art. no. 2835.

[23] J. Tang, “Stochastic primal-dual deep unrolling networks for imaging inverse problems,” 2021, arXiv:2110.10993.

[24] D. Wu, K. Kim, and Q. Li, “Computationally efficient deep neural network for computed tomography image reconstruction,” Med. Phys., vol. 46, no. 11, pp. 4763–4776, 2019.

[25] C. McCollough et al., “Low dose CT image and projection data (LDCT-and-Projection-data) (version 5) [data set],” Cancer Imag. Arch., 2020, doi: 10.7937/9NPB-2637.

[26] A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problems with applications to imaging,” J. Math. Imag. Vis., vol. 40, no. 1, pp. 120–145, 2011.

[27] T. Chen, B. Xu, C. Zhang, and C. Guestrin, “Training deep nets with spiral scan,” Phys. Med. Biol., vol. 43, no. 4, 1998, Art. no. 1015.

[28] J. Hsieh, “Computed tomography: Principles, design, artifacts, and recent advances,” Bellingham, Washington USA: SPIE Press, 2003.

[29] J. B. Thibault, Z. Yu, K. Sauer, C. Bouman, and J. Hsieh, “Correction of gain fluctuations in iterative tomographic image reconstruction,” in Proc. Int. Mtg. Fully 3D Image Recon. Rad. Nuc. Med., 2007, pp. 112–5.

[30] I. A. Elbakri and J. A. Fessler, “Statistical image reconstruction for polyenergetic X-ray computed tomography,” IEEE Trans. Med. Imag., vol. 21, no. 2, pp. 89–99, Feb. 2002.

[31] Y. E. Nesterov, “A method for solving the convex programming problem with convergence rate $o(1/k^2)$,” in Proc. USSR Acad. Sci., (Doklady Akademii Nauk SSSR), 1983, vol. 269, pp. 543–547.

[32] W. Zhijewski and F. J. Beekman, “Characterization and suppression of edge and aliasing artefacts in iterative X-ray CT reconstruction,” Phys. Med. Biol., vol. 49, no. 1, 2003, Art. no. 145.

[33] W. Zhijewski and F. J. Beekman, “Comparison of methods for suppressing edge and aliasing artefacts in iterative X-ray CT reconstruction,” Phys. Med. Biol., vol. 51, no. 7, 2006, Art. no. 1877.

[34] J. Adler, H. Kohr, and Ö. Öktem, “Operator discretization library (ODL),” 2017. [Online]. Available: https://github.com/odlgroup/odl

[35] W. Van Aarle et al., “Fast and flexible X-ray tomography using the ASTRA toolbox,” Opt. Exp., vol. 24, no. 22, pp. 25129–25147, 2016.

[36] A. Paszke et al., “Automatic differentiation in PyTorch,” in Proc. 31st Conf. Neural Inf. Process. Syst., 2017. [Online]. Available: https://openreview.net/pdf?id=BJJrmtfCZ

[37] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

[38] P. J. Huber, Robust Statistics, vol. 523. Hoboken, NJ, USA: Wiley, 2004.

[39] A. Effland, E. Kobler, K. Kunisch, and T. Pock, “Variational networks: An optimal control approach to early stopping variational methods for image restoration,” J. Math. Imag. Vis., vol. 62, no. 3, pp. 396–416, 2020.

[40] A. A. Hendriksen, D. M. Pelt, and K. J. Batenburg, “Noise2inverse: Self-supervised deep convolutional denoising for tomography,” IEEE Trans. Comput. Imag., vol. 6, pp. 1320–1335, 2020.

[41] D. Chen, J. Tschella, and M. E. Davies, “Robust equivariant imaging: A fully unsupervised framework for learning to image from noisy and partial measurements,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 5647–5656.

[42] J. Hoffman, S. Young, F. Noo, and M. McConnell-Day, “FreeCT_wFBP: A robust, efficient, open-source implementation of weighted filtered backprojection for helical, fan-beam CT,” Med. Phys., vol. 43, no. 3, pp. 1411–1420, 2016.

[43] K. Stierstorfer, A. Rauscher, J. Boese, H. Bruder, S. Schaller, and T. Flohr, “Weighted FBP—a simple approximate 3D FBP algorithm for multislice spiral CT with good dose usage for arbitrary pitch,” Phys. Med. Biol., vol. 49, no. 11, 2004, Art. no. 2209.