Deep learning for tomographic reconstruction with limited data

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof. dr. ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op donderdag 3 maart 2022
klokke 15:00 uur

door

Allard Adriaan Hendriksen

geboren te Muscat, Oman
in 1991

Promotor:
Prof. dr. K. J. Batenburg

Co-promotor:
Dr. D. M. Pelt

Promotiecommissie:
Prof. dr. F.A.J. de Haas
Prof. dr. R.M. van Luijk
Prof. dr. P.D. Grünwald
Prof. dr. ir. B.P.F. Lelieveldt
Dr. F. Marone Welford (Paul Scherrer Institute)
Prof. dr. O. Öktem (KTH Royal Institute of Technology)

The research presented in this dissertation was carried out at the Centrum Wiskunde & Informatica (CWI) in Amsterdam.

Financial support was provided by the Netherlands Organisation for Scientific Research (NWO), programme 639.073.506.

Contents

1 Introduction 1

1.1 Background . . . 2

1.2 Research questions . . . 14

2 Improving image resolution 21

2.1 Background and Notation . . . 24

2.2 Method . . . 27

2.3 Results . . . 33

2.4 Discussion . . . 43

2.5 Conclusions . . . 44

3 Noise2Inverse: Self-supervised denoising 45

3.1 Notation and concepts . . . 47

3.2 Noise2Inverse . . . 51

3.3 Results . . . 58

3.4 Discussion and conclusion . . . 70

4 Noise2Inverse for synchrotron tomography 73

4.1 CNN-based denoising . . . 75

4.2 Method . . . 76

4.3 Results . . . 82

4.4 Discussion and Conclusion . . . 89

5 Tomosipo: Flexible tomography in Python 91

5.1 Standard tomography problem . . . 93

5.2 Framework concepts . . . 94

5.3 Case studies . . . 102

5.4 Experimental data . . . 109

5.5 Benchmarks . . . 110

5.6 Discussion and conclusion . . . 112

6 Conclusion and outlook 115

6.1 Contributions and limitations . . . 115

6.2 Outlook . . . 118

Bibliography 121

List of publications 139

Samenvatting 141

Curriculum Vitae 147

Acknowledgments 149

A Appendices 151

A.1 Data acquisition 152

A.2 Noise2Inverse training 152

A.3 Synthetic noise procedure for XRD-CT 153

A.4 Total-variation minimization 154

1
Introduction

Tomography is a powerful technique to non-destructively determine the interior structure of an object. Usually, a series of projection images (e.g. X-ray images) is acquired from a range of different positions. from these projection images, a reconstruction of the object's interior is computed. Many advanced applications require fast acquisition, effectively limiting the number of projection images and imposing a level of noise on these images. These limitations result in artifacts (deficiencies) in the reconstructed images. Recently, deep neural networks have emerged as a powerful technique to remove these limited-data artifacts from reconstructed images, often outperforming conventional state-of-the-art techniques. To perform this task, the networks are typically trained on a dataset of paired low-quality and high-quality images of similar objects. This is a major obstacle to their use in many practical applications. In this thesis, we explore techniques to employ deep learning in advanced experiments where measuring additional objects is not possible.

In this chapter, we first describe several applications of tomography and the importance of accurate reconstructed images. Next, we describe the factors that determine the quality of the reconstructed images and how they are related. In Section 1.1.3, the image acquisition process is introduced with a focus on noise statistics that are relevant to proposed techniques in later chapters. In Sections 1.1.4 and 1.1.5, the tomographic inverse problem is formulated and relevant reconstruction algorithms are reviewed. Next, deep learning techniques for improving the quality of reconstructed tomographic images are introduced. In Section 1.1.7, these deep learning techniques are considered from a Bayesian viewpoint that allows predicting their properties. Next, practical obstacles to the collection of a training dataset are discussed. Finally, an overview is given of the main research questions and the chapters that aim to answer them.

1.1 Background

1.1.1 Applications of tomography

Tomography is a powerful technique to non-destructively determine the interior structure of an object. Since the 1960s, tomography has steadily gained popularity in a variety of applications, ranging from protein structure determination at the nanometer scale [17] to inspection of airplane assemblies of up to several meters [52]. In this section, we discuss tomography as it is used in three places: the hospital, the laboratory, and at a synchrotron facility (particle accelerator). For each, we discuss their characteristics and example use cases, as well as the importance of obtaining high-quality reconstructed images.

Hospital. The use of X-rays in medical applications is perhaps the most familiar to the public and can serve as an introduction to the topic. Investigation using only a single 2D X-ray image is known as radiography and is widely used to diagnose easily visible injuries such as a broken bone. Here, X-rays pass through the body and are collected on a detector. As different parts of the body absorb X-rays to a varying degree, a gray-scale image can be formed that corresponds to the absorption of the X-rays by the body.

Multiple X-ray images are required to investigate an object using tomography, which is known as CT (computed tomography) in medical applications. Here, multiple 2D X-ray images are acquired from varying positions, usually in a continuous fashion. From these images, a 3D reconstruction of the interior of the patient's body can be computed, enabling the diagnosis of diseases that require precise localization of low-contrast features in the patient's body, such as a tumor [85]. Commonly, the X-ray source and detector rotate in a helical pattern around a patient that lies flat, as displayed in Figure 1.1. On the right, a reconstructed image is shown, displaying a cross-section of the hip area. The achievable resolution is often quite coarse, making medical CT unsuitable for many non-medical applications.

Laboratory. Laboratory-based micro-CT is used in a variety of industrial and scientific applications. It is used to scan smaller objects where it provides substantially finer resolution than medical X-ray scanners. Typically, the sample under investigation is mounted on a rotation stage between an X-ray point source and a detector. This type of acquisition is known as circular cone beam geometry, and is illustrated in Figure 1.1. More flexible acquisition is possible, for instance, using the micro-CT scanner at the FleX-ray Laboratory at the CWI [42]. This scanner has ten degrees of freedom, enabling the user to zoom in (by moving the object closer to the source), or to obtain a larger virtual detector by repeating an acquisition with the detector at multiple positions and stitching the projection images together afterwards.

Micro-CT is used to investigate a wide variety of objects, of which we give a few examples here. One application of micro-CT is in art history, where it can be used to assign a date to wooden objects using the tree ring pattern hidden in their interior structure of the wood [24]. Another application is geology, where micro-CT is used to examine one-of-a-kind fossils and to quantify geological properties such

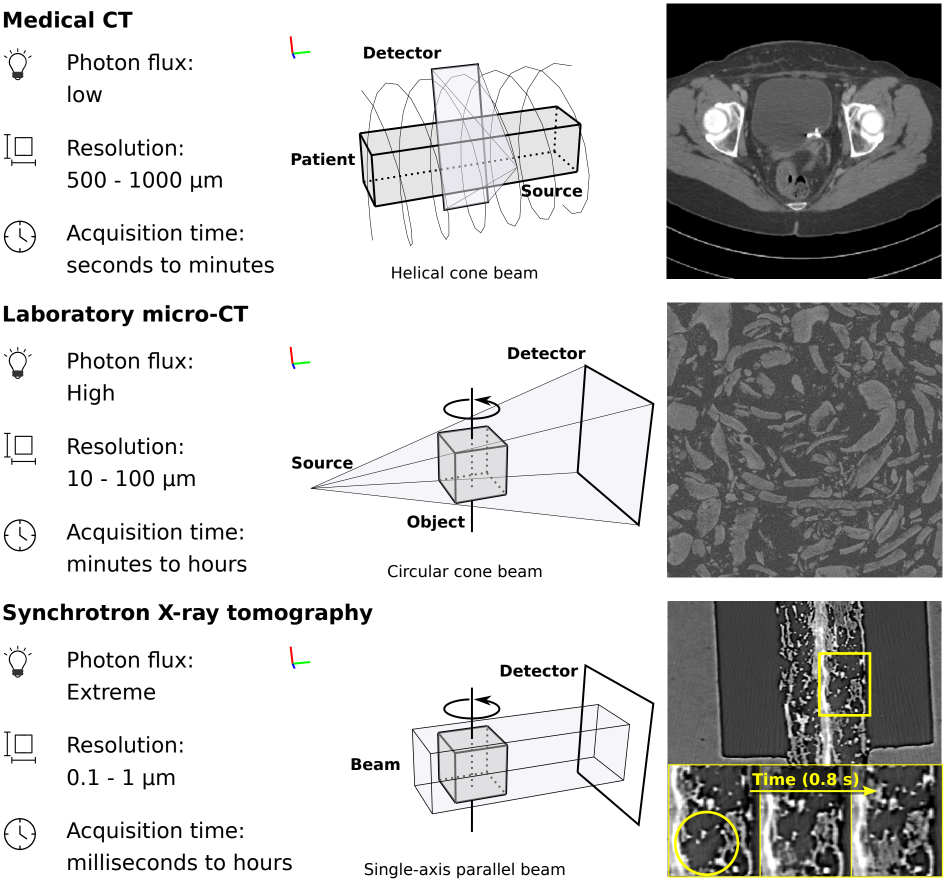


Figure 1.1: Three application areas of tomography. Medical CT is used to diagnose patients using little radiation and at a relatively coarse resolution. Often, the acquisition proceeds using a helical cone beam geometries, where the source-detector pair rotates around the patient. A reconstructed image is show on the right, containing a cross-section of the hip area. Laboratory micro-CT provides higher rates of radiation to investigate smaller objects a higher resolution than medical CT. Typically, the circular cone beam geometry is used, where an objects rotates between a point source and a flat panel detector. On the right, a reconstructed image of oatmeal grains is shown with a voxel size of 17 µm. Synchrotron-based X-ray tomography provides extremely high photon flux and even smaller resolution. The commonly used single axis parallel beam acquisition geometry is shown in the middle. On the right, a 4D (space + time) reconstruction of a hydrogen fuel cell is shown. The region in the yellow square is shown at three time steps, demonstrating that a water bubble is forming inside the fuel cell in the area indicated by the yellow circle.

as porosity and permeability of rock samples [92]. Properties that are important for the manufacturing and development of new batteries can also be quantified using micro-CT [164]. Because X-rays do not easily penetrate the heavy materials present in batteries, however, scanning can take multiple hours [48]. Faster scanning is possible at more advanced facilities.

Synchrotron. Many important scientific and industrial advances are enabled by tomography at one of 50 X-ray synchrotron facilities worldwide [190]. These facilities generate extremely bright X-ray beams using a particle accelerator, enabling acquisition at substantially higher speeds and smaller scales than laboratory-based micro-CT setups. In a typical setup, the sample is mounted on a rotation stage and the photons in the X-ray beam move along parallel lines, resulting in a single-axis parallel beam geometry, which is illustrated in Figure 1.1. In this way, scanning a battery sample takes 5 minutes instead of 4 hours [48].

