Optics EXPRESS

How auto-differentiation can improve CT workflows: classical algorithms in a modern framework

RICHARD SCHOONHOVEN,[1,2,*] ALEXANDER SKORIKOV,[1,2] WILLEM JAN PALENSTIJN,[2] DANIËL M. PELT,[2] ALLARD A. HENDRIKSEN,[1] AND K. JOOST BATENBURG[1,2]

[1] *Computational Imaging Group, Centrum Wiskunde & Informatica, 1098 XG Amsterdam, The Netherlands*
[2] *Leiden Institute of Advanced Computer Science, Leiden University, 2311 EZ Leiden, The Netherlands*
[*] *richard.schoonhoven@cwi.nl*

Abstract: Many of the recent successes of deep learning-based approaches have been enabled by a framework of flexible, composable computational blocks with their parameters adjusted through an automatic differentiation mechanism to implement various data processing tasks. In this work, we explore how the same philosophy can be applied to existing "classical" (i.e., non-learning) algorithms, focusing on computed tomography (CT) as application field. We apply four key design principles of this approach for CT workflow design: end-to-end optimization, explicit quality criteria, declarative algorithm construction by building the forward model, and use of existing classical algorithms as computational blocks. Through four case studies, we demonstrate that auto-differentiation is remarkably effective beyond the boundaries of neural-network training, extending to CT workflows containing varied combinations of classical and machine learning algorithms.

1. Introduction

In recent years, deep learning and other data-driven machine learning approaches have become increasingly popular in computed tomography. Deep neural networks have achieved strong results in X-ray CT applications by improving reconstruction quality [1], reducing metal artifacts [2], performing beam hardening correction [3], and classification [4–6]. The progress in deep learning has shown the power of data driven end-to-end optimization using auto-differentiation software, often in combination with hardware acceleration using graphical processing units (GPUs).

Despite promising results, deep learning approaches suffer from interpretability and reproducibility challenges. Furthermore, a serious issue is the behaviour of deep learning algorithms when presented with new data, and potential hallucination of object features [7]. These considerations have hampered the adoption of learned algorithms by experts in for example the medical domain where doctors can be reluctant to trust black-box learned approaches.

In contrast, classical (i.e. non-learned) algorithms often excel in interpretability and robustness. Additionally, classical algorithms often come with an intuition for their applicable input data domain, and the results they should produce. This can lead to expert users favouring such traditional methods over deep learning approaches. However, parameters of these algorithms are often either kept fixed or are adjusted by the user based on manual experimentation. Since these parameters are not learned in a data-driven way, they may not be optimally chosen for the particular dataset and application.

In this work, we apply concepts from the philosophy that made deep learning-based methods so successful, and transfer it to CT workflows. We create workflows of computed tomography algorithms for various problems and show how those problems can be solved by end-to-end

#502920 https://doi.org/10.1364/OE.502920
Journal © 2024 Received 8 Aug 2023; revised 20 Dec 2023; accepted 9 Jan 2024; published 28 Feb 2024

optimization based on auto-differentiation. By doing so, we reconcile classical algorithms with deep learning. We perform four case studies, representative of real-world tomography problems. To do so, we create pipelines using the following core design principles:

End-to-end learning: We implement the pipelines such that all pieces facilitate gradient propagation, meaning that parameters at all steps of the pipeline can be optimized jointly.

Explicit quality criteria: All pipelines use explicit quality criteria as objectives for optimization, thereby enabling automatic optimization of parameters.

Declarative algorithm construction: Each pipeline is created by building the forward model, and we optimize its parameters using auto-differentiation and generic readily-available algorithms.

Use existing building blocks: The pipelines re-use building blocks derived from both classical algorithms and deep learning methods in a way that enables gradient propagation. This allows for seamless compatibility with deep learning-based methods.

Our portfolio of case studies sketches the outline of a new generation of powerful software toolboxes that enable users to leverage the power of auto-differentiation for advanced computational CT pipeline construction.

This work is structured as follows. In Section 2. we explore related work in auto-differentiation software for classical methods, and approaches for learning CT algorithms in a data-driven way. In Section 3. we describe the methodology. In Section 4. we present our results in the form of four case studies. In Section 5. we discuss our findings and present key potential benefits that arise from our approach. We present our final conclusions in Section 6.

2. Related work

The idea of extending auto-differentiation techniques [8] to classical algorithms is actively investigated across several domains. In the field of robotics, for non-linear optimization problems, Meta AI constructed Theseus [9] which is a library for building custom non-linear optimization layers that were shown to be useful for differentiable kinematics. In [10], robotic controllers are auto-tuned using the DiffTune package which works with forward-mode auto-differentiation. Furthermore, end-to-end differentiable optimization enabled the coupling of the prediction and planning module in autonomous vehicles [11].

In the field of cosmology, JAX-Cosmo [12] is a recently developed end-to-end GPU accelerated library for cosmological calculations. Using automatic differentiation, JAX-COSMO exposes derivatives for certain cosmological quantities, and enables previously impractical methods such as Hamiltonian Monte Carlo and Variational Inference. Furthermore, by embedding the cosmological algorithms in JAX, the algorithms can be run on accelerated hardware, and can benefit from automatic code optimizations and techniques such as just-in-time compilation.

Embedding classical operators in neural networks has been demonstrated as an effective technique for end-to-end learning based on auto-differentiation. In [13] the tomographic backprojection operator is embedded as an algorithm within a neural network. Here the backprojection parameters are not trainable themselves, but rather the algorithm introduces prior knowledge about image reconstruction in the neural network. From the field of seismic imaging, a 3-step reflective seismic imaging method is learned end-to-end by turning the Delay-And-Sum (DAS) operator into a network layer in [14]. Similarly, the wave-physics-formation algorithm is placed between two neural networks in [15] to facilitate single-plane seismic wave imaging (SFW).

Automatic differentiation techniques have already been applied to solve specific problems in computed tomography in recent years. In [16], beam hardening correction for X-ray microscopy of mouse bones is performed with a polynomial correction model that is optimized through a PyTorch-based differentiable FDK algorithm. An elaborate outline of algorithmic differentiation for phase retrieval is presented in [17]. In [18,19], automatic differentiation is used in ptychography where the object wave function is obtained with a gradient-based method by minimizing the ptychography loss for each pixel. In [20] the 3D reconstruction problem for objects beyond depth-of-focus (DOF) is formulated as a minimization problem with a data fidelity and total variation term, which is solved with a gradient descent algorithm. How automatic differentiation techniques can be used for different imaging modalities is shown in [21], where compressive sensing, single image super resolution (SISR), and ptychography reconstructions are covered. For tomography, the reconstruction is obtained with a gradient based approach by minimizing a total variation functional, and the authors show how this is beneficial for sparse and limited angle data.

In the field of nanotomography, the auto-differentiation framework Adorym [22] for flexible reconstruction has been developed. In Adorym, a flexible forward model allows for optimization of experimental parameters such as probe position, object tilt, absorption/refraction relation coefficient, and propagation distance. The authors show improved reconstruction quality for ptychography and multi-distance holography reconstructions, by finetuning these experimental parameters. For conventional CT, they accelerate ART with gradient descent optimizers and acquire improved results over FBP.

Compared to the related work outlined above, the scope and focus of the present paper is more general. We focus on the core design principles underlying CT workflows using auto-differentiation and demonstrate the efficacy of the approach through a varied set of four case studies.

3. Methodology

3.1. Auto-differentiation

For the case studies included in this paper, we implement the CT workflows in the auto-differentiation framework PyTorch [23]. An auto-differentiation system breaks down a program in a series of primitive operations for which fixed procedures are known to compute derivatives. The series of primitive computations is collected in a computational graph. Most auto-differentiation systems, including PyTorch, trace the computational graph implicitly during the forward computation through the program. After the forward computation, the error or loss is computed. Here, we use gradient-based optimization, meaning that an auto-differentiation package needs to obtain partial derivatives of the loss with respect to the parameters.

Often each primitive function in an auto-differentiation package specifies vector-Jacobian products. Suppose two quantities are related by a primitive function $f(\mathbf{x}) = \mathbf{y}$, and $\mathbf{y}$ is used further in the computation. Denote by $\bar{\mathbf{y}}$ the derivative of a loss L with respect to $\mathbf{y}$. A vector-Jacobian product defines a way to express the derivatives of L with respect to $\mathbf{x}$, and is defined as

$$\frac{\partial L}{\partial x_j} = \sum_i \frac{\partial y_i}{\partial x_j}\frac{\partial L}{\partial y_i}, \qquad \bar{\mathbf{x}} = J^T\bar{\mathbf{y}},$$

where J is the Jacobian. For a primitive operation f, the gradient of the input $\bar{\mathbf{x}}$ can be calculated from the output gradient $\bar{\mathbf{y}}$, the input $\mathbf{x}$ and the output $\mathbf{y}$. For example, in the case of $y = f(x) = -x$ the gradient of the input can be computed as $-\bar{y}$, for $y = f(x) = e^x$ it will be $y \cdot \bar{y}$, and in the case of $y = \log(x)$ it will be $\frac{\bar{y}}{x}$. The procedure of computing gradients in reverse (starting from $\bar{L} = 1$) is called back propagation. For gradient descent, parameters are updated by

$$\mathbf{x} \leftarrow \mathbf{x} - \lambda\bar{\mathbf{x}},$$

for some step size λ (often called the learning rate in deep learning). Variants of gradient descent exist such as Nesterov's accelerated gradient descent [24]. In our experiments we will either use (stochastic) gradient descent, or Adam [25].

A drawback of auto-differentiation frameworks can be excessive memory consumption because copies of many intermediate outputs are stored for fast computation of the back propagation algorithm. This problem can be addressed by several general approaches. Gradient checkpointing stores only a subset of the intermediate outputs for gradient computation, whereas other outputs are recomputed during the back propagation step [26]. In this manner, computational performance can be traded for improved memory efficiency. Another approach involves just-in-time (JIT) compilation where a compiler attempts to optimize the computational graph into a more memory and compute efficient set of instructions. In this work, we apply PyTorch implementations of both techniques for certain memory or computationally expensive operations. In section 4.3, we illustrate the effectiveness of these techniques using one of the case studies and discuss the involved trade-off between computational performance and memory efficiency.