Because it is a non-destructive technique, synchrotron-based tomography is ideally suited to track internal structural dynamics over time [115]. Here, a full 3D reconstruction of a dynamically evolving process inside an object is made at several time steps, resulting in a 3D movie. This enables examining batteries while they are discharging, for example, and has important implications for understanding and optimizing battery performance [12, 111]. Another application is the the investigation of water dynamics inside a hydrogen fuel cell [195]. This is illustrated in Figure 1.1, which shows the formation of a water bubble inside a fuel cell. The frame rate of these experiments (where each frame is one 3D volume) is typically around 10 Hz, but some experiments have demonstrated frame rates of up to 208 Hz — three times the frame rate of a regular computer monitor [56].

In all of these applications, the quality of the reconstructed images is important. In medical settings for instance, visually identifying a lesion (tumor) is complicated, as the contrast of the lesion with respect to the surrounding tissue is small [123]. Similar considerations arise in art and geology applications where fine-scale features, such as tree rings, can be extremely important. In addition to the visual inspection, quantitative analysis is used to automatically extract parameters of interest out of the reconstructed images, such as the porosity of a rock sample [115]. Here, the images are usually segmented to identify separate features or parts. Preferably, coarse-grained techniques, such as thresholding, flood-filling and clustering are used that can easily be manually tuned, inspected, and applied to large volumes at once. These techniques can be sensitive to imperfections in the reconstructed image, and can therefore benefit from high-quality reconstructions.

1.1.2 Factors that influence image quality

The quality of the reconstructed image depends on several factors in the acquisition process and they are all to some degree interconnected. The relationship between these factors is displayed schematically in Figure 1.2. We first discuss how these factors contribute to the quality of the reconstructed image and then discuss limits that occur in practice.

First of all, measurement noise on the projection images is an important factor

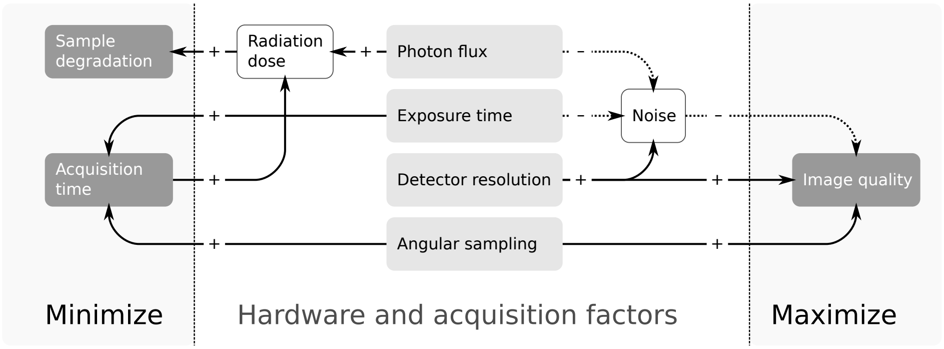


Figure 1.2: The relation between the aims of tomographic imaging and factors that can be influenced during acquisition. In the dark gray boxes, we identify three aims: preventing radiation damage, minimizing acquisition time, and maximizing image quality. The acquisition parameters are shown in the middle column. Solid arrows with a "+" indicate a positive correlation between factors and dashed arrows with a "−" indicate a negative correlation.

as noise carries over into the reconstructed images. The noise level of a projection image is influenced by the exposure time, detector resolution, and photon flux, the rate of photon emission from the source. As more photons are measured, the projection image becomes less noisy. Therefore, the noise level can be reduced by increasing the photon flux or the exposure time of each projection image. Detector resolution plays an important role as well, as doubling the pixel size in each dimension quadruples the number of photons measured per pixel, reducing the level of noise.

Second, the pixel resolution of the detector determines the maximally attainable voxel resolution of the reconstructed volume. In parallel beam setups, the detector pixel width and height determine the minimal resolvable width and height of the voxel. In cone beam setups, the detector and reconstruction resolution are additionally related by the magnification factor that results from moving the object closer to the source. In addition, blurring can be induced on the detector by cross-talk between nearby detector pixels and a large focal spot size, decreasing the effective resolution [31, 161].

Finally, the resolution of the reconstructed images is influenced by the angular sampling rate. As projection images are acquired from more angles, the resolution of the reconstructed images improves. The angular sampling rate is related to the size of the reconstructed volume by the Crowther criterion [95]. In parallel beam setups, for instance, achieving full resolution on an N^3 voxel grid requires ${}^{\pi N}\!/_{2}$ projection images to be acquired over an angular range of 180°. If the angular sampling rate is too low compared to the rotation speed, the projection images may suffer from motion blur. This occurs when the sample rotates too much in the course of one exposure [41]. Often, this effect is negligible in practice.

Other factors may also cause a deterioration of image quality. These typically involve a mismatch between physical reality the mathematical model used for reconstruction. For instance, beam hardening artifacts may be introduced into the reconstructed images when the X-ray beam is not monochromatic [31]. We do not consider these problems in this thesis.

Practical challenges. In practice, the ability to obtain high-quality measurements can be restricted by the wish to prevent radiation damage and minimize acquisition time. In addition, other factors come in play that vary from application to application. In medical tomography, the photon flux and acquisition time are limited to prevent radiation damage. Because the diagnostic use of CT continues to increase, radiation damage is a growing concern: it has been estimated that the percentage of cancer cases in the United States that may be attributable to the radiation from CT studies has increased from 0.4% in 1996 to $1.5 - 2.0\%$ in 2007 [25]. Radiation damage is not limited to human studies and is a frequent concern in many applications.

The photon flux may also be limited by the available hardware. As previously mentioned, the photon flux in laboratory micro-CT is limited, which causes the scan of a battery to take hours. At synchrotron facilities, experiments may also encounter flux limits, for instance during a scan of an operating battery, where radiation heats up the battery and can disrupt the process under investigation.

In order to complete more scans faster, ways of minimizing the acquisition time are always sought for in the hospital, laboratory, and the synchrotron. The acquisition time can also face a hard limit. In 4D tomography, the speed of the interior dynamics of the object restricts the acquisition time of a single time step, as the reconstruction becomes blurred if too much movement occurs during its acquisition. An example of interior dynamics is shown in Figure 1.1, where a water bubble appears in a fuel cell in the course of a second, and intermediate reconstructions are necessary to track its growth.

The acquisition of a single time step is usually accelerated by reducing the angular sampling rate (resulting in undersampling) or by reducing the exposure time of each projection image. Decreasing the exposure time has a limit, however, as most detectors have a maximum frame rate [131]. This imposes a maximum on the angular sampling rate when the acquisition time is fixed.

In laboratory micro-CT, the achievable resolution in the reconstruction can be limited by the object size. Typically, the projection images can be magnified by moving the object closer to the source. If the object is too large however, its projection may not fall entirely within the detector, which can lead to truncation artifacts in the reconstruction. This imposes a maximum magnification, and therefore a tension exists between object size and achievable reconstruction resolution.

In summary, several factors influence the quality of the reconstructed images that must be balanced to take into account hardware limits, total scan time, and acceptable radiation dose.

1.1.3 Noise statistics in absorption-contrast tomography

As discussed above, noise and noise statistics play an important role in tomography, and specifically in the algorithms proposed in this thesis. We use the properties of

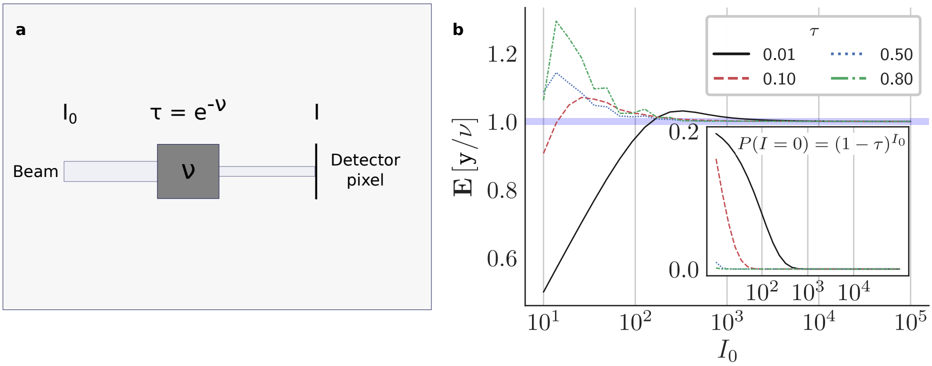


Figure 1.3: Absorption of an X-ray beam in a single-pixel detector setup. (**a**), An X-ray beam with initial intensity I_0 passes through a uniform mass with absorption coefficient ν. Each photon has a probability τ of being transmitted through the mass. (**b**), The relative bias in y due to non-linear effects in post-processing as a function of I_0, the emitted photon count. The horizontal blue bar indicates a 1% offset from ν. In the inset, the effect of photon starvation is plotted for the same values of I_0.

the noise in tomographic imaging to improve reconstructions using deep learning. One specific feature that the noise is assumed to have is that it is unbiased. A random variable a is said to be an unbiased estimate of b if

$$b = \mathbb{E}\left[\mathsf{a}\right] = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} a_i, \tag{1.1}$$

i.e., if the average value of an increasing number of measurements $a_i \sim \mathsf{a}$ converges to b [178], where the second equality is a consequence of the strong law of large numbers. In this section, using an example of a single-pixel detector, we show that under reasonable conditions the bias of the noise is very mild.

First, we introduce the single-pixel detector setup and describe how absorption can be modeled as a random process. Next, we describe how the absorption process can become biased as a result of photon starvation and Jensen's inequality and describe conditions under which the bias is negligible. Finally, we discuss other sources of randomness in the acquisition process.

Suppose that an X-ray beam is measured on a single pixel detector, after it has passed through a uniform mass. Furthermore, suppose that the X-ray source emits I_0 photons and that the mass of unit width has an attenuation coefficient ν, describing its tendency to absorb photons. Each individual photon has a probability $\tau = e^{-\nu}$ to be transmitted through the object as a result of Beer-Lambert's law of attenuation [31]. The number of photons detected at the detector, I, is therefore $B(I_0, \tau)$ binomially distributed, with I_0 the number of emitted photons. The goal is to recover the attenuation coefficient ν from the measured number of photons I. This setup is illustrated in Figure 1.3 (**a**).