3.2. Computed tomography

In computed tomography [27] a 3D image of an object is recovered from a series of projections that are taken at different angles. Tomographic reconstruction can be modelled as the problem to recover an object volume $\mathbf{x} \in \mathcal{X} := \mathbb{R}^{N_x \times N_y \times N_z}$ from the measured projection data $\mathbf{y} \in \mathbb{R}^{N_\theta \times N_u \times N_v}$. Here, N_u and N_v are the number of detector rows and columns, and N_θ is the number of projection angles. The projection process can be approximated by a linear operator A, and can be expressed as a matrix using the aforementioned discretization

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

where $\mathbf{x}$ and $\mathbf{y}$ are collapsed to a vector. For parallel beam tomography the object can be reconstructed by the commonly used filtered backprojection algorithm (FBP)

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

Here, the projection data is convolved with a 1D filter $\mathbf{h} \in \mathbb{R}^{N_v}$ (Ram-Lak in this study), and subsequently backprojected by applying A^T. For circular cone-beam tomography, the object can be recovered by the Feldkamp-Davis-Kress (FDK) [28] algorithm, where the projection data is weighted in order to compensate for the diminishing intensity at distance from the detector center. An alternative to direct methods are iterative methods that reformulate the equation system 1 as an optimization problem of the form

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{X}}{\arg\min} \; \|A\mathbf{x} - \mathbf{y}\|_2^2 \, . \tag{3}$$

Variational methods additionally aim to incorporate prior knowledge in the form of a regularization term $R(\mathbf{x})$ to the functional, e.g.,

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{X}}{\arg\min} \; \|A\mathbf{x} - \mathbf{y}\|_2^2 + R(\mathbf{x}). \tag{4}$$

3.3. CT workflows

The CT workflows that we consider here contain input data in the form of projection images, contain several data processing steps of which at least one is a reconstruction step, and contain an objective function that scores the final result. We consider potentially non-sequential workflows, i.e., the data processing graph can contain several branches, or cyclical sections. The computational blocks can contain parameters that we want to update in order to minimize the

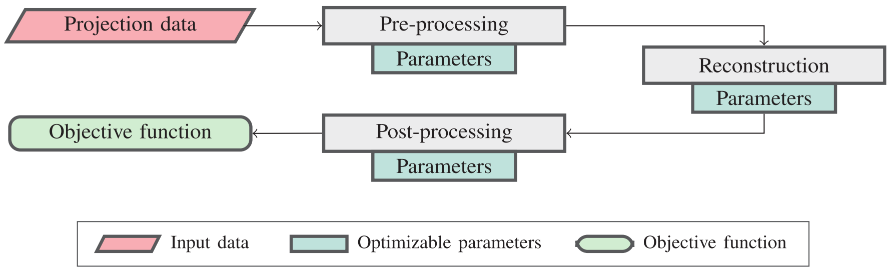


Fig. 1. Example CT workflow diagram.

objective function (see Fig. 1). Those parameters must be real or complex numbers in order to facilitate gradient-based optimization.

For the four case studies, we have implemented the computational blocks to be differentiable with respect to the learnable parameters. Non-differentiable steps can potentially be replaced by a differentiable approximation function. For example, to make thresholding (and segmentation) differentiable, we implement the thresholding operation using a hyperbolic tangent

$$\tau_\gamma(\mathbf{x}, t) = \frac{1}{2}\left(1 + \tanh\left(\gamma\left(\frac{\mathbf{x}}{t} - 1\right)\right)\right), \tag{5}$$

where the volume $\mathbf{x}$, and threshold t have been scaled to $[0, 1]$. The parameter γ regulates the sharpness of the clipping.

3.4. Software implementation

To compute gradients end-to-end for workflows that contain tomographic (back)projection operators, we will make use of the matrix identity $\nabla A\mathbf{x} = A^T \nabla \mathbf{x}$. Tomographic projection operations are implemented in the ASTRA toolbox [29] in a computationally efficient GPU accelerated manner. The tomosipo package [30] implements PyTorch support for ASTRA, and contains projection operators that propagate gradients in PyTorch using the aforementioned identity. Here, we use tomosipo projection operators in our workflows to propagate gradients end to end. In addition to reconstruction algorithms, the workflows we construct in this work contain other classical CT algorithms, such as Paganin's phase retrieval, phase contrast projection, and spectral projectors. We implemented these algorithms in PyTorch which facilitates auto-differentiation, and makes them GPU compatible, and the code is made publicly available for all case studies [31].

4. Case studies

4.1. Rotation axis alignment

4.1.1. Introduction

In the first case study, we consider the problem of alignment in tomographic reconstruction [32]. Various experimental factors can introduce errors in the geometric parameters that are used to perform the reconstruction. A common occurrence of this effect in CT is a misalignment of the rotation axis position. In the case of a rotation axis misalignment $\boldsymbol{\delta}$, for a ray parametrized by angle $\boldsymbol{\omega}$ and position vector $\boldsymbol{a}$ from scanning set Γ, the projection p of the object function $f : \mathbb{R}^3 \to \mathbb{R}$ is given by the Radon transform as

$$p(\boldsymbol{a}, \boldsymbol{\omega}) = \int_0^\infty f(\boldsymbol{a} + \boldsymbol{\delta} + t\boldsymbol{\omega})\, dt, \quad \boldsymbol{\omega} \in \mathbb{S}^2, \quad \boldsymbol{a} \in \Gamma. \tag{6}$$

This discrepancy (most prominently a lateral shift) between the assumed and the true rotation axis position introduces severe artifacts in the resulting reconstruction, which appear differently depending on the acquisition geometry. In cone-beam CT this leads to blurring and "doubling" of object features, which significantly affect the sharpness of the reconstruction.

4.1.2. Experiments

Here we propose to optimize for the rotation axis position by minimizing the magnitude of misalignment-induced artefacts as measured by an appropriate measure of reconstruction quality [33,34]. In cone-beam CT, to account for the associated edge blurring and "doubling" artifacts, we use a well-known image contrast measure based on the variance of voxel intensities [35]. The intuition behind this metric is that upon blurring the intensities of neighboring voxels average out, reducing the intensity variations throughout the reconstruction. Since variance computation is differentiable, this provides a suitable metric for gradient-based optimization.

To enable gradient-based optimization of rotation axis position, we adjust the lateral shift of all projection images using a differentiable bicubic interpolation-based image shift operator implemented in PyTorch. The shifted projections are then backprojected and the quality of this intermediate result is quantified using voxel intensity variance. Since all the pieces of the described workflow are implemented in a differentiable manner within PyTorch, a generic gradient-based optimization algorithm (SGD) can now be used to find the shift of the rotation axis that maximizes the reconstruction quality. A diagram of the proposed pipeline is shown in Fig. 3. Both experiments in this section were performed on a workstation with 64 GB RAM, an Nvidia GeForce GTX 1070 GPU, and Intel i7-7700K CPU.

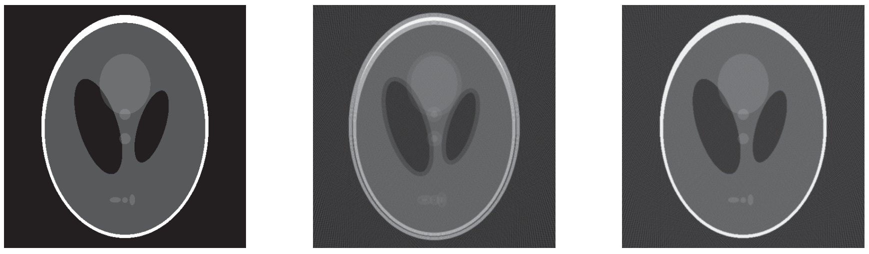
4.1.3. Simulated experiment: Shepp-Logan phantom

To test the proposed gradient-based rotation axis alignment method, we first employ simulated data based on a 512×512 px Shepp-Logan phantom (Fig. 2(a)) that is projected using a geometry with a rotation axis shift of 3 px. As can be observed in Fig. 2(b), even a minor misalignment in the axis position introduces significant blurring in the reconstruction. The proposed gradient-based contrast optimization method successfully compensates the rotation axis shift (Fig. 2(c-e)), converging in only about 3 iterations. The proposed method is computationally efficient as running 6 iterations took 97 ms on our system.

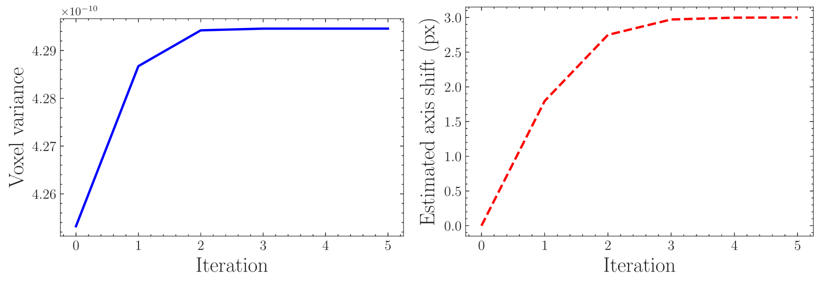
4.1.4. Real-world experiment: High-resolution cone-beam CT of a walnut

Next, we evaluate the performance of the proposed rotation axis shift compensation method on experimental data using an open dataset of high-resolution cone-beam CT of walnuts acquired at the Flex-ray lab [36,37]. Figure 4(a) demonstrates an FDK reconstruction obtained with the geometry parameters specified in the dataset metadata, and Fig. 4(b) shows the reconstruction after axis alignment method has been applied. It can be observed that although there are no obvious artifacts present in the initial reconstruction (which makes this kind of misalignment easy to miss with manual inspection), the rotation axis position optimization significantly improves the resolution of the reconstruction. The improved resolution brings up details that were not visible before, such as pores in the central part of the walnut that can be seen in the zoom-in of Fig. 4(b). The walnut slice is 550×550 px, with 501 projection angles, and running 6 iterations took 295 ms on our system.

To summarize, our workflow for rotation axis position alignment, implemented in an automatic differentiation framework, results in an intuitive formulation of the axis alignment problem. In addition, the resulting method enjoys a quick convergence, can retrieve sub-pixel axis shifts in a straightforward manner, and can be easily extended to multivariate optimization if other geometry parameters, such as rotation axis tilt and the cone angle of illumination, are included in the workflow.