The expection of I has an exponential relation to ν, i.e.,

$$\mathbb{E}\left[\mathsf{I}\right] = \sum_{k=0}^{I_0} \tau^k (1-\tau)^{I_0-k} \binom{I_0}{k} k = I_0 \tau = I_0 e^{-\nu}. \tag{1.2}$$

In practice, the attenuation coefficient ν is therefore recovered by taking the logarithm, i.e,

$$\nu \approx \mathsf{y} := -\ln \frac{\mathsf{I}}{I_0}, \tag{1.3}$$

where we define the attenuation projection value y as the logarithmic fraction on the right-hand side.

The question is whether y is an unbiased estimate of ν. Since Equation (1.3) is non-linear, it is not self-evident that the noise is unbiased, i.e., $\mathbb{E}\left[\mathsf{y}\right] = \nu$. Two phenomena exert an opposite bias on this expectation: photon starvation and Jensen's inequality.

Photon starvation occurs when a pixel does not collect any photons, i.e., the event $\{\mathsf{I} = 0\}$. This event occurs with probability

$$P(\mathsf{I} = 0) = (1-\tau)^{I_0}. \tag{1.4}$$

When this happens, calculating y is more difficult, since the logarithm of zero is undefined. A practical solution is to set the photon count to 1 before taking the logarithm. This introduces a downward bias in the statistics of y.

The other effect on the expected attenuation projection function is purely mathematical. Since the negative logarithm is a convex function, Jensen's inequality [66] states that we have

$$\nu = -\ln \frac{\mathbb{E}\left[\mathsf{I}\right]}{I_0} \leq \mathbb{E}\left[-\ln \frac{\mathsf{I}}{I_0}\right] = \mathbb{E}\left[\mathsf{y}\right]. \tag{1.5}$$

Here, the equality on the left-hand side is a direct result of Equation (1.2). Hence, Jensen's inequality exerts an upward bias on y and the photon starvation correction exerts a downward bias.

The effect of photon starvation and Jensen's inequality is illustrated in Figure 1.3. Here, the measurement process has been repeated a million times to obtain an accurate estimate of $\mathbb{E}\left[\mathsf{y}\right]$. For small values of the transmittance τ, photon starvation correction exerts a strong downward bias at smaller values of I_0. For larger values of τ, the upward bias slowly reduces, until at roughly 1000 emitted photons, the bias is mostly removed. As a rule of thumb, for non-highly attenuation objects, at 1000 emitted photons per pixel there appears to be little bias in the noise.

For the sake of argument, some aspects of the tomographic acquisition pipeline have been simplified. In the standard treatment, the measurement of a photon is a statistical process that can be divided into three parts [31, 192]: (1) the probability of a photon being emitted from the source, (2) the probability of an emitted photon being transmitted through the object, and (3) the probability of a transmitted photon being detected by the detector. The emitted number of photons I_0 per time step is typically Poisson-distributed [192]. The binomial process describing the absorption of photons by the object commonly approximated using a Poisson distribution as emission rate I_0 is typically large [31]. The measurement process on the detector can be modeled using a Poissonian-Gaussian model that is influenced by the quantum efficiency of converting photons to electrons (Poisson) and electronic noise (Gaussian) due to thermal noise or bias currents [55, 192]. In the regime where the absorption process is unbiased, however, these effects are minor.

1.1.4 Formulation of the inverse problem

Many common tomography setups can be modeled as a collection of line integrals through space where the ith measurement $y_i \in \mathbb{R}$ is obtained as a line integral

$$y_i = \int_{\mathbb{R}} x(\mathbf{s}_i + t\boldsymbol{\eta}_i) \, \mathrm{d}t \tag{1.6}$$

through a point $\mathbf{s}_i \in \mathbb{R}^3$ with direction $\boldsymbol{\eta}_i \in \mathbb{R}^3$ [173].

The canonical case is absorption contrast tomography, which is also used in medical applications. In absorption contrast tomography, the reconstruction problem can be posed as a linear discrete inverse problem. Suppose measurements $\mathbf{y} \in \mathcal{Y} = \mathbb{R}^{N_\theta \times N_p^2}$ are acquired from N_θ positions using a square detector that is divided into N_p^2 pixels. Define the cubic reconstruction volume $\mathbf{x} \in \mathcal{X} = \mathbb{R}^{N_v^3}$ on a voxel grid and let $\mathbf{A}$ denote the projection matrix such that $\mathbf{A}_{ij}$ describes the absorption by object voxel j of the ray to measurement i. The goal is to determine the value of $\mathbf{x}$ that gave rise to the measurement

$$\mathbf{A}\mathbf{x} = \mathbf{y}. \tag{1.7}$$

An algorithm, $\mathbf{R} : \mathcal{Y} \to \mathcal{X}$, that recovers the volume $\mathbf{x}$ from the measurements $\mathbf{y}$ is known as a tomographic reconstruction algorithm.

1.1.5 Tomographic reconstruction algorithms

Several approaches to tomographic reconstruction can be distinguished. Each approach can be characterized in terms of computational speed, sensitivity to noise and undersampling, and their ability to be applied to different acquisition geometries.

Fast filtered backprojection (FBP)-type reconstruction algorithms have been developed over the years for common acquisition geometries, such as single-axis parallel-beam (FBP) [134], circular cone beam (FDK) [53], and the helical cone beam (Katsevich's algorithm) geometry [89]. Each algorithm is specific to a single acquisition geometry. They are well-suited for fast, parallel computation [140] and are derived from a continous formulation of the inverse problem. These algorithms typically consist of two steps. First, the acquired projection data $\mathbf{y}$ is convolved with a filter $\mathbf{h}$. Next, the filtered data is backprojected $(\mathbf{A}^T)$ into the volume, leading to

$$\mathbf{x}_{\text{FBP}} = \mathbf{A}^T \left(\mathbf{h} * \mathbf{y}\right). \tag{1.8}$$

Reconstructions computed by FBP-type methods can suffer from significant artifacts when the number or quality of the measurements decreases. Artifacts can be remedied to an extent by altering the filter $\mathbf{h}$, making it more similar to a low-pass filter [134]. An algorithm that is similarly fast, specific to one acquisition geometry, and sensitive to noise, is the GridRec algorithm [118]. Doing away with the backprojection step, the algorithm can be considered as a non-uniform fast fourier transform [54], which aids its computational performance as images become larger [141]. The GridRec and FBP-type methods are collectively referred to as direct methods, distinguishing them from iterative and variational methods that are described next.

Iterative reconstruction algorithms aim to solve Equation 1.7 by treating it as a minimization problem that is typically formulated as

$$\mathbf{x}_{\text{iter}} = \underset{\mathbf{x} \in \mathcal{X}}{\arg\min} \, \|\mathbf{A}\,\mathbf{x} - \mathbf{y}\|_2^2. \tag{1.9}$$

As the name implies, the methods operate by refining the reconstruction over a number of iterations. An elementary example is the Landweber iteration [105]

$$\mathbf{x}_0 = 0 \in \mathcal{X}, \tag{1.10}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \eta \mathbf{A}^T \left(\mathbf{y} - \mathbf{A}\,\mathbf{x}_i\right). \tag{1.11}$$

In each iteration, the residual error $\mathbf{y} - \mathbf{A}\,\mathbf{x}_i$ is backprojected and added to the current estimate after multiplication by a step size η. Common iterative reconstruction algorithms include ART [59], SART [6], CGLS [76], and SIRT [57, 60]. An advantage of iterative reconstruction methods is that they can be applied to any (non-standard) acquisition geometry where the forward operator $\mathbf{A}$ can be computed. Compared to direct methods however, they are substantially more computationally intensive. In addition, like direct methods, they can suffer from artifacts in case of noise or undersampling, which is typically remedied by early stopping, i.e., stopping the iteration before convergence has been reached. This effectively retains the low-frequency components of the reconstruction and moderates the high-frequency components.

Variational reconstruction algorithms aim to maximize not just the agreement with respect to the measured data, but also impose prior knowledge on the reconstruction through regularization. A common varational method in tomography is Total-Variation Minimization (TV-MIN) [168] that obtains a reconstruction through minimizing the objective

$$\mathbf{x}_{\text{var}} = \underset{\mathbf{x} \in \mathcal{X}}{\arg\min} \, \|\mathbf{A}\,\mathbf{x} - \mathbf{y}\|_2^2 + \lambda g(x), \tag{1.12}$$

where the regularization function is defined by $g(x) = |||\nabla \mathbf{x}|||_1$ with ∇ the discrete gradient operation. This regularization function promotes piecewise constant reconstructions. The degree to which the solution must satisfy prior knowledge is modulated by the regularization parameter λ. Algorithms for minimizing the objective in Equation (1.12) include FISTA [15] and the Chambolle-Pock algorithm [33]. Compared to the previously described iterative reconstruction methods, variational methods tend to be slower to compute, although the addition of regularization can make them more robust to noise and undersampling. As the prior knowledge that informs the regularization function g depends on the object class being scanned, manual tuning of both g and λ may be required to obtain optimal results on new classes of objects [122]. Deep learning methods hold the promise of learning prior knowledge by example.

1.1.6 Convolutional neural networks (CNNs) in tomography

Among techniques for removing artifacts from reconstructed images, deep convolutional neural network (CNN)-based methods have shown strong results, often outperforming conventional state-of-the-art techniques [5, 81, 85, 140]. A substantial subset of these techniques are known as post-processing techniques. Here, a reconstruction is first computed using a fast reconstruction algorithm and a CNN is used to post-process the reconstructed image. Because the post-processing approach has proven to be applicable to large-scale tomographic problems [140], it is the main focus of this thesis.

CNNs are usually organized in layers: the input image is copied into the first layer, the output image is the final layer, and computations are performed in the intermediate layers. The CNN computes convolutions that are parameterized by thousands to millions of small filters (typically 3 × 3). In each intermediate layer, the images in the previous layer are convolved with these filters, after which a pixel-wise non-linear function is applied. A common non-linearity is the ReLU function, which is the identity for positive arguments and zero otherwise [79, 133]. A prototypical CNN is illustrated in Figure 1.4. In practice, results can be improved substantially by using more complex network structures. Throughout this thesis, we use the UNet [38, 156], DnCNN [199], and MS-D networks [140, 143], which are introduced in the relevant chapters.

A CNN is usually prepared to perform a specific image-to-image translation task by supervised training. During training, the network is presented with low-quality input images and high-quality target images from a training dataset. On each image pair, the parameters of the convolutions are optimized to minimize the training loss, i.e., the difference between the output and target image. The training loss is commonly quantified by the pixel-wise mean square error [66] and minimized using stochastic gradient descent [153].

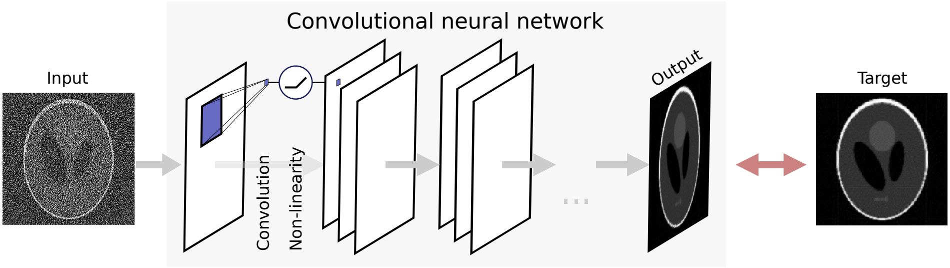


Figure 1.4: A prototypical convolutional neural network (CNN) in a supervised training arrangement for denoising. The CNN is composed of multiple layers. To compute an image in the next layer, images in the previous layer are convolved with a small filter (depicted in blue), after which a non-linear function is applied. To train a CNN to perform denoising, the output image is compared to a target image, and the convolution filters are updated accordingly.

1.1.7 CNN training as Bayesian function estimation

When formulated in a Bayesian statistical framework [9], the solution of the training minimization problem can be described in closed form. This solution can be used to predict important properties of the trained neural network, which we use in the methods developed in this thesis.

Suppose the pairs of input and target images are drawn from a prior distribution as follows

$$\mathrm{u}_i, \mathrm{v}_i \sim (\mathsf{u}, \mathsf{v}), \quad i = 1, \ldots, N, \tag{1.13}$$

where u and w are image-valued random variables. Then the solution of the training minimization problem on this dataset can be considered as approximating the regression function

$$h^* = \arg\min_h \mathbb{E}_{\mathsf{u},\mathsf{v}} \left[\| h(\mathsf{u}) - \mathsf{v} \|_2^2 \right] \approx \arg\min_h \sum_{i=1}^{N} \| h(\mathrm{u}_i) - \mathrm{v}_i \|_2^2, \tag{1.14}$$

that minimizes the expected prediction error among a class of functions [66]. In the case of the squared L_2-norm, the regression function is known [4] and equals the conditional expectation

$$h^*(u) = \mathbb{E}\left[\mathsf{v} \mid \mathsf{u} = u \right]. \tag{1.15}$$

In practice, the trained neural network does not equal h^* and an approximation is obtained. However, since equation (1.15) describes the solution in the ideal case, it can be used to predict properties of the output of trained neural networks in practice, as we show in Chapter 3.

1.1.8 Deep learning challenges in tomography applications

Tomographic applications of deep learning face a challenge in obtaining a dataset that is suitable for supervised training. We first identify three issues that underly this challenge, followed by common strategies to work around them. First of all, as outlined in Section 1.1.2, obtaining high-quality images may be complicated, expensive, or impossible.

Second, obtaining a large quantity of similar data can be difficult. That is, data that is similar enough to the object under investigation that it can provide a useful inductive bias during neural network training. At synchrotron facilities, beam time may be limited or too expensive. Using micro-CT, scanning multiple objects may take prohibitively long (batteries, for instance). In addition, the object under investigation may be unique or expensive, for instance in the case of a historical art object. In medical applications, large quantities of similar data are available, as many scans are performed every day. Nonetheless, the collection and dissemination of large datasets may encounter ethical concerns and the normalization of data acquired using different scanners is challenging in both CT and magnetic resonance imaging (MRI) [35].

Finally, the exact pairing of the low- and high-quality datasets can be time consuming, requiring manual tuning to register the images. This is important because the accuracy of registration directly impacts the quality of the trained network [166]. When the object moves or degrades during a scan, accurate pairing is even more complicated and may require advanced deformable registration methods [49]. Therefore, deficiencies in data quality, sample quantity, and image pairing are important to address.

In practice, these issues are circumvented using several strategies, each with their own downsides. The most prevalent is the use of synthetic degradation of high-quality data. For instance, noise can be added to high-quality projection images to simulate a low dose acquisition [123]. Another tactic that is popular in MRI is the use of undersampling, where the low-quality dataset is obtained by removing a fraction of the projection images [96, 140]. Synthetic degradation solves the problem of pairing, but obtaining high-quality data remains an issue.

Another approach is to use a computationally expensive reconstruction algorithm to compute a “gold standard” reconstruction. An example of this approach is found in neutron imaging [181], where cheap FBP-type reconstructions are paired with expensive model-based reconstructions. Here, the trained networks do not exceed the reconstruction quality of the expensive reconstruction algorithm and the aim is mainly to speed up future computations of the reconstruction.

When accurate pairing is a problem, but low-quality data and high-quality data can be obtained separately, then training using cycle-consistency may be used. In this approach, two translation networks are trained. One is trained to convert the low-quality to high-quality data and the other is trained to perform the reverse transformation. Meanwhile, two discriminator networks are trained to distinguish between the outputs of the networks and real collected data. The translation networks are optimized to minimize their chance of detection by the discriminator networks and to maximize their cycle consistency: ideally, the application of the first translation network followed by the second should yield the original image. This strategy is applied to multiphase coronary CT angiography, where multiple CT measurements of the heart are taken using different levels of radiation dose [84].

In practice, however, the training process is extremely sensitive to hyperparameters and may require repetitive manual tuning to prevent optimization instability [7, 62] and mode collapse [125, 151].

Finally, synthetic data may be used to obtain a training dataset. In practice, however, it is questionable whether the data is similar and representative enough to trust the trained neural network on real-world data. This vulnerability has raised concerns about the proliferation of synthetic data in medical applications, for instance [36]. The use of synthetic data is therefore advised mainly to complement real-world data.

1.2 Research questions

In this thesis, strategies are explored to employ deep learning in situations where a large set of similar data is not available, or high-quality measurements are absent. An outline of the thesis is provided first, followed by a description of the research questions on the next pages.

In Chapter 2, we investigate how the resolution of reconstructed images can be improved using deep learning when only a single object is available. This involves an acquisition strategy where the object is scanned twice without scanning the full object at high-resolution. Using this approach, it is possible to construct a supervised training dataset containing paired low-resolution and high-resolution images, from which a neural network can be trained.

In Chapter 3, we study the properties of noise in tomography and determine whether techniques from photographic image denoising carry over into the domain tomographic of reconstructed images. In addition, we develop a deep learning technique for denoising (Noise2Inverse) that can be applied to tomography and related inverse problems. This approach does not require a supervised training dataset containing high-quality images: in fact, the method can be applied to any existing tomographic dataset.

In Chapter 4, the applicability of Noise2Inverse to real-world synchrotron data is explored. Specifically, the use of additional information from the space, time, and spectrum-like domain is investigated as a means to improve reconstruction quality.

While validating the aforementioned approaches using real-world data, a recurring problem was determining the exact geometry of the tomographic acquisition. In Chapter 2 for instance, a low-resolution and high-resolution reconstruction had to be perfectly registered (made to overlap) in order to train a neural network. This was complicated by the fact that the positions of the source and detector were not known with sufficient precision, requiring sensitive tuning after the acquisition was complete. In Chapter 4, a substantial improvement in image quality depended on a tiny deviation of the investigated object's rotation speed from its reported value that was only discovered after careful investigation.

It has therefore become apparent that software is needed that can provide support to perform small and common adjustments to standard geometries, as well as support to devise new acquisition geometries that are increasingly used in advanced experiments. This is the topic of Chapter 5.

The chapters deal with the following research questions, which are described on a separate page each.

Research question 1. *Can deep learning be used to improve the resolution of reconstructed images using a tomographic acquisition of a single object?*

In Chapter 2, we propose a deep learning method for improving the resolution of reconstructed images of a single object. It involves a custom acquisition protocol for cone beam micro-CT that enables the reconstruction of a region of interest at low-resolution and at high-resolution. The reconstructions of the region of interest serves as a training dataset for a deep learning model. The trained model is applied to the full low-resolution reconstruction to obtain images of the whole object at high resolution.

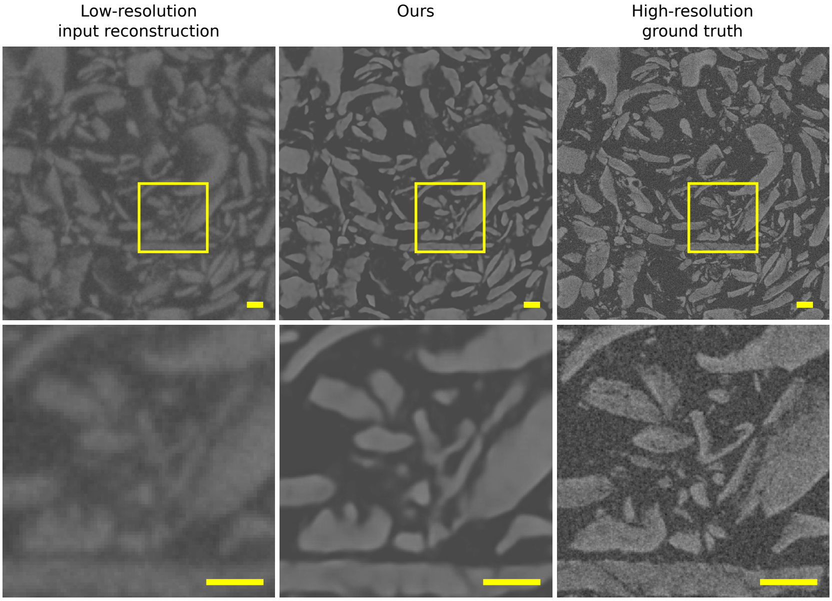


Figure 1.5: The result of applying our proposed method for resolution improvement to a dataset containing oatmeal grains. The images display a slice through a region of interest that was acquired for validation. The top row shows the original low-resolution reconstruction, the results of the proposed method, and a ground truth reconstruction that was used for validation. The bottom row shows a 4× magnification of the region indicated by the yellow square.

Research question 2A. *Many image denoising methods depend on the assumption that noise in one pixel is uncorrelated to noise in any other pixel. Can such image denoising techniques achieve similar performance on tomographic reconstructed images as they do on noisy images that satisfy the no-correlation assumption?*

In Chapter 3, the properties of noise in photographic images and in reconstructed images is compared. Many photographic image denoising methods depend on the assumption that noise in one pixel is not correlated to noise in another pixel. In tomography, however, backprojection smears out the noise in a detector pixel across a line through the reconstructed images, which may cause the noise in one pixel to be correlated to noise in other pixels of the reconstructed image. The effect of applying methods with the no-correlation assumption to tomographic images is investigated in Section 3.3.4.