(a) Shepp-Logan phantom (b) FDK (before optimization) (c) FDK (after optimization)



(d) Contrast measure during optimization (e) Rotation axis shifts during optimization

Fig. 2. (a) Shepp-Logan phantom and its FDK reconstructions obtained from projection data simulated using (b) shifted rotation axis and (c) rotation axis position compensated by the proposed gradient-based optimization method. (d) Image variance-based contrast measure and (e) estimated rotation axis shift during the optimization.

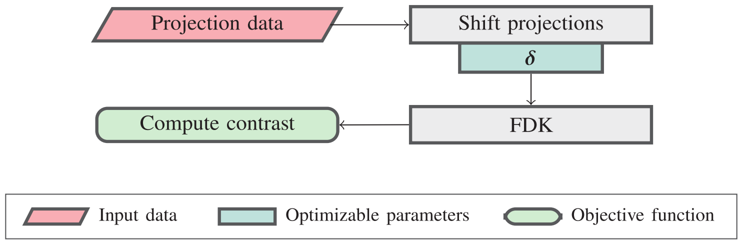


Fig. 3. CT workflow for self-supervised rotation axis alignment.

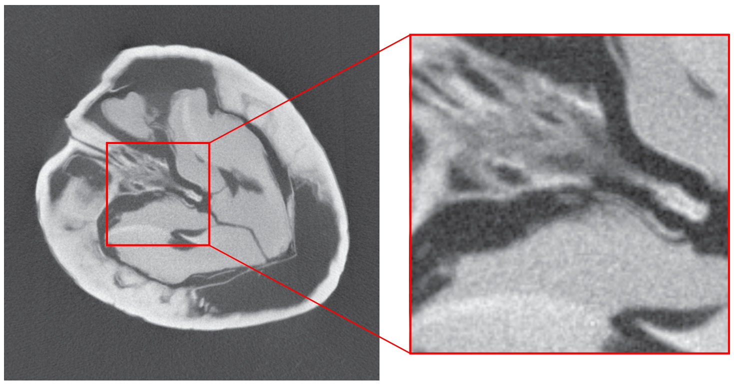
4.2. Phase retrieval

4.2.1. Introduction

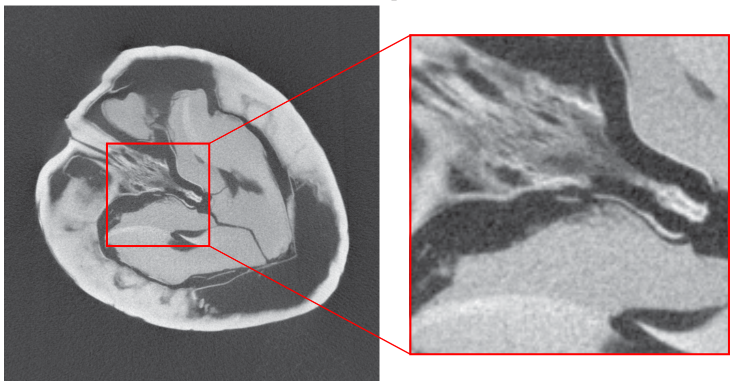
In the second case study we apply an end-to-end optimization approach to phase contrast imaging (PCI) [38]. PCI can reach nanometric resolution in tomographic imaging [39], and requires an additional reconstruction step known as phase retrieval. In PCI, the image is reconstructed based on changes to the wave front due to the material that is present along the wave path. A widely used experimental PCI setup is phase propagation-based imaging where projections are acquired from several different distances using a coherent beam. Next, a phase retrieval algorithm is used to calculate phase maps.

Here, we will consider Paganin's phase retrieval algorithm [40], which requires the material refractive index δ, and attenuation β to be known. Paganin assumes a single material object, and retrieves the projected thickness of the object $\mathbf{T}$ by

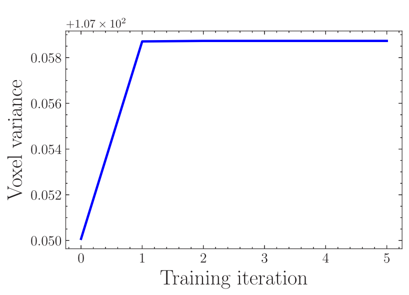
$$\mathbf{T}(\mathbf{r}_\perp) = -\frac{1}{\beta}\log\left(\mathcal{F}^{-1}\left(\frac{\beta\mathcal{F}\{I(\mathbf{r}_\perp, z = R_2)\}/I_0}{R_2\delta\,\|\mathbf{k}_\perp\| + \beta}\right)\right). \tag{7}$$



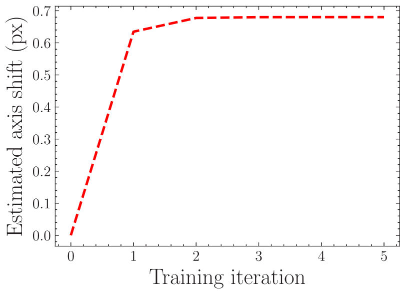
(a) FDK (before optimization)



(b) FDK (after optimization)



(c) Contrast measure during optimization



(d) Rotation axis shifts during optimization

Fig. 4. (a) FDK reconstruction of the walnut dataset. (b) FDK reconstruction after rotation axis position was compensated by the proposed gradient-based optimization method. (c) Image variance-based contrast measure and (d) estimated rotation axis shift during the optimization.

Here I is the intensity function, I_0 the incident intensity, $\mathbf{r}_\perp$ the position vector perpendicular to the optical axis, $\mathbf{k}_\perp$ the wave vector, and R_2 the source-detector distance. A common practice is to divide both the numerator and the denominator inside the inverse Fourier transform of equation 7 by β. The resulting fraction δ/β is sometimes designated as α. Expert users will often pick α manually to get a good reconstruction. However, this can be a time-consuming process, it is subjective, and it can make it more difficult for other researchers to reproduce results. Therefore, in this section we will show that we can optimize both β and δ in an unsupervised way for a clear objective. We do so by constructing a pipeline of operators that propagate gradients end-to-end. We choose to optimize β and δ separately, rather than $\alpha = \delta/\beta$, to investigate whether our unsupervised approach can recover the original β and δ values.

4.2.2. Experiments

In the experiments, we use a pipeline that takes raw projections as input, and performs Paganin phase retrieval to produce phase maps. A reconstruction based on the phase maps is made with filtered backprojection (FBP), and a binary segmentation is made with an implementation of Otsu's method [41] that makes use of equation 5. We use binary segmentation since Paganin assumes a single material.

After segmentation, using the same refraction and attenuation indices, projections are simulated based on the segmentation using a wave propagation projector. Finally, the simulated projections are scaled so that their mean and standard deviation align with the raw input. As a loss function we take the mean-squared error loss between the scaled simulated projections and the original input projections. A diagram of the pipeline is given in Fig. 5. Both experiments in this section were performed on a server with 384 GB RAM, an Nvidia Titan RTX GPU, and two Intel Xeon Gold 6130 CPUs.

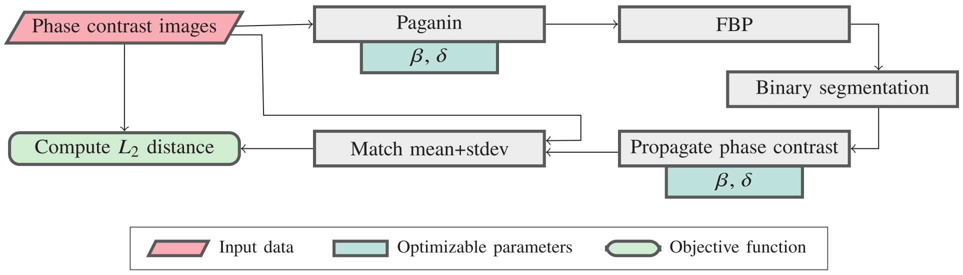


Fig. 5. PCI pipeline for self-supervised optimization of β and δ.

4.2.3. Simulated experiment: Calcium carbonate cube

First, we perform a simulated experiment on a 3D phantom (192^3 volume, 572 angles) of a calcium carbonate hollow cube, with a smaller cube attached to one of its sides (see Fig. 6(d)). Next, we simulate phase contrast images using the material parameters of calcium carbonate, and add Poisson noise. As reference, we show the FBP reconstruction of the phase maps acquired with Paganin with the correct β and δ in Fig. 6(a). We initialize the attenuation and refraction indices to those of water. We use a gradient descent algorithm (Adam) optimization algorithm to update β and δ. For a smoother optimization we optimize these parameters on an exponential scale.

The FBP reconstructions before and after optimization, and corresponding Otsu segmented images, are given in Fig. 6. We see that the initial FBP reconstruction using water material indices has a halo artifact. This artifact is no longer visible after optimization. The optimization loss and the attenuation and refraction values (logarithm) per iteration are given in Fig. 7. While

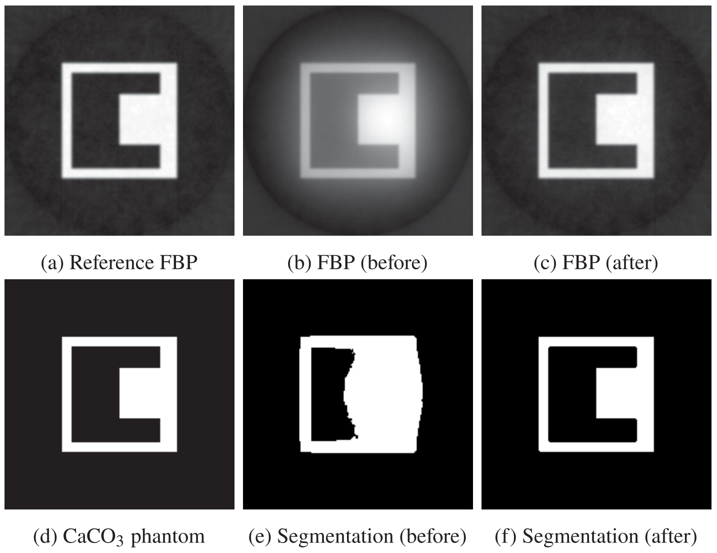


Fig. 6. (a) FBP reconstruction using correct β and δ. (b) FBP of phase projections retrieved with the material indices for water that were used as initialization. (c) FBP of phase projections retrieved with learned material parameters after optimization. (d) Slice through calcium carbonate simulated phantom. (e) Otsu segmented slice of FBP reconstruction (before). (f) Otsu segmented slice of FBP reconstruction (after).

the reconstruction quality has improved, the pipeline is not able to fully recover the original material index values; the final values after optimization are $\ln(\beta) = -21.50$ and $\ln(\delta) = -15.69$, whereas the original values for calcium carbonate are $\ln(\beta) = -19.59$, $\ln(\delta) = -13.94$. However, the value of $\alpha = \delta/\beta$ is only 17.6% off the original. Running 120 iterations of optimization on this pipeline took 16 minutes and 48 seconds on our system.