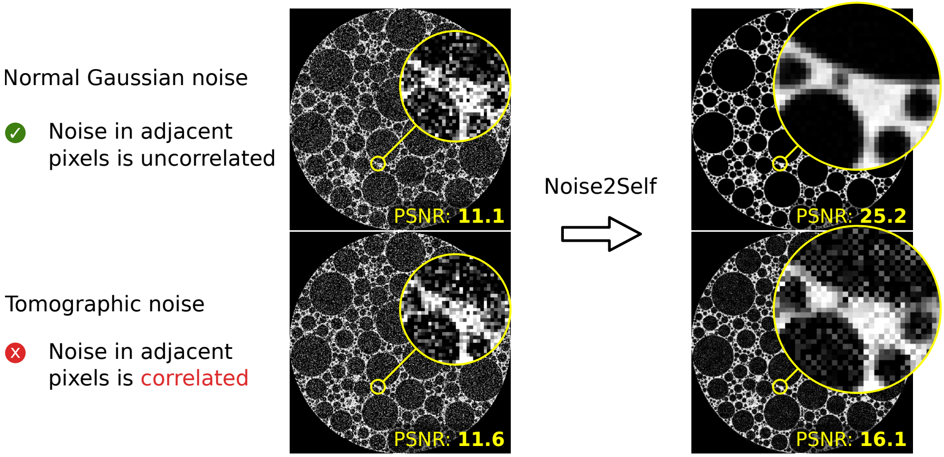


Figure 1.6: The performance of Noise2Self, a photographic image denoising method introduced in Chapter 3, is compared as it is applied to noise common to photography (Gaussian noise, top row) and to noise common to tomography (bottom row). The noise levels (PSNR with respect to ground truth) of the noisy images are similar (left column). After application of Noise2Self both images are less noisy (right column), but the PSNR improvement is substantially smaller in the case of tomographic noise.

Research question 2B. *Can deep learning be used to denoise a single 3D tomographic acquisition without any additional training data?*

In Chapter 3, we propose Noise2Inverse, a method for training deep neural networks to denoise reconstructed images using only noisy training images, i.e., no additional noise-free measurements are required. By considering the training process of neural networks in an idealized mathematical framework, the solution to the training process can be written down in simple terms, using Bayesian probabability theory. Making use of standard results on statistical independence and linearity in probability theory, an argument is made that any solution to the proposed training scheme is forced to remove noise from its input. It is demonstrated that neural networks trained using the proposed scheme indeed remove noise. In fact, they obtain better image metrics than conventional reconstruction methods on simulated data.

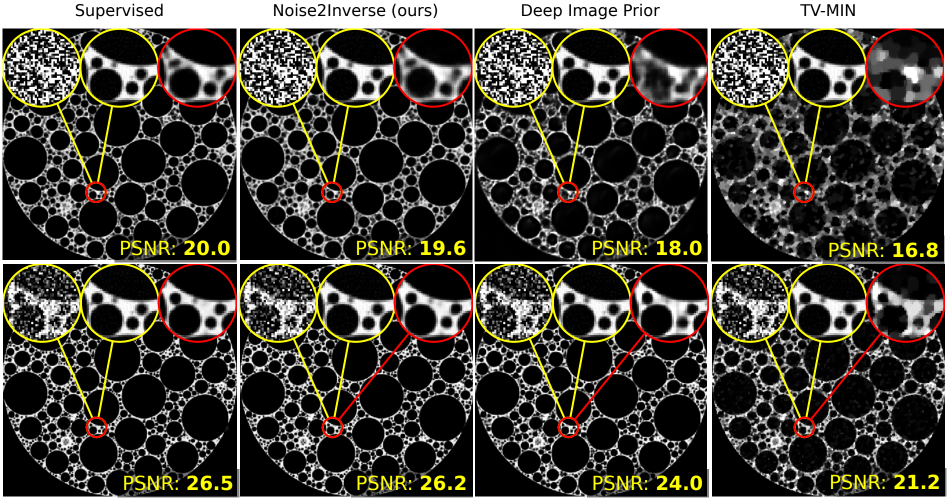


Figure 1.7: A comparison of the proposed Noise2Inverse method to other reconstruction methods on a simulated noisy foam phantom. Results are displayed of supervised training (with access to noise-free data), the proposed Noise2Inverse method (with access to only noisy data), the unsupervised Deep Image Prior, and total-variation minimization (TV-MIN). In each panel, the insets show a magnification of the noisy FBP reconstruction, the ground truth, and the result of the reconstruction method (in red). The noise level in the top row is more challenging than in the bottom row.

Research question 3. *Can self-supervised denoising be applied to real-world synchrotron-based tomographic experiments?*

In Chapter 4, we demonstrate that the Noise2Inverse method can be used to denoise challenging real-world tomographic datasets obtained at synchrotron X-ray facilities. In Chapter 3, only 2D spatial information was taken into account in applications of Noise2Inverse. In this chapter, we expand the application of Noise2Inverse in space, time, and spectrum-like domains. This development enhances applications to static and dynamic micro-tomography as well as X-ray diffraction computed tomography (XRD-CT). We show results on a micro-tomography dataset, a dynamic micro-tomography dataset, and on an X-ray diffraction computed tomography dataset. These results demonstrate the ability of Noise2Inverse to perform accurate denoising and its potential to enable a substantial reduction in acquisition time while maintaining image quality.

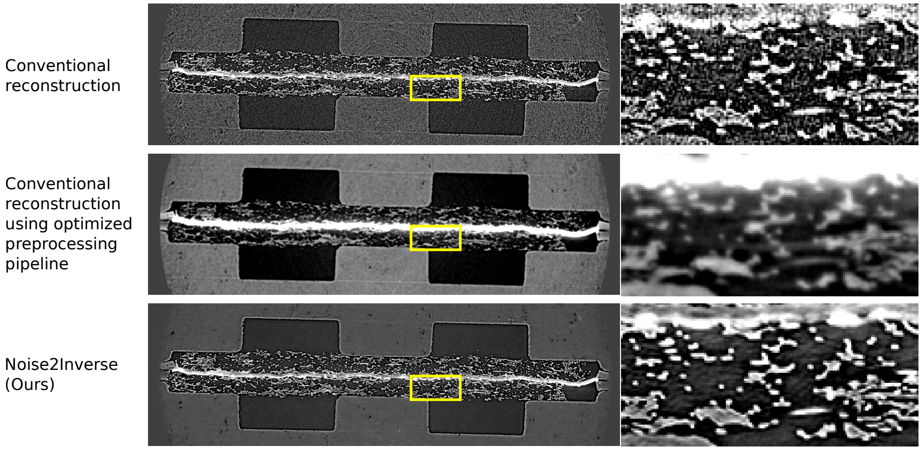

Conventional reconstruction
Conventional reconstruction using optimized preprocessing pipeline
Noise2Inverse (Ours)

Figure 1.8: Reconstructions of a noisy synchrotron micro-tomography dataset of a fuel cell. From top to bottom, reconstructions of a horizontal slice using a conventional reconstruction algorithm (GridRec [118]), using a manually optimized preprocessing pipeline [29], and a reconstruction using Noise2Inverse. Maginifications of the region indicated by the yellow square are shown on the right.

Research question 4. *Can the implementation of reconstruction algorithms for advanced synchrotron tomography techniques be aided by a concise and efficient way to express acquisition geometries in terms of basic building blocks, primitive geometric transformations, and their compositions?*

In Chapter 5, we present the tomosipo software package. Its application programming interface (API) provides primitives to create common tomographic acquisition geometries and several composable tools to manipulate them, such as geometric transformations. In addition, the package allows making full use of the graphics processing unit (GPU) to efficiently compute reconstructions using advanced algorithms.

We demonstrate the ease of making common adjustments to an acquisition geometry, such as changing the center of rotation. In addition, the design and implementation of recently developed synchrotron-based tomography techniques is demonstrated, specifically diffraction contrast tomography (DCT) [184] and X-ray scattering tensor tomography (XSTT) [93]. Reconstructions of real-world data from synchrotron and laboratory micro-CT sources are shown, computed using several common reconstruction algorithms. Finally, bechmarks demonstrate the utility of being able to take full advantage of the GPU.

```
t = np.linspace(-1, 1, 100)   # Time t = -1.0, -.98, ..., 1
s = 2 * np.pi * t             # Angle
radius = 2                    # Radius of helix
h = 1.0                       # Vertical "speed"

vg = ts.volume()
pg = ts.cone(src_orig_dist=radius, src_det_dist=2 * radius)

R = ts.rotate(pos=0, axis=(1, 0, 0), angles=s)
T = ts.translate(axis=(1, 0, 0), alpha = h * s / (2 * np.pi))
H = T * R

ts.svg(vg, H * pg.to_vec())
```

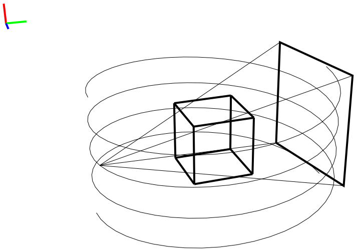


Figure 1.9: A demonstration of composing primitive geometric transforms in tomosipo. A helical acquisition geometry is defined using the code on the left and displayed on the right. First, a non-moving reconstruction volume and source-detector pair are defined. Next, a helical movement is defined by composing the primitive translation and rotation transforms. Finally, the helical transform is applied to the source-detector pair to set it in motion.

2 Improving image resolution

“I think that the referee did not allow much, but didn’t see much. You can hardly whistle for something you cannot see.”

“Ik denk dat die scheidsrechter niet zoveel toe liet, maar zag niet zoveel. Je kan moeilijk fluiten als je iets niet ziet.”

Johan Cruijff,
NOS television, 7 July 1998

Tomography is a powerful technique for reconstructing an image of the interior of an object from a series of its projections, acquired from a range of angles. In computed tomography (CT), a 3D volumetric representation of the object is computed from a set of projections using a tomographic reconstruction algorithm. A broad range of imaging modalities can be used for acquiring the projection images, including X-ray imaging [39], electron imaging [128], neutron imaging [86], and optical imaging [8], all resulting in similar computational reconstruction problems.

Improving the resolution of tomographic 3D volumes is an important goal in the development of new tomographic scanners and their accompanying software. Improvements in resolution enable new developments in materials science [159], geology [39], and other fields of inquiry [172]. In some cases, the resolution of the acquired projections is limited by the effective pixel size of the detector, which imposes a discretization on the measured data that is carried over into the reconstructed 3D volume. For certain tomographic scanners, this limit can be overcome by zooming into a region of interest, decreasing the effective pixel

This chapter is based on:
A. A. Hendriksen, D. M. Pelt, W. J. Palenstijn, S. B. Coban, and K. J. Batenburg. “On-The-Fly Machine Learning for Improving Image Resolution in Tomography”. *Applied Sciences* 9.12 (2019).

size of the detector [39]. Often, other properties of the imaging process also limit the resolution, such as the spot-size of the radiation source or pixel cross-talk on the detector. In such cases, the actual resolution at which a 3D image can be reconstructed is lower than the theoretical maximum based on the detector resolution [31]. When using standard reconstruction algorithms, such as the well-known Filtered Backprojection method (FBP) or Algebraic Reconstruction Technique (ART), the resolution of the reconstructed volume is inherently limited by the resolution and signal-to-noise ratio of the acquired projection data [61, 132].

Various tomographic techniques permit zooming in to a specific parts of an object, leading to magnified projection images. At synchrotron light sources, X-ray images are acquired by converting the high energy photons transmitted through the object into visible light using a scintillator. The visible light is converted into digital images by a conventional high-resolution image sensor. Magnification of the region of interest is achieved by magnifying the visible light using optical instruments placed between the scintillator and image sensor [172]. Laboratory CT systems, on the other hand, have a natural zooming ability, since the X-rays emanate from a point source and are projected onto a linear (fan beam) or planar detector (cone beam). Therefore, moving the object closer to the source magnifies the projected image on the detector. In this chapter, we focus on the 3D cone-beam setup, although the proposed approach is applicable to other tomographic techniques as well, including synchrotron tomography.

A variety of strategies have been proposed for increasing the resolution of tomographic volumes, either by changing the scanning process, or by changing the reconstruction method. When changing the scanning process, several strategies are commonly used:

- High-resolution 3D images can be obtained if a small section can be physically extracted from the object and then scanned at a higher magnification. This provides only information on the particular section, and involves the destruction of the full sample [92].
- Region-of-interest tomography focuses the imaging system on a sub-region of the object, obtaining a set of projections in which that region is always visible, but also superimposed on the surrounding structures. This only recovers high-quality 3D information of the region of interest and leads to challenging image reconstruction problems, as truncation artifacts can hamper the reconstruction quality [135].
- In some cases, it is possible to move the detector while performing a scan, creating a large projection by stitching the images for several detector positions [183]. This permits capturing the full object at a high zoom factor, yet at the cost of a strong increase in radiation dose, scanning time, and computation time.

The resolution of the reconstruction can also be increased by incorporating certain prior knowledge about the scanned object into the reconstruction algorithm, resulting in super-resolution reconstruction. In CT, such prior knowledge comes in many forms, e.g., sparsity of the reconstructed image [108], sparsity of its gradient [19, 169], or knowledge of the materials constituting the object and their attenuation coefficients [13]. However, such prior knowledge is often not available in practice and introduces the risk of making assumptions that are not in good agreement with the actual scanned object, which can decrease the quality of the reconstructed image.

In recent years, machine learning has shown the ability to improve resolution in a range of imaging applications [106, 155]. In particular, convolutional neural networks (CNN) have been applied successfully to attain super-resolution for a wide range of imaging modalities [106]. By training the CNN to compute the mapping between low-resolution data and specially obtained high-resolution training data, the characteristics of the datasets can be learned, removing the need for manually choosing a model. However, this approach relies on the availability of high quality training data for a series of similar objects, which for tomography would consist of high- and low-resolution scans of these objects. Such data are often difficult to obtain in practice for two main reasons:

- when the objects under investigation are unique (i.e., no batches of similar objects are available), it is not possible to obtain the training data;
- creating high-resolution scans requires long scanning time and may also require high radiation dose, which can be unacceptable for dose-sensitive objects.

Therefore, existing machine learning approaches to achieve super-resolution in tomographic imaging are often infeasible in practice.

In this chapter, we propose a novel technique for computationally improving image resolution in tomography. The technique integrates a specially designed scheme for acquiring the scan data and a machine learning method for super-resolution imaging. The data acquisition protocol ensures that certain sub-regions of the scanned object can be reconstructed at high resolution, while still keeping the total scanning time and dose relatively low. By using these high-resolution sub-regions as a training target for creating a super-resolution version of the full object, key morphological properties of the scanned object can be captured even if the scanned object is unique and no similar objects are available.

Rather than exploiting similarity between a large batch of objects, our approach exploits self-similarity within a single object. As such, it is expected to perform properly if the object has high self-similarity, meaning that the local structure of the material is consistent throughout the entire object. Objects with such self-similar structures are abundantly available in nature, and are investigated in detail specifically for their structure. Examples include the crystallization of rare fossils and meteorites, or the porosity and texture of rocks and soils in geology [92]. Furthermore, in materials science, investigations are conducted on the micro-structure of metal foams, batteries, and other materials to improve their physical properties [87, 159]. The goal of this chapter is to present an approach for improving the resolution of tomographic reconstructions of these type of objects.

We show visual and quantitative results for simulated and experimentally acquired cone-beam CT datasets.

This chapter is structured as follows. Section 2.1 introduces necessary notation. In Section 2.2, we present our method. In Section 2.3, we describe the experiments that were performed to evaluate the accuracy of the proposed approach. Results are presented on simulated and experimentally acquired data. In Section 2.4, we discuss these results and provide possibilities for further research. Finally, our conclusions are presented in Section 2.5.

2.1 Background and Notation

2.1.1 Tomography

In circular cone-beam CT, an X-ray point source and flat-panel detector are fixed opposite to each other at some distance, which we refer to as the *source-detector distance* (*SDD*). In between, at a distance *SOD* (*source-object distance*) to the source, the object under study is mounted on a rotation stage. Equivalently, the object can be fixed while the source and detector rotate around it. In either case, X-ray images are acquired at discrete angles during rotation. The resulting stack of pictures is called the *projection dataset*. The setup is displayed in Figure 2.1.

Cone beam tomography can be modeled as follows. Let the *object function* $f : \Omega \to \mathbb{R}$ represent the density of an unknown 3D volume supported on $\Omega \subset \mathbb{R}^3$. From the frame of reference of the object, the formation of the projection on the detector is given by

$$P_\theta(u, v) = \int_0^1 f(l(t))\,\mathrm{d}t, \tag{2.1}$$

where $l(t) : [0, 1] \to \mathbb{R}^3$ is the parameterization of the line segment connecting the position of the source at angle θ with position (u, v) on the detector.

In practice, only a finite number N_θ of projections are acquired on a discrete grid of $N_u \times N_v$ detector pixels of size $r \times r$. Hence, the projection data can be described by a vector $\mathbf{p} \in \mathbb{R}^m, m = N_\theta \times N_u \times N_v$. Likewise, the object f is represented by a three-dimensional voxel grid $\mathbf{x} \in \mathbb{R}^{N_\mathbf{x} \times N_\mathbf{x} \times N_\mathbf{x}}$ describing a physical volume $\Omega_\mathbf{x} \supseteq \Omega$. Here, solely for ease of exposition, we choose all grids to be cubes. The physical dimensions of a single voxel, the *voxel size*, is naturally determined by the number of voxels in $\mathbf{x}$ and choice of $\Omega_\mathbf{x}$, and can in principle be made arbitrarily small. The discrete formulation gives rise to the linear inverse problem

$$\mathbf{A}\mathbf{x} = \mathbf{p}, \tag{2.2}$$

with $\mathbf{A} = (a_{ij})$ an $m \times n$ matrix where a_{ij} represents the contribution of volume voxel j to detector pixel i. The goal of tomographic reconstruction is to estimate $\mathbf{x}$ such that it matches the $N_\mathbf{x} \times N_\mathbf{x} \times N_\mathbf{x}$ voxel discretization of f.

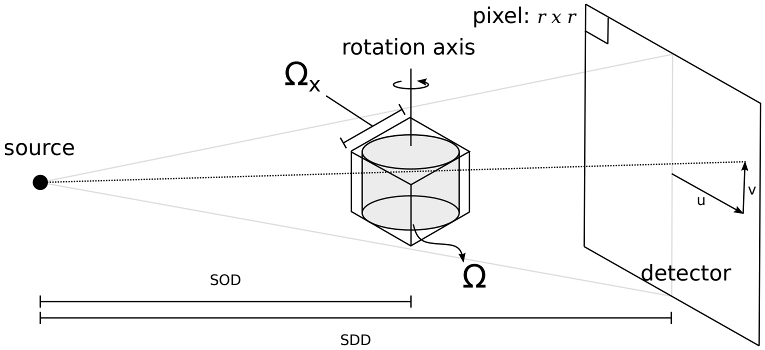


Figure 2.1: A schematic overview of the cone-beam geometry. The object is mounted on a rotation stage at distance SOD from the source. The object is supported on Ω, and the voxel grid $\mathbf{x}$ is supported on $\Omega_{\mathbf{x}}$. The detector is at distance SDD from the source and is divided into $N_u \times N_v$ pixels of size $r \times r$. We use the coordinates (u, v) to denote a location on the detector.

For the circular cone beam trajectory, several methods exist to compute $\mathbf{x}$ from projections $\mathbf{p}$. One of these methods is the widely used Feldkamp–Davis–Kress (FDK) algorithm [53], $\mathbf{FDK} : \mathbb{R}^{N_\theta \times N_u \times N_v} \to \mathbb{R}^{N_\mathbf{x} \times N_\mathbf{x} \times N_\mathbf{x}}$, which computes

$$\widetilde{\mathbf{x}} = \mathbf{FDK}\,(\mathbf{p}) = \mathbf{A}^T (\mathbf{h} * \widetilde{\mathbf{p}})_{1D}, \tag{2.3}$$

where $\mathbf{A}^T \in \mathbb{R}^{n \times m}$, the transpose of $\mathbf{A}$, is the discrete backprojection of $\mathbf{p}$ onto $\mathbf{x}$, $\mathbf{h} \in \mathbb{R}^{N_u}$ is the convolution kernel associated with the FDK algorithm, and $(\mathbf{h} * \widetilde{\mathbf{p}})_{1D}$ is the horizontal one-dimensional convolution of $\mathbf{h}$ with $\widetilde{\mathbf{p}}$, a weighted version of $\mathbf{p}$ with diminished intensity at higher distance from the detector center. The FDK algorithm is computationally efficient and results in accurate reconstructions when the projection data have a low noise profile and have been acquired from a sufficient number of angles.

The resolution of the reconstruction cannot be arbitrarily improved by choosing a finer reconstruction grid. Even in the absence of noise and other effects, the minimal voxel size is limited by the detector pixel size [26]

$$v = r \cdot \frac{\mathrm{SOD}}{\mathrm{SDD}}. \tag{2.4}$$

The angular resolution of the CT scan also influences the voxel size. The number of angles from which the projection data has been acquired also influences the minimal voxel size, but, when a sufficiently large number of angles has been used, the voxel size as determined in Equation (2.4) is not further limited [95].