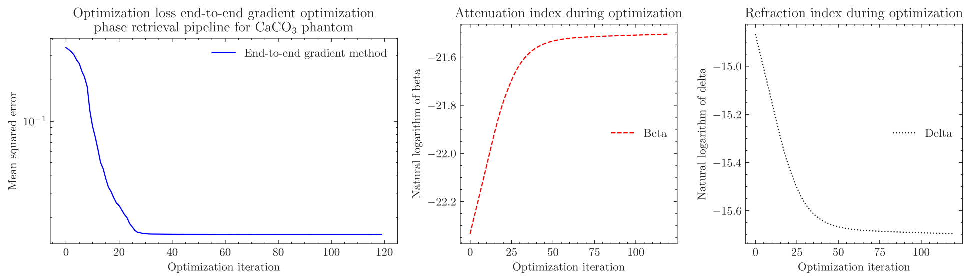


Fig. 7. (Left) L2-loss between input phase maps and output simulated phase maps per iteration. Attenuation β (middle), and refraction δ (right) value per iteration.

4.2.4. Real-world experiment: Hydrogen fuel cell

We perform a real-world data experiment on a hydrogen fuel cell dataset acquired at the TOMCAT beamline of the Swiss Light Source (PaulScherrer Institut) [42]. The experiment is performed on a central slab of the full volume of size 160 × 1001 × 1476 to reduce computation time. For the experiment we use the same setup as for the simulated experiment, and again initialize the attenuation and refraction indices to those of water. In Fig. 8 we show reconstructed central slices of the fuel cell before and after optimization. We see that after optimization small-scale features such as bubbles are visible. The zoomed images show that the initial reconstruction is blurred, and the final reconstruction after optimization is much sharper. In Fig. 9 we display the optimization loss and the attenuation and refraction values per iteration. Running 200 iterations of optimization on this pipeline took 1 hour and 38 minutes on our system.

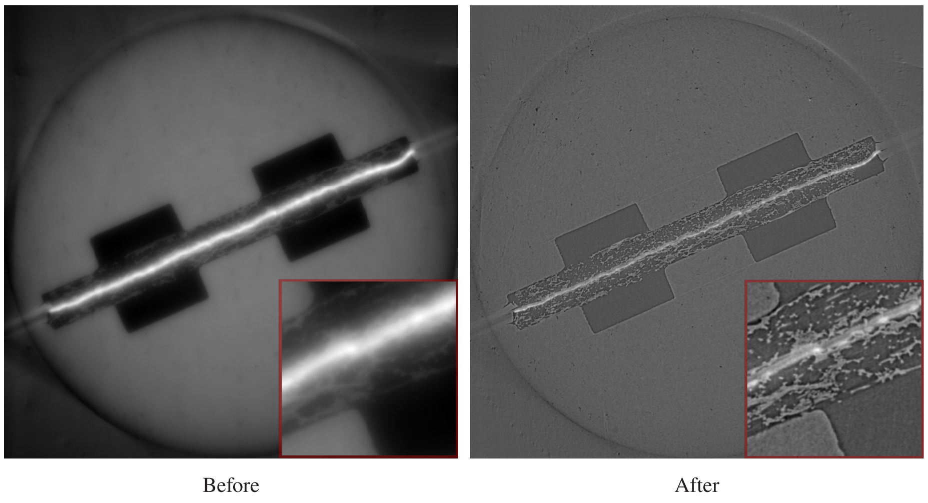


Fig. 8. Hydrogen fuel cell [42] reconstructed with FBP with default material parameters for water (before), and after optimizing parameters end-to-end (after).

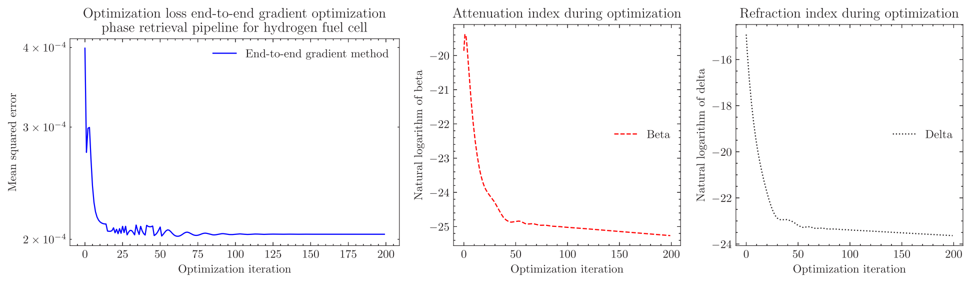


Fig. 9. (Left) L2-loss between input phase maps and output simulated phase maps per iteration. Attenuation β (middle), and refraction δ (right) value per iteration.

Overall the experiments show that we can use a generic gradient based approach to optimize for reconstruction quality in an self-supervised way in phase contrast imaging. We showed on both a simulated and real-word dataset that the pipeline can be optimized for a clear objective, as opposed to manual selecting the $\alpha = \delta/\beta$ parameter. First, this reduces the work for operators as they no longer have to tune parameters manually. Second, this makes the procedure more reproducible for different datasets.

4.3. Beam hardening correction

4.3.1. Introduction

For the third case study, we implemented self-supervised correction of artifacts introduced by beam hardening [43,44]. The number of materials is denoted by N, with attenuation coefficients $\mu_n(E)$, and the beam spectrum has E_m energy bins with intensities I_e. Furthermore, let $l_{i,j}$ be the intersection length of ray i ($i = 1 \ldots D$) with voxel j ($j = 1 \ldots J$), d_j the relative density of voxel j, and $s_{n,j}$ a variable that is 1 if voxel j contains material n, and 0 otherwise. Then we can denote for a ray i and a material n the projected relative density $P_{i,n}$, and the monochromatic measured intensity by Beer-Lambert as

$$P_{i,n} = \sum_{j=1}^{J} l_{i,j} d_j s_{n,j}, I_{\text{mono},i} = I_0 e^{-\sum_{n=1}^{N} \mu_n(E_0) P_{i,n}}, \tag{8}$$

for a monochromatic beam with energy E_0 and intensity I_0. The measured polychromatic intensity (discrete) is given by

$$I_{\text{poly},i} = \sum_{e=1}^{E_m} I_e e^{-\sum_{n=1}^{N} \mu_{n,e} P_{i,n}}, \tag{9}$$

for a polychromatic beam with energies e and intensities I_e.

For a monochromatic X-ray, the attenuation coefficient μ is linearly related to the thickness of the object by Beer-Lambert's law. However, in practice X-ray beams are often polychromatic and this relation is no longer linear as lower energy photons get absorbed more than higher energy photons. Therefore, as a polychromatic beam travels through an object it "hardens", i.e., the average photon energy increases. If this effect is not taken into account by the reconstruction algorithm, it causes streaking and cupping artifacts because the lower absorbance due to higher average energy is mistakenly reconstructed as a lower density material.

4.3.2. Experiments

The aim of the case study is to highlight how combining an explicit forward model with a few lines of code and generic gradient-optimization can result in sophisticated algorithm construction, even for cases that used to require lengthy, hand-crafted implementations. In the experiments we will perform unsupervised beam hardening correction by learning the beam spectrum, and the energy-dependent attenuation coefficients per material. We will base the study on [45], where three different iterative unsupervised beam hardening correction algorithms are proposed and compared. We use the best performing algorithm in [45] as a comparison; the *iterative sinogram preprocessing* (ISP) method. The ISP method is an iterative scheme with multiple steps, such as a bruteforce segmentation step, a local optimization step, and a reconstruction update step. For a detailed explanation of ISP we refer to [45]. ISP assumes that the number of materials is known beforehand, but no knowledge of the beam spectrum, or attenuation coefficients is assumed.

As opposed to constructing a specialized algorithm such as ISP, we compare to a generic gradient-based approach where beam spectrum, attenuation coefficients, and thresholds are learned jointly. The generic approach is self-supervised using the same loss function as ISP

$$\varphi(\mu, \mathbf{I}, \mathbf{s}, \mathbf{d}) = \frac{1}{D} \sum_{i=1}^{D} \left(\log\left(\frac{I_{\text{poly},i}^{\text{meas}}}{I_0} \right) - \log\left(I_{\text{poly},i}^{\text{sim}} \right) \right)^2, \tag{10}$$

which is the L_2-loss between the simulated polychromatic projections and the original input projections. To initialize the thresholds we use Otsu's method on an initial reconstruction R_0 made with FBP or FDK. After the initialization we iterate as

1. Update segmentation thresholds, attenuation coefficients, and beam intensities,

$$\mathbf{s}^k, \mu^k, \mathbf{I}^k = \arg\min_{\mathbf{s},\mu,\mathbf{I}} \varphi\big(\mu, \mathbf{I}, g(R^{k-1}, \mathbf{s}), \mathbf{d} = 1\big), \tag{11}$$

   where $g(R^{k-1}, \mathbf{s})$ is the differentiable thresholding function described before (equation 5), φ is the ISP loss function, and arg min is performed with gradient descent.

2. Update the sinograms and corrected reconstruction R^k as for ISP.

A diagram of the workflow is shown in Fig. 10. By using an implementation of the spectral projector that can propagate gradients, we can optimize the attenuation, intensity, and thresholds jointly. This improves the complexity of the algorithm as the thresholding step in ISP is exponential in the number of materials and computationally expensive. For our experiments, we created a PyTorch based implementation of ISP and our generic gradient-based approach. Both experiments in this section were performed on a workstation with 64 GB RAM, an Nvidia RTX 2070 Super GPU, and AMD Ryzen 7 3800X CPU.

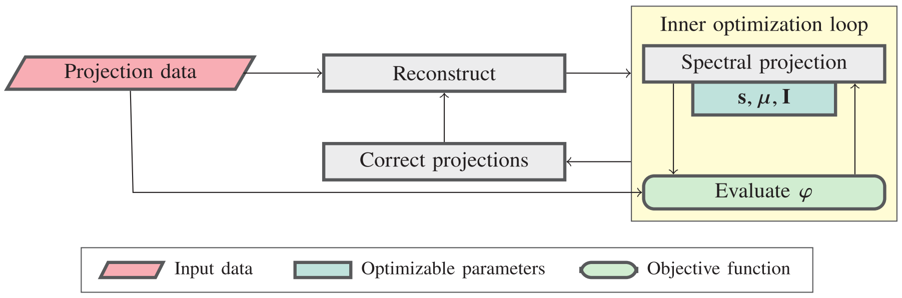


Fig. 10. CT pipeline for self-supervised beam hardening correction.

4.3.3. Simulated experiment

We perform a simulated experiment based on the Barbapapa phantom [45] where we created a simulated phantom (see Fig. 11) of polymethyl methacrylate (PMMA) filled with aluminium rods with size 256×256. Next, we use a spectral projector with a simulated effective beam spectrum to simulate beam hardening artifacts. Finally, we add 2% Gaussian noise on the projections (see Fig. 11 for reconstruction of noisy projections). To perform beam hardening correction we performed 40 steps of ISP. We ran the generic gradient-based approach for the same amount of objective function evaluations. In Fig. 12 we show the FBP reconstructions of the corrected sinograms for both algorithms, and the accompanying material segmentations. We see that both methods reduced cupping and streaking artifacts. Furthermore, Figs. 12(d) to 12(f) show a significant improvement in segmentation quality for both methods, which is close to the original phantom.

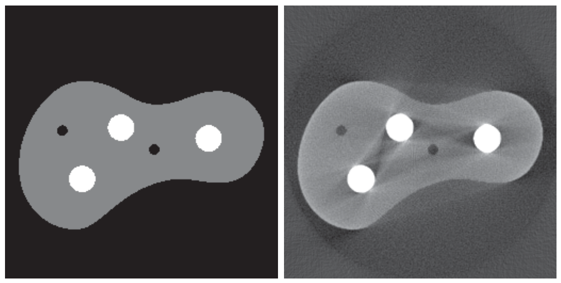


Fig. 11. (Left) Barbapapa phantom consisting of PMMA filled with aluminium cylinders. (Right) FBP reconstructions of simulated spectral projections (with added Gaussian noise).

Line profiles for each of the three reconstructions are given in Fig. 13(a). The line profiles confirm that the cupping artifacts are significantly reduced for both the PMMA material and the aluminium rods. In Fig. 13(b) we plot the self-supervised loss for both methods. Both methods reach a similar optimum, but the combined gradient method shows slightly faster convergence.

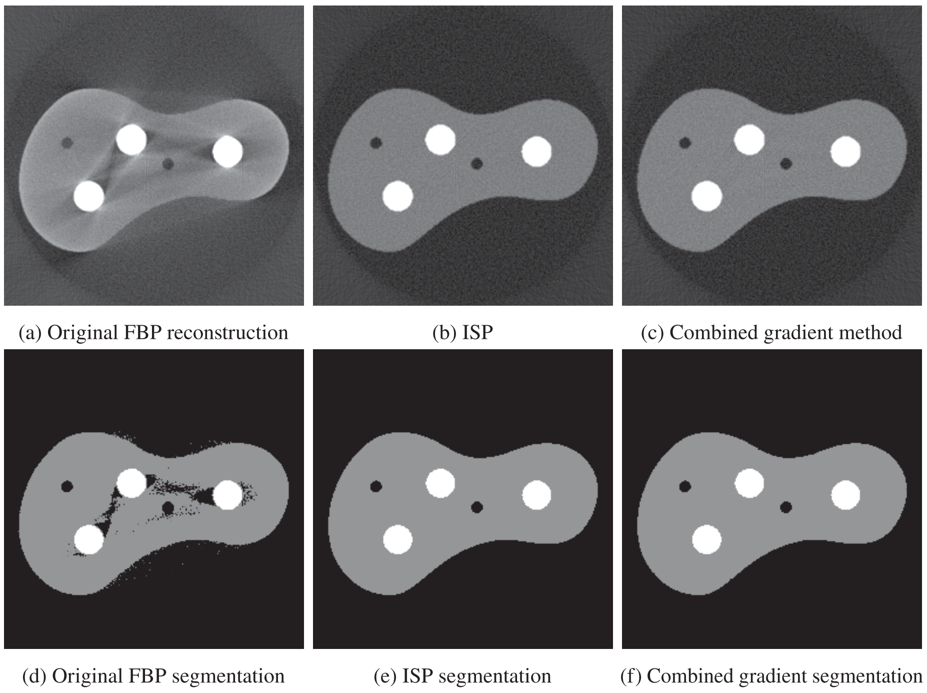

(a) Original FBP reconstruction
(b) ISP
(c) Combined gradient method
(d) Original FBP segmentation
(e) ISP segmentation
(f) Combined gradient segmentation

Fig. 12. (Top) Barbapapa phantom reconstructed with the Filtered BackProjection (FBP) algorithm for ISP and a generic gradient-based approach. (Bottom) Material segmentations of corrected reconstructions.

Note that the combined gradient method also runs twice as fast, which is due to the missing brute-force threshold selection step. This is also reflected in the runtimes; ISP ran for 16 minutes and 15 seconds, whereas the generic gradient-based approach ran for 8 minutes and 9 seconds.

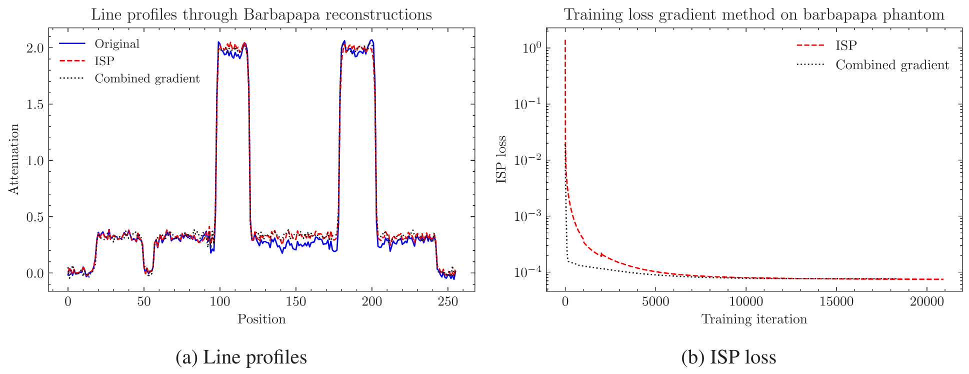
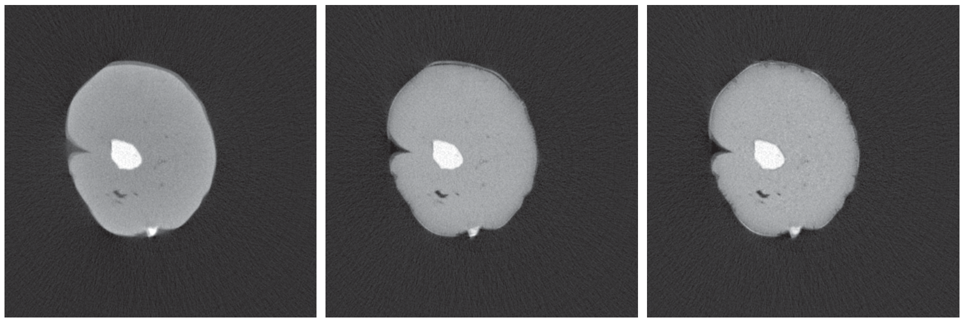


Fig. 13. (a) Three line profiles through the original, ISP corrected, and combined gradient corrected Barbapapa reconstructions. (b) Optimization losses per φ evaluation.

4.3.4. Play-Doh foreign object X-ray CT dataset

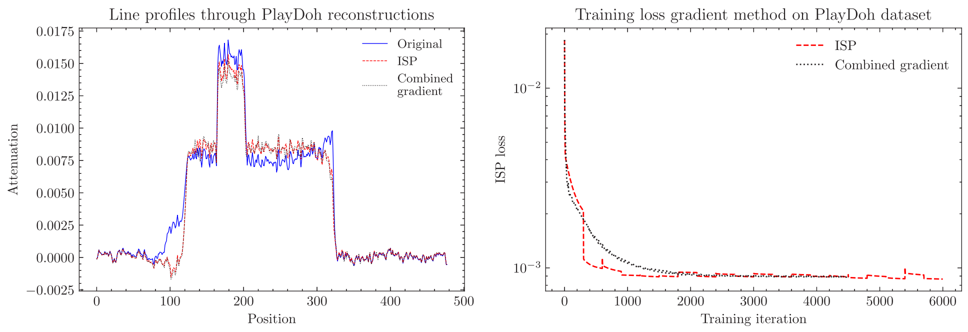
We also evaluate both beam hardening correction methods on a real-world X-ray CT dataset of Play-Doh objects filled with stones [46,47]. The experiment was performed on a 478×478 central slice. The Play-Doh objects exhibit significant cupping artifacts. In Fig. 14 we show the

FBP reconstructions of the corrected sinograms for both algorithms. We see that both methods reduced cupping, which is also clearly visible on the line profiles shown in Fig. 15(a). In Fig. 15(b) we plot the self-supervised loss for both methods. Both methods reach a similar optimum, but the combined gradient method shows a more stable convergence. The jumps in the loss curve for ISP happen when a new optimal set of thresholds is determined with brute-force calculation after the local minimization steps. Since the combined gradient method jointly optimizes all parameters every step, it does not suffer from such jumps. The improved computational efficiency is more prominent due to faster convergence of the local optimization step; ISP ran for 19 minutes and 51 seconds, whereas the generic gradient-based approach ran for 4 minutes and 35 seconds.