In addition to the detector pixel size, multiple optical phenomena influence the formation of projection images. Many of these optical effects introduce a blurring effect on the projection images. For instance, pixel cross-talk causes an element of the photo-multiplier array to measure some fraction of the incoming photons of its neighboring pixels [147]. Cross-talk can occur both in the scintillator, where the high-energy photons are converted into the visible spectrum, and in the photo-sensor itself. Blurring can also be introduced when the focal spot, the region where the X-rays originate from, is a larger disc-like region rather than a point source [31, Chapter 9]. These optical effects can be modeled by a *point spread function* (PSF), e.g., a 2D Gaussian filter, describing the projection on the detector of a point-like object. A consequence of these optical effects is that the effective resolution on the detector is lower than the pixel resolution, thereby increasing the effective voxel size [31].

Even though detector resolution is limited, Equation (2.4) shows that the voxel resolution can be increased by modifying the acquisition geometry. The projection of a feature located on the rotation axis has a *magnification factor* of

$$\alpha = \frac{\mathrm{SDD}}{\mathrm{SOD}}. \tag{2.5}$$

By moving the object closer to the source, this magnification factor is increased, which, thereby, as a consequence of Equation (2.4), decreases the minimal voxel size. As the object is moved closer to the source, the projections of the object may become *truncated.* Consequently, only a part of the object, the *region of interest (ROI)*, is consistently projected onto the detector at every angle. This causes *truncation artifacts* in the reconstruction. Various techniques exist to minimize these artifacts [98, 176, 180, 194]. Nonetheless, these methods only improve resolution in the region of interest, hence new methods are required to reconstruct the entire volume at high resolution.

2.1.2 Deep Convolutional Neural Networks

A recently proposed strategy to improve the resolution of photographic images is the use of deep *convolutional neural networks* (CNNs) [106]. This class of machine learning algorithms processes the image by applying multiple convolution kernels and saves the intermediate results in *layers.* Each layer consists of several images, and is computed by convolving the previous layer with learned kernels, adding a scalar bias term, and applying a nonlinear activation function to each pixel. The network

$$\mathrm{Net}_{C,\varphi} : \mathbb{R}^{C\times N\times N} \to \mathbb{R}^{N\times N} \tag{2.6}$$

is *trained* to find a set of parameters $\hat{\varphi} \in \Phi$ such that it best transforms a stack of C images into a desired image of $N \times N$ pixels. In modern CNN architectures, the number of parameters can range up to millions.

The training procedure applies the network to a *training set* of *input data*, $(x_i)_{i=1}^n$, and compares the output to *target data* $(z_i)_{i=1}^n$, where the goal is to minimize a user-specified *empirical loss function* L. Here, we use the pixel-wise mean square error

$$L(\varphi) = \frac{1}{n}\sum_{i=1}^{n} \|\mathrm{Net}_{C,\varphi}(x_i) - z_i\|_2^2. \tag{2.7}$$

Training is not usually carried out to completion, i.e., the training is stopped before the absolute minimum of L is attained on the training set. This is done to prevent *overfitting* the network to the training data. When the network is overfit, the empirical loss is low for data in the training set, but high for data not included in the training set. The performance of a network that is being trained is commonly tested on a *validation set* that is not part of the training set. Once the performance on the validation set stops improving, training is stopped. An alternative to using a validation set is *early stopping*, where training is stopped at some predetermined time.

Most existing state-of-the-art CNNs process 2D images. To process 3D volumes, we can subdivide the input and target voxel grids $\widetilde{\mathbf{x}}^{\text{input}}$ and $\widetilde{\mathbf{x}}^{\text{target}}$ into 2D horizontal *slices*. As input, the network is provided with a *slab* containing not just a single input slice but also surrounding slices, supplying the network with *quasi-3D* information. Although network architectures exist that process entire 3D volumes at once, their memory requirements are challenging when applied to voxel grids with sizes that are common in tomography.

2.2 Method

In this section, we describe the main contribution of this chapter: an integrated data acquisition and machine learning approach for improving resolution in cone-beam tomography. Our approach consists of several steps. First, we perform a standard CT scan of the object under study. In addition to a standard CT scan, we acquire additional projection images with the object moved closer to the radiation source. Second, we reconstruct the entire volume at low resolution. Next, using both projection datasets, we reconstruct a region of interest at high resolution. These two reconstructions are used to train a neural network to compute a mapping between the low-resolution volume in the region of interest and its high-resolution counterpart. The trained neural network is then applied to the entire low-resolution volume to obtain a final high-resolution volume representation. This process is illustrated in Figure 2.2. In the next subsections, we describe the steps of our data acquisition and reconstruction strategy in detail.

2.2.1 Data Acquisition

Our data acquisition protocol consists of two scans. The first scan with magnification factor $\alpha_{\text{low}} = {}^{SDD}/_{SOD}$ is a standard circular cone beam scan and yields projection dataset $\mathbf{p}^{\text{low}}$, as shown in Figure 2.1. In addition to the standard acquisition, we propose to acquire an additional projection dataset $\mathbf{p}^{\text{high}}$. Here, the object is moved closer to the source such that the resulting projection on the detector has a magnification factor of $\alpha_{\text{high}} = {}^{SDD}/_{SOD_2}$. Now, only a region of interest $\Omega_{\text{ROI}} \subseteq \Omega \subseteq \mathbb{R}^3$ is in full view on the detector at all times, as is depicted in Figure 2.3. The addition of the second projection dataset $\mathbf{p}^{\text{high}}$ permits the

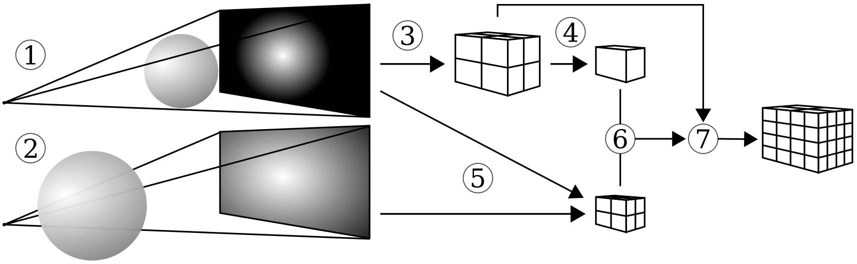


Figure 2.2: Method summary. We make a high- and low-resolution reconstruction of a region of interest to train a neural network to obtain a high-resolution representation of the entire object. (1) Acquire projection data; (2) Acquire zoomed in projection data; (3) Reconstruct on a coarse grid; (4) Restrict to a region of interest; (5) Reconstruct the region of interest on a fine grid using both projection datasets; (6) Train a neural network to transform the low-resolution region of interest to a high-resolution volume; (7) Apply the network to the entire low-resolution volume.

high-resolution reconstruction of a rectangular sub-region of the region of interest. This is described below.

2.2.2 Reconstruction

From the acquired datasets, two reconstructions are computed. The first reconstruction contains the entire object at low resolution, and the second reconstruction describes the region of interest at high resolution. Because these reconstructions are later used for the training of a neural network, we ensure that their voxel grids are aligned.

The dimensions of the low-resolution reconstruction grid are determined using the pixel size r and the physical dimensions Ω of the object. We calculate the effective voxel size $v_{\text{low}} = {}^{r}/_{\alpha_{\text{low}}}$ using Equations (2.4) and (2.5). Given the voxel size, the shape of the voxel grid $\mathbf{x}^{\text{low}}$, $N_{\text{low}} \times N_{\text{low}} \times N_{\text{low}}$, is established such that its real-world volume $\Omega_{\text{low}} \subset \mathbb{R}^3$ covers the object, i.e., $\Omega \subseteq \Omega_{\text{low}}$. On this grid, the low-resolution reconstruction is computed using the FDK algorithm

$$\tilde{\mathbf{x}}^{\text{low}} = \mathbf{FDK}\left(\mathbf{p}^{\text{low}}\right) \in \mathbb{R}^{N_{\text{low}}^3}. \tag{2.8}$$

Next, we determine the shape and voxel size v_{high} of the high-resolution voxel grid $\mathbf{x}^{\text{high}}$. We set the voxel size to equal $v_{\text{high}} = {}^{v_{\text{low}}}/_{k}$, where k equals ${}^{\alpha_{\text{high}}}/_{\alpha_{\text{low}}}$ rounded to the nearest integer. In practice, magnification factors α_{low} and α_{high} can be carefully chosen to avoid the need for rounding. The voxel size v_{high} is close to the limit defined by Equation (2.4), since, by Equation (2.5), we have

$$v_{\text{high}} = \frac{v_{\text{low}}}{k} \approx v_{\text{low}} \frac{\alpha_{\text{low}}}{\alpha_{\text{high}}} = \frac{r}{\alpha_{\text{low}}} \frac{\alpha_{\text{low}}}{\alpha_{\text{high}}} = \frac{r}{\alpha_{\text{high}}}. \tag{2.9}$$

We fix the physical volume of the high-resolution grid, Ω_{high}, to be contained in the region of interest, i.e., $\Omega_{\text{high}} \subseteq \Omega_{\text{ROI}}$, and set its shape, $N_{\text{high}} \times N_{\text{high}} \times N_{\text{high}}$,

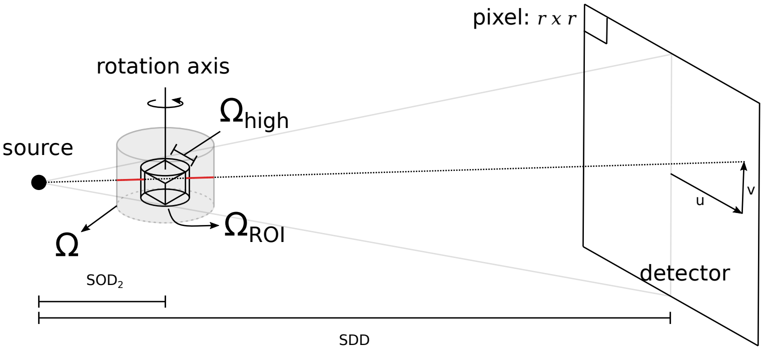


Figure 2.3: An illustration of the second tomographic scan. The object, supported on Ω, is scanned at magnification $\alpha_{\text{high}} = {}^{SDD}/_{SOD_2}$. Now, only the region of interest, Ω_{ROI}, is always in full view of the detector. The high-resolution reconstruction grid $\mathbf{x}^{\text{high}}$ is thus constrained to the rectangular volume Ω_{high}. Part of the object outside of Ω_{high} contributes to the acquired projection data. This is highlighted in red.

such that N_{high} is divisible by k. Now, voxels from $\mathbf{x}^{\text{low}}$ relate to cubes of $k \times k \times k$ voxels in $\mathbf{x}^{\text{high}}$. This assists in modeling the forward projection of the second scan.