(a) Original FBP reconstruction (b) ISP (c) Combined gradient method

Fig. 14. Play-Doh and stones reconstructed with the Filtered BackProjection (FBP) algorithm for ISP and a generic gradient-based approach.



(a) Line profiles (b) ISP loss

Fig. 15. (a) Three line profiles through the original, ISP corrected, and combined gradient corrected Play-Doh reconstructions. (b) Optimization losses per φ evaluation.

Overall the experiments show that by using a generic gradient-based approach for all parameters, we can create an algorithm that performs similarly to the specialized ISP algorithm while demonstrating faster runtime. The construction of such an algorithm is more straightforward as we can create the forward model and embed the workflow in an auto-differentiation framework. This allows us to optimize end-to-end in a straight-forward manner.

4.3.5. Balancing computation speed and memory consumption

The differentiable spectral projection function used in this case study is a memory-intensive and computationally expensive operation. This can limit its applicability to realistic sizes of data if a naive implementation is used. In this work, we employ two optimization techniques to tackle this problem: gradient checkpointing and just-in-time (JIT) compilation (see section 3). In this section we analyze how both of these techniques influence the computation speed and memory usage for different sizes of input data.

The spectral projection operation described by equation 9 is a loop over the energy bins e that adds each term to the result to compute I_{poly}. When differentiating this function via naive back propagation, every intermediate term of size $N_\theta \times N_u \times N_v$ in the loop needs to be saved for the backward pass, leading to a high memory consumption. This effect can be alleviated by placing gradient checkpoints after every few iterations, which means that some intermediate steps are recomputed during the backward pass instead of being stored in memory. The placing of gradient checkpoints can be optimized depending on the desired trade-off between the memory requirements and computational efficiency, which can be done optimally using several approaches, for example tree decomposition [48]. In the present case study we were limited primarily by the amount of available GPU memory, and therefore checkpoints were placed after every loop iteration. JIT compilation can be applied to the function that computes each term in the sum, potentially improving both memory and computational efficiency – the overhead of compilation in this case is justified by the much longer runtime of the executed function. In our implementation we used the TorchScript JIT compiler available in PyTorch.

Whether gradient checkpointing and JIT compilation are beneficial depends on the size of the input data $N_\theta \times N_u \times N_v$ and the number of energy bins E_m, since for small data sizes we are potentially losing time on associated overheads with little memory gain. To illustrate the potential benefits of both approaches in this profiling experiment we use a slab of $N_u = 48$ slices and vary E_m. In Fig. 16 we show the wall time (in seconds), and GPU memory consumption (in gigabytes) for different E_m measured for a single optimization step of our combined gradient method.

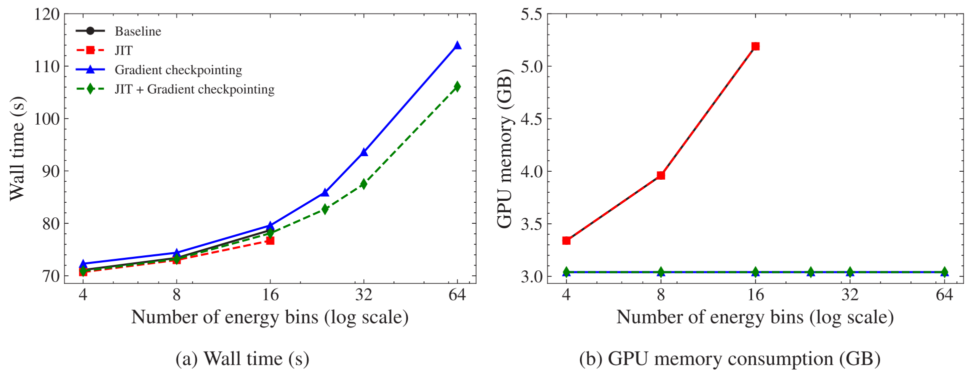


Fig. 16. (a) Wall time (in seconds) and (b) GPU memory consumption (in gigabytes) for a single optimization step of the combined gradient method for different code optimizations used; baseline, JIT compilation, gradient checkpointing, and JIT + gradient checkpointing. Metrics are shown for different numbers of energy bins E_m.

The metrics show that without gradient checkpointing, both the baseline and JIT-compiled method run out of the available GPU memory (8 GB for Nvidia RTX 2070 Super used here) when the number of energy bins is higher than 16. On the other hand, memory consumption is constant irrespective of the number of energy bins when gradient checkpointing is used, while the overhead of using gradient checkpointing was approximately constant at about 1 s. Applying JIT compilation in this experiment did not reduce memory consumption, but did improve computational efficiency, providing about 0.5 s of speedup over baseline for small numbers of energy bins. For the larger numbers of energy bins, when using JIT compilation in combination with gradient checkpointing, the speedup was much higher. This is likely because the JIT compiled code is run once for every energy bin, spreading out the compilation overhead and multiplying the computational gains over a higher number of repeated computations. In this case study, we chose to use the combination of JIT and gradient checkpointing because it allowed for a significantly reduced memory consumption while maintaining a comparable computational cost to the baseline method.

The discussed profiling experiment shows that techniques such as gradient checkpointing and JIT compilation can enable GPU-accelerated optimization based on auto-differentiation for input data sizes that are significantly larger compared to what is possible with a naive back propagation algorithm. However, the effectiveness and usefulness of such techniques will depend on the details of computation and therefore needs to be analyzed on the use case basis.

4.4. Optimizing total variation reconstruction and neural networks end-to-end

4.4.1. Introduction

In the final case study we will focus on denoising CT reconstructions using convolutional neural networks (CNNs) jointly with total variation reconstruction (TV) [49–51]. Since neural networks are often large, black-box models that are hard to interpret, it can be desirable to let more computation steps be performed by interpretable algorithms and use a smaller network, rather than using simpler algorithms and a larger neural network. In the experiment we show how embedding CT algorithms in an auto-differentiation framework allows for easy interfacing with deep learning algorithms, and the pipeline can be trained end-to-end. This method allows us to design a more interpretable pipeline using variational methods with similar accuracy, while using a smaller neural network.

Total variation reconstruction is a commonly used variational method to incorporate prior knowledge that the gradient of the image should be sparse. For TV, the regularization term in equation 4 becomes the magnitude of the gradient

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \left\{ \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{x}\|_{\mathrm{TV}} \right\} = \arg\min_{x} \left\{ \|A\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \|\nabla \mathbf{x}\|_1 \right\}. \tag{12}$$

To minimize the functional we use an implementation based on Chambolle-Pock [52]. The regularization parameter λ controls the trade-off between data fidelity and regularization term. A small λ will result in a reconstruction $\mathbf{x}^*$ that is close to the raw data, but potentially with high noise. A large λ can lead to less noise and more connected components of equal value. For more information on total variation regularization we refer to Rudin-Osher-Fatemi [53].

4.4.2. Experiments

We perform an experiment on simulated 2D foam-like phantoms of size 256×256. These phantoms contain both large and small scale features (see Fig. 17(a)). We simulate parallel beam projections from an angular range of -60° to 60°, creating missing wedge artifacts, and add Poisson noise to the projection data (see Fig. 17(b) for an example FBP reconstruction). For TV reconstruction (Figs. 17(c) and (d)), this creates a situation where a small λ keeps smaller features, but creates a high noise reconstruction, while a larger λ removes more noise but creates connected components of voxels which removes smaller features. We created a training dataset of 100 randomly generated target phantoms, and corresponding noisy limited angle projections. In our experiments we in each case have the noisy projections as input data, and use the original phantom as target data. Our implementation of total variation reconstruction uses PyTorch primitives, and λ can therefore be optimized end-to-end with gradient-based approaches in conjunction with deep learning algorithms. In total, we trained four pipelines end-to-end.

1. **FBP + Small CNN:** An FBP reconstruction is input for a small CNN that denoises the image. The CNN consists of 3 layers of 3×3 kernels arranged in 1×64, 64×32, and 32×1 channels (19.393 parameters in total). Only the CNN weights are learned.

2. **Single TV + Small CNN:** A single TV reconstruction is input for a small CNN that denoises the image. The CNN consists of 3 layers of 3×3 kernels arranged in 1×64, 64×32, and 32×1 channels (19.393 parameters in total). Both λ and the CNN weights are learned jointly. λ is initialized at $\lambda = 10^{-3}$.

3. **Double TV + Small CNN:** Two TV reconstructions with different λ are input for a small CNN that uses both inputs to create a single denoised output image. The CNN consists of 3 layers of 3×3 kernels arranged in 2×64, 64×32, and 32×1 channels (19.969 parameters in total). Both λ_1, λ_2, and the CNN weights are learned jointly. The λ's are initialized as $\lambda_1 = 10^{-3}$, and $\lambda_2 = 10^{-8}$.

4. **FBP + Large CNN:** An FBP reconstruction is input for a larger CNN that densoises the image. The CNN consists of 4 layers of 3×3 kernels arranged in 1×160, 160×96, 96×64, and 64×1 channels (195.873 parameters in total). Only the CNN weights are learned.

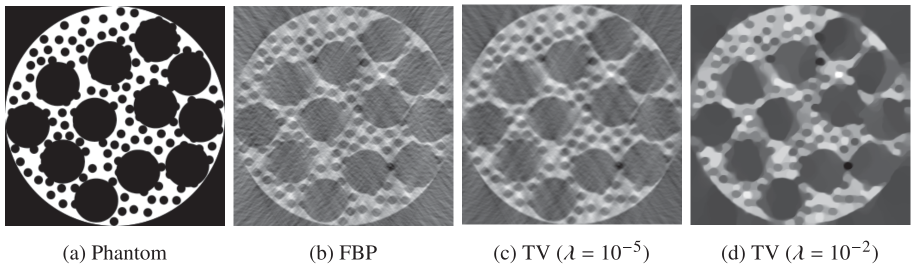


Fig. 17. (a) Foam CT binary phantom, (b) FBP reconstruction of phantom projections with added severe noise, (c) TV reconstruction with $\lambda = 10^{-8}$, and (d) TV reconstruction with $\lambda = 10^{-2}$.

A diagram of the Double TV + Small CNN pipeline is shown in Fig. 18. All 4 experiments are run for an equal number of training iterations, and were performed on a workstation with 64 GB RAM, an Nvidia RTX 2070 Super GPU, and AMD Ryzen 7 3800X CPU. For validating the results we generated an additional random phantom that was not in the training set. The resulting denoised validation reconstructions are shown in Fig. 19.

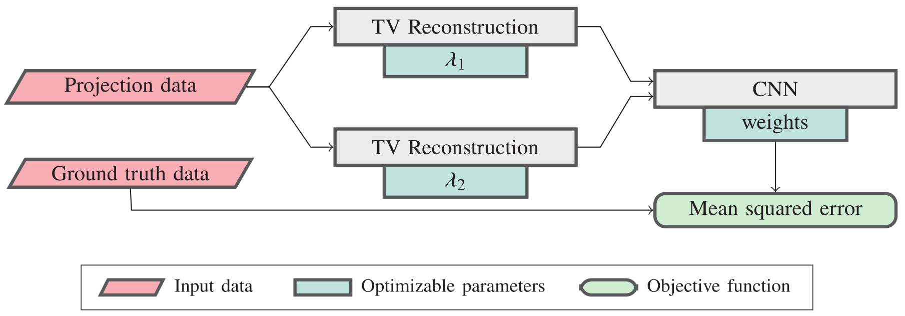


Fig. 18. Pipeline for supervised denoising of CT data.

We see that all pipelines can denoise the reconstruction significantly, but struggle to remove artifacts introduced by the missing angular information. The FBP+SmallCNN pipeline seemingly performs worse on visual inspection of the zoomed images. Arguably, both TV pipelines perform better than FBP+LargeCNN, even though the larger CNN had 10 times more parameters. This suggests that the prior knowledge incorporated by total variation makes for an easier denoising

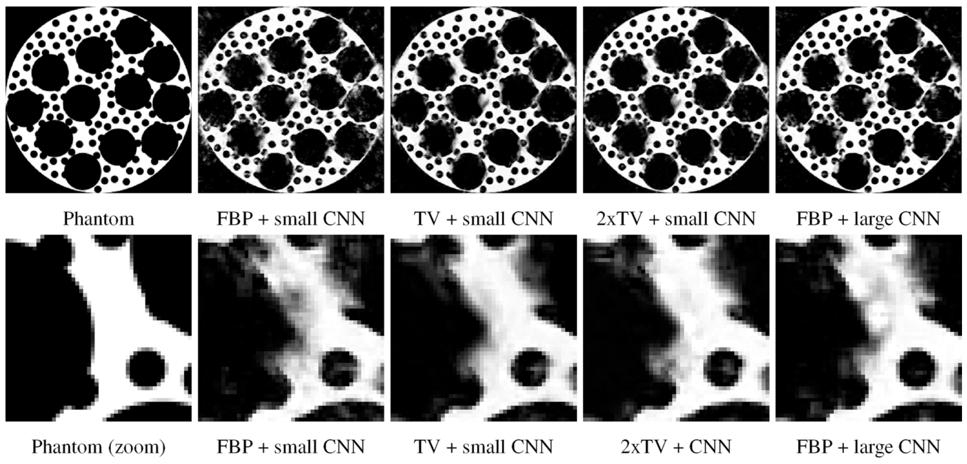


Fig. 19. (Top) From left to right; Foam phantom, output of FBP reconstruction followed by a small CNN, output of a single total variation reconstruction followed by a CNN, output of two total variation reconstructions followed by a CNN, output of FBP reconstruction followed by a larger CNN. (Bottom) Zoom of top row.

problem for the CNN. It is unclear whether the addition of a second total variation operator benefited the denoising quality. However, in validation loss the double total variation operator performs better with $1.805 \cdot 10^{-2}$ loss versus $1.864 \cdot 10^{-2}$ for the single TV operator. The validation loss for the FBP+SmallCNN pipeline is $2.322 \cdot 10^{-2}$, and for the FBP+LargeCNN pipeline $2.268 \cdot 10^{-2}$.

We plot the training losses and λ values during training in Fig. 20. An interesting observation is that single TV learned a λ with a value in between λ_1 and λ_2 of the double TV pipeline. We hypothesize that the single TV pipeline had to compromise λ between leaving small features intact, and denoising the reconstruction.

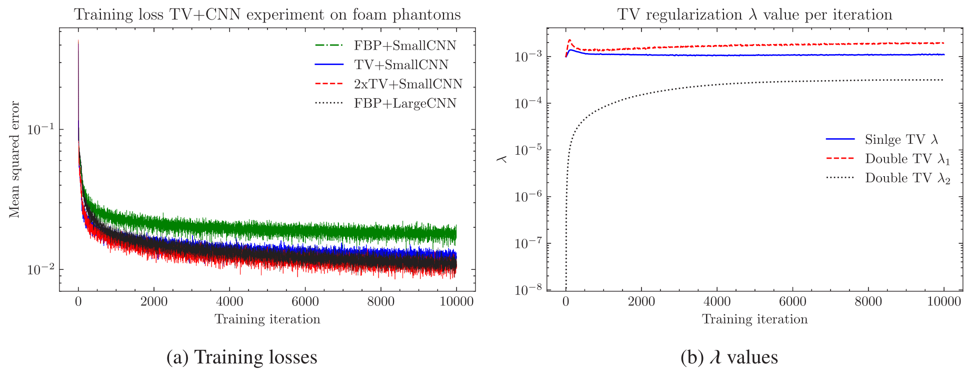


Fig. 20. (a) Mean squared error loss during training. (b) TV regularization λ values during training.

Running total variation in a training procedure comes at large computational cost; a single training step (100 images) took 99.6, and 198.3 seconds for single and double TV respectively whereas FBP + large CNN took 1.82 seconds. For 3D it could therefore be infeasible to train this pipeline. For single scan tuning of λ a surrogate TV approach can be considered during optimization, such as [54]. Alternatively, λ could be tuned on the central slice for 3D cases.

Overall the experiment shows how embedding classical CT algorithms, such as TV reconstruction, in GPU accelerated auto-differentiation frameworks allows for the easy prototyping of mixed classical and deep learning pipelines. We were able to replace a large CNN with 200 thousand parameters by a stack of two TV operators to achieve better denoising accuracy, and reduce the CNN size to 20 thousand parameters. In addition, the total variation pipelines are more interpretable as the behaviour of TV for small or large λ is well understood. In general, this easy interfacing of deep learning and classical methods enables end-to-end learning of parameters, which opens up new areas of research. Additionally, the resulting pipeline may be more interpretable since parameters of classical algorithms are often linked to physical or mathematical concepts that are better understood.

5. Discussion

Our four use cases and the corresponding experiments demonstrate that a broad range of CT workflows can be implemented as end-to-end optimized pipelines using auto-differentiation. Depending on the particular use case, implementing CT workflows in an end-to-end optimizable manner yields several benefits. When all pieces of a data processing pipeline facilitate optimization, parameters at all steps of the pipeline can be optimized jointly for a criterion calculated at any given step. In combination with explicit quality criteria this allowed us to design workflows where the learnable parameters were used in the earlier stages of the pipeline, while the objective function was more naturally defined at the end of the pipeline. For example, in both the rotation axis alignment and beam hardening correction experiments we were able to optimize parameters that are used in the projection domain for metrics that require volume domain computation. Another benefit is that implementing CT workflows as end-to-end optimizable pipelines allows for efficient automatic optimization of parameters that may otherwise be chosen manually. This additionally improves the transferability of CT workflows when applied to new data as the parameters can be optimized in an objective manner using the same quality criterion.

Auto-differentiation made it possible to implement workflows with a relatively low development cost by using existing building blocks, and by defining the workflow in a declarative manner, i.e., implementing the forward model and then optimizing its parameters with a generic off-the-shelf optimizer. Creating workflows in this manner is typically less time consuming to develop and more flexible compared to specialized methods. For example, in the beam hardening experiment we showed that a gradient descent on the forward model of the physical effect results in a quality comparable to a specialized correction method. Using auto-differentiation allowed for seamless compatibility between classical and deep learning-based approaches. As classical approaches often come with an intuition of their behaviour, this combination of classical algorithms and deep learning can lead to more robust and interpretable workflows. In the last experiment we showed that using existing classical algorithms in conjunction with deep learning can create workflows that perform similarly to purely deep learning based approaches while using smaller neural networks.

Even though gradient-based end-to-end optimization has several potential benefits, it can create certain practical challenges. First, convergence of a gradient descent optimizer is not always guaranteed, and can potentially produce suboptimal solutions. Second, the learning rate needs to be picked manually, which is especially challenging when the involved parameters have significantly different magnitudes. Despite the disadvantages, this approach often represents a well-performing heuristic solution to many mathematically complicated problems, as illustrated by its use in the field of deep learning, which produced transformative results in many branches of science and technology in the recent years.

6. Conclusion

We have shown how implementing classical CT algorithms in an auto-differentiation framework can improve their transferability and interpretability, enable efficient automatic end-to-end optimization, and allow for solving real-world problems without having to develop specialized algorithms. We have explored four use cases experimentally: rotation axis alignment, phase contrast imaging, beam hardening correction, and end-to-end denoising with deep learning and total variation reconstruction, demonstrating that a wide range of CT workflows can be implemented in such a framework. In the future, the key benefits demonstrated in this paper can be utilized in computational toolboxes that leverage auto-differentiation for improved construction and execution of advanced CT workflows.

Funding. Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWA.1160.18.316).

Disclosures. The authors declare no conflicts of interest.

Data availability. Data underlying the results presented in this paper are available in [31] with code to reproduce each experiment. Links and installation instructions to external data are provided in the notebooks; [37] used in section 4.1, [42] used in section 4.2, and [46,47] used in section 4.3.

References

1. C. McLeavy, M. Chunara, R. Gravell, A. Rauf, A. Cushnie, C. S. Talbot, and R. Hawkins, "The future of CT: deep learning reconstruction," Clin. Radiol. **76**(2), 407–415 (2021).
2. Y. Zhang and H. Yu, "Convolutional neural network based metal artifact reduction in X-ray computed tomography," IEEE Trans. Med. Imaging **37**(6), 1370–1381 (2018).
3. A. Ziabari, S. Venkatakrishnan, M. Kirka, P. Brackman, R. Dehoff, P. Bingham, and V. Paquit, "Beam hardening artifact reduction in X-ray CT reconstruction of 3D printed metal parts leveraging deep learning and CAD models," in *ASME International Mechanical Engineering Congress and Exposition* (American Society of Mechanical Engineers, 2020), vol. 2B: Advanced Manufacturing.
4. K.-L. Hua, C.-H. Hsu, S. C. Hidayati, W.-H. Cheng, and Y.-J. Chen, "Computer-aided classification of lung nodules on computed tomography images via deep learning technique," OncoTargets Ther. **8**, 2015–2022 (2015).
5. T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. Rajendra Acharya, "Automated detection of COVID-19 cases using deep neural networks with X-ray images," Comput. Biol. Med. **121**, 103792 (2020).
6. J. Chen, L. Wu, J. Zhang, *et al.*, "Deep learning-based model for detecting 2019 novel coronavirus pneumonia on high-resolution computed tomography," Sci. Rep. **10**(1), 1–11 (2020).
7. S. Bhadra, V. A. Kelkar, F. J. Brooks, and M. A. Anastasio, "On hallucinations in tomographic image reconstruction," IEEE Trans. Med. Imaging **40**(11), 3249–3260 (2021).
8. A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation* (SIAM, 2008).
9. L. Pineda, T. Fan, M. Monge, *et al.*, "Theseus: A library for differentiable nonlinear optimization," Advances in Neural Information Processing Systems 35, 3801–3818 (2022).
10. S. Cheng, M. Kim, L. Song, Z. Wu, S. Wang, and N. Hovakimyan, "Difftune: Auto-tuning through auto-differentiation," arXiv, arXiv:2209.10021 (2022).
11. Z. Huang, H. Liu, J. Wu, and C. Lv, "Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving," IEEE Trans. Neural Networks Learn. Syst. (2023).
12. J.-E. Campagne, F. Lanusse, J. Zuntz, A. Boucaud, S. Casas, M. Karamanis, D. Kirkby, D. Lanzieri, Y. Li, and A. Peel, "JAX-COSMO: An end-to-end differentiable and GPU accelerated cosmology library," arXiv, arXiv:2302.05163 (2023).
13. A. K. Maier, C. Syben, B. Stimpel, T. Würfl, M. Hoffmann, F. Schebesch, W. Fu, L. Mill, L. Kling, and S. Christiansen, "Learning with known operators reduces maximum error bounds," Nat Mach Intell **1**(8), 373–380 (2019).
14. G. Pilikos, L. Horchens, K. J. Batenburg, T. van Leeuwen, and F. Lucka, "Fast ultrasonic imaging using end-to-end deep learning," in *International Ultrasonics Symposium* (IEEE, 2020), pp. 1–4.
15. G. Pilikos, C. L. de Korte, T. van Leeuwen, and F. Lucka, "Single plane-wave imaging using physics-based deep learning," in *International Ultrasonics Symposium* (IEEE, 2021), pp. 1–4.
16. M. Thies, F. Wagner, Y. Huang, *et al.*, "Calibration by differentiation–self-supervised calibration for X-ray microscopy using a differentiable cone-beam reconstruction operator," J. Microsc. **287**, 81–92 (2022).
17. A. S. Jurling and J. R. Fienup, "Applications of algorithmic differentiation to phase retrieval algorithms," J. Opt. Soc. Am. A **31**(7), 1348–1359 (2014).
18. Y. S. Nashed, T. Peterka, J. Deng, and C. Jacobsen, "Distributed automatic differentiation for ptychography," Procedia Computer Science **108**, 404–414 (2017).
19. S. Kandel, S. Maddali, M. Allain, S. O. Hruszkewycz, C. Jacobsen, and Y. S. G. Nashed, "Using automatic differentiation as a general framework for ptychographic reconstruction," Opt. Express **27**(13), 18653–18672 (2019).

20. M. Du, Y. S. G. Nashed, S. Kandel, D. Gürsoy, and C. Jacobsen, "Three dimensions, two microscopes, one code: Automatic differentiation for X-ray nanotomography beyond the depth of focus limit," Sci. Adv. **6**, 1 (2020).
21. F. Guzzi, A. Gianoncelli, F. Billé, S. Carrato, and G. Kourousias, "Automatic differentiation for inverse problems in X-ray imaging and microscopy," Life **13**(3), 1 (2023).
22. M. Du, S. Kandel, J. Deng, X. Huang, A. Demortiere, T. T. Nguyen, R. Tucoulou, V. De Andrade, Q. Jin, and C. Jacobsen, "Adorym: A multi-platform generic X-ray image reconstruction framework based on automatic differentiation," Opt. Express **29**(7), 10000–10035 (2021).
23. A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Adv. Neural Inf. Process. Syst.* **32** (2019).
24. Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," in *Doklady Akademii Nauk* (Russian Academy of Sciences, 1983), vol. 269, pp. 543–547.
25. D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations* (2014).
26. T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," arXiv, arXiv:1604.06174 (2016).
27. A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging* (SIAM, 2001).
28. L. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone-beam algorithm," J. Opt. Soc. Am. A **1**(6), 612–619 (1984).
29. W. van Aarle, W. J. Palenstijn, J. Cant, E. Janssens, F. Bleichrodt, A. Dabravolski, J. D. Beenhouwer, K. J. Batenburg, and J. Sijbers, "Fast and flexible X-ray tomography using the ASTRA toolbox," Opt. Express **24**(22), 25129–25147 (2016).
30. A. A. Hendriksen, D. Schut, W. J. Palenstijn, N. Viganó, J. Kim, D. M. Pelt, T. Van Leeuwen, and K. J. Batenburg, "Tomosipo: fast, flexible, and convenient 3D tomography for complex scanning geometries in Python," Opt. Express **29**(24), 40494–40513 (2021).
31. R. A. Schoonhoven, "Optimizing CT workflows with auto-differentiation 2023 paper," Github (2023), https://github.com/schoonhovenrichard/AutodiffCTWorkflows.
32. T. Van Leeuwen, S. Maretzke, and K. J. Batenburg, "Automatic alignment for three-dimensional tomographic reconstruction," Inverse Problems **34**(2), 024004 (2018).
33. T. Donath, F. Beckmann, and A. Schreyer, "Automated determination of the center of rotation in tomography data," J. Opt. Soc. Am. A **23**(5), 1048–1057 (2006).
34. D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen, "Tomopy: a framework for the analysis of synchrotron tomographic data," J. Synchrotron Radiat. **21**(5), 1188–1193 (2014).
35. F. C. Groen, I. T. Young, and G. Ligthart, "A comparison of different focus functions for use in autofocus algorithms," Cytometry **6**, 81–91 (1985).
36. S. B. Coban, F. Lucka, W. J. Palenstijn, D. Van Loo, and K. J. Batenburg, "Explorative imaging and its implementation at the FleX-ray Laboratory," J. Imaging **6**(4), 18 (2020).
37. M. J. Lagerwerf, S. B. Coban, and K. J. Batenburg, "High-resolution cone-beam scan of twenty-one walnuts with two dosage levels," Zenodo (2020), https://zenodo.org/doi/10.5281/zenodo.3763411.
38. M. Endrizzi, "X-ray phase-contrast imaging," Nucl. Instrum. Methods Phys. Res., Sect. A **878**, 88–98 (2018).
39. P. J. Withers, "X-ray nanotomography," Mater. Today **10**(12), 26–34 (2007).
40. D. Paganin, S. C. Mayo, T. E. Gureyev, P. R. Miller, and S. W. Wilkins, "Simultaneous phase and amplitude extraction from a single defocused image of a homogeneous object," J. Microsc. **206**(1), 33–40 (2002).
41. N. Otsu, "A threshold selection method from gray-level histograms," IEEE Trans. Syst., Man, Cybern. **9**(1), 62–66 (1979).
42. F. De Carlo, D. Gürsoy, D. J. Ching, *et al.*, "TomoBank: a tomographic data repository for computational X-ray science," Meas. Sci. Technol. **29**(3), 034004 (2018).
43. R. A. Brooks and G. Di Chiro, "Beam hardening in X-ray reconstructive tomography," Physics in medicine & biology **21**(3), 390–398 (1976).
44. G. T. Herman, "Correction for beam hardening in computed tomography," Physics in Medicine & Biology **24**(1), 81–106 (1979).
45. G. Van Gompel, K. Van Slambrouck, M. Defrise, K. J. Batenburg, J. De Mey, J. Sijbers, and J. Nuyts, "Iterative correction of beam hardening artifacts in CT," Med. Phys. **38**(S1), S36–S49 (2011).
46. M. T. Zeegers, T. van Leeuwen, D. M. Pelt, S. B. Coban, R. van Liere, and K. J. Batenburg, "A tomographic workflow to enable deep learning for X-ray based foreign object detection," Expert Systems with Applications **206**, 117768 (2022).
47. M. T. Zeegers, "A collection of 131 CT datasets of pieces of modeling clay containing stones - Part 1 of 5," Zenodo (2022), https://zenodo.org/doi/10.5281/zenodo.5866227.
48. R. Kumar, M. Purohit, Z. Svitkina, E. Vee, and J. Wang, "Efficient rematerialization for deep networks," *Adv. Neural Inf. Process. Syst.* **32** (2019).
49. C. R. Vogel and M. E. Oman, "Iterative methods for total variation denoising," SIAM J. Sci. Comput. **17**(1), 227–238 (1996).
50. V. Panin, G. Zeng, and G. Gullberg, "Total variation regulated EM algorithm [SPECT reconstruction]," IEEE Trans. Nucl. Sci. **46**(6), 2202–2210 (1999).

51. M. Persson, D. Bone, and H. Elmqvist, "Total variation norm for three-dimensional iterative reconstruction in limited view angle tomography," Phys. Med. Biol. **46**(3), 853–866 (2001).
52. E. Y. Sidky, J. H. Jørgensen, and X. Pan, "Convex optimization problem prototyping for image reconstruction in computed tomography with the Chambolle–Pock algorithm," Phys. Med. Biol. **57**(10), 3065–3091 (2012).
53. L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," Phys. D **60**(1-4), 259–268 (1992).
54. M. J. Lagerwerf, W. J. Palenstijn, F. Bleichrodt, and K. J. Batenburg, "An efficient interpolation approach for exploring the parameter space of regularized tomography algorithms," Fundamenta Informaticae **172**(2), 143–167 (2020).