As shown in Figure 2.3, the rays forming the projection data, $\mathbf{p}^{\text{high}}$, have been transmitted through both Ω_{high} and $\Omega \setminus \Omega_{\text{high}}$. This is modeled by the discrete linear equation

$$\mathbf{p}^{\text{high}} \approx \mathbf{A}^{(\text{high})}\mathbf{x}^{\text{high}} + \mathbf{A}^{(\text{low}\to\text{high})}\mathbf{M}\,\mathbf{x}^{\text{low}}, \tag{2.10}$$

where $\mathbf{M} \in \mathbb{R}^{N_{\text{low}}^3 \times N_{\text{low}}^3}$ is a matrix that masks all voxels in $\mathbf{x}^{\text{low}}$ that are contained in Ω_{high}. In other words, $\mathbf{M}$ is a diagonal matrix where the diagonal is 0 on row i if voxel i of the large grid $\mathbf{x}^{\text{low}}$ overlaps Ω_{high}, and is 1 elsewhere. The matrix $\mathbf{A}^{(\text{low}\to\text{high})} = (a_{ij})$ is defined such that a_{ij} represents the contribution of the low-resolution volume voxel $\mathbf{x}_j^{\text{low}}$ to detector pixel $\mathbf{p}_i^{\text{high}}$ at magnification α_{high}. The matrix $\mathbf{A}^{(\text{high})}$ similarly defines the projection of the high-resolution volume $\mathbf{x}^{\text{high}}$ at magnification α_{high}.

To reconstruct $\mathbf{x}^{\text{high}}$, we subtract the reprojection of the masked reconstruction $\mathbf{M}\,\tilde{\mathbf{x}}^{\text{low}}$ from the acquired high-resolution projection data

$$\hat{\mathbf{p}} = \mathbf{p}^{\text{high}} - \mathbf{A}^{(\text{low}\to\text{high})}\mathbf{M}\,\tilde{\mathbf{x}}^{\text{low}}. \tag{2.11}$$

Next, we apply the FDK algorithm to the processed projection data to obtain the reconstruction

$$\tilde{\mathbf{x}}^{\text{high}} = \mathbf{FDK}\left(\hat{\mathbf{p}}\right) \in \mathbb{R}^{N_{\text{high}}^3}. \tag{2.12}$$

In a conventional FDK reconstruction, the outside contributions of $\Omega \setminus \Omega_{\text{high}}$ to the projection data $\mathbf{p}^{\text{high}}$ cause truncation artifacts. Because we have approximately removed these contributions, we can apply the FDK algorithm to obtain a high-resolution reconstruction of the region of interest. Note that similar subtraction methods have been proposed before to remove truncation artifacts in region-of-interest tomography [114, 176, 180]. Some artifacts may occur on the border of the region of interest, which may be due to the specifics of the interpolation kernel that is used in the forward projection $\mathbf{A}^{(\text{low}\to\text{high})}$ or due to discontinuity at the image borders enhanced by the FDK filter kernel. Water cylinder extrapolation or truncation robust FBP methods like [45] may alleviate these artifacts. These artifacts are not a problem in practice, as the affected border voxels may be ignored.

2.2.3 Machine Learning

At this point in the process, the tomographic reconstruction method yields a low-resolution voxel grid $\widetilde{\mathbf{x}}^{\text{low}}$ and a high-resolution voxel grid $\widetilde{\mathbf{x}}^{\text{high}}$. For the neural network to match the low-resolution input to the high-resolution target, it is necessary that both input and target relate to the same physical space, and have the same voxel size. In this section, we outline how $\widetilde{\mathbf{x}}^{\text{low}}$ and $\widetilde{\mathbf{x}}^{\text{high}}$ are processed to match in physical space and voxel size. We also describe the training procedure, and motivate the choice of neural network architecture.

The voxel grid $\widetilde{\mathbf{x}}^{\text{low}}$ corresponds to the physical space Ω_{low} and $\widetilde{\mathbf{x}}^{\text{high}}$ corresponds to Ω_{high}. To serve as input data for neural network training, we use the voxels from $\widetilde{\mathbf{x}}^{\text{low}}$ that are contained in Ω_{high}. We denote this restriction operation by $\mathbf{R} : \mathbb{R}^{N_{\text{low}}^3} \to \mathbb{R}^{N_{\text{high}}^3/k^3}$. The resulting voxel grid

$$\widetilde{\mathbf{x}}_{\text{ROI}}^{\text{low}} = \mathbf{R}\left(\widetilde{\mathbf{x}}^{\text{low}}\right) \in \mathbb{R}^{N_{\text{high}}^3/k^3} \tag{2.13}$$

corresponds to the physical volume Ω_{high}, and each of its voxels can thus be matched with voxels from $\widetilde{\mathbf{x}}^{\text{high}}$.

The voxel sizes of $\widetilde{\mathbf{x}}_{\text{ROI}}^{\text{low}}$ and $\widetilde{\mathbf{x}}^{\text{high}}$ can be equalized by either up-sampling the input data $\widetilde{\mathbf{x}}_{\text{ROI}}^{\text{low}}$ or down-sampling the target data $\widetilde{\mathbf{x}}^{\text{high}}$. We present both options, which we designate as method A and method B, respectively. By up-sampling the input data with an interpolation method $\mathbf{U}_k : \mathbb{R}^{N_{\text{high}}^3/k^3} \to \mathbb{R}^{N^3}$ by a factor k with $N = N_{\text{high}}$, we obtain training data on a fine grid

$$\widetilde{\mathbf{x}}^{\text{input}} = \mathbf{U}_k\left(\widetilde{\mathbf{x}}_{\text{ROI}}^{\text{low}}\right) \in \mathbb{R}^{N^3}, \qquad \text{(method A)} \tag{2.14}$$

$$\widetilde{\mathbf{x}}^{\text{target}} = \widetilde{\mathbf{x}}^{\text{high}} \in \mathbb{R}^{N^3}. \tag{2.15}$$

By down-sampling the target data by a factor k with an interpolation method $\mathbf{D}_k : \mathbb{R}^{N_{\text{high}}^3} \to \mathbb{R}^{N^3}$ with $N = N_{\text{high}}/k$, we obtain training data on a coarse grid

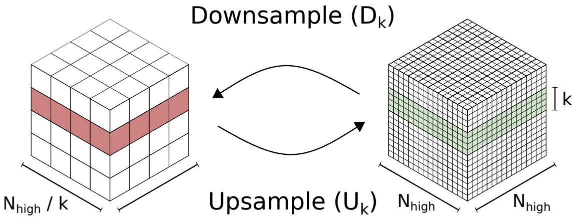


Figure 2.4: Two voxel grids at different resolution. The low-resolution grid on the left can be up-sampled ($\mathbf{U}_k$) to a high-resolution grid. This transforms a *slice* (in red, left) into a *slab* (in green, right), which consists of multiple slices. The down-sampling operation ($\mathbf{D}_k$) performs the reverse transformation. Here, the up-sampling and down-sampling are performed with magnification factor $k = 4$, which increases or decreases the number of voxels in each dimension by a factor of k.

$$\tilde{\mathbf{x}}^{\text{input}} = \tilde{\mathbf{x}}^{\text{low}}_{\text{ROI}} \in \mathbb{R}^{N^3}, \qquad \text{(method B)} \tag{2.16}$$

$$\tilde{\mathbf{x}}^{\text{target}} = \mathbf{D}_k \left(\tilde{\mathbf{x}}^{\text{high}}\right) \in \mathbb{R}^{N^3}. \tag{2.17}$$

The interpolation methods $\mathbf{U}_k$ and $\mathbf{D}_k$ are open to choice. Various methods can be used, including, for instance, cubic interpolation. The up- and down-sampling operations are illustrated in Figure 2.4.

The properties of the acquired projection data can be used to inform the choice whether to up-sample the input (method A), or down-sample the target (method B). When optical blurring on the detector is negligible, the small-scale features that can be seen in $\tilde{\mathbf{x}}^{\text{high}}$ will be hard to visualize on the coarse grid of $\tilde{\mathbf{x}}^{\text{low}}_{\text{ROI}}$. Therefore, it is recommended to use method A where $\tilde{\mathbf{x}}^{\text{target}}$ is defined on a fine grid. If, on the other hand, optical blurring limits the effective resolution of $\tilde{\mathbf{x}}^{\text{low}}_{\text{ROI}}$, and its resolution can be improved without reducing the voxel size, then it is recommended to use method B, where the grid size is smaller, and training and applying the network is thus computationally faster. The differences between the input and target grids are summarized in Table 2.1.

The input and target voxel grids $\tilde{\mathbf{x}}^{\text{input}}$ and $\tilde{\mathbf{x}}^{\text{target}}$ serve to train the convolutional neural network. The voxel grids $\tilde{\mathbf{x}}^{\text{input}}$ and $\tilde{\mathbf{x}}^{\text{target}}$ are divided into slices $\tilde{\mathbf{x}}^{\text{input}}_i, \tilde{\mathbf{x}}^{\text{target}}_i, i = 1, \ldots, N$ as illustrated in Figure 2.4. In this chapter, $\mathbf{x}_i$ denotes both a single voxel at position i and a horizontal slice i of a voxel grid $\mathbf{x}$. The surrounding text always distinguishes between a voxel and a slice.

The input slices are combined into slabs $\left[\tilde{\mathbf{x}}^{\text{input}}_j\right]_{j=i-s}^{i+s}, i = s+1, \ldots, N-s$, containing the input slice and the s slices above and below. The target slices are not combined into slabs. The training procedure now minimizes

$$L_s(\varphi) = \frac{1}{N} \sum_{i=s+1}^{N-s} \| \text{Net}_{2s+1,\varphi} \left(\left[\tilde{\mathbf{x}}^{\text{input}}_j\right]_{j=i-s}^{i+s} \right) - \tilde{\mathbf{x}}^{\text{target}}_i \|_2^2, \tag{2.18}$$

	Method A	**Method B**
Training		
Grid size (N^3)	N^3_{high}	N^3_{high}/k^3
Input ($\widetilde{\mathbf{x}}^{\text{input}}$)	$\mathbf{U}_k(\widetilde{\mathbf{x}}^{\text{low}}_{\text{ROI}})$	$\widetilde{\mathbf{x}}^{\text{low}}_{\text{ROI}}$
Target ($\widetilde{\mathbf{x}}^{\text{target}}$)	$\widetilde{\mathbf{x}}^{\text{high}}$	$\mathbf{D}_k(\widetilde{\mathbf{x}}^{\text{high}})$
Processing		
Grid size	$(kN_{\text{low}} - 2s)^3$	$(N_{\text{low}} - 2s)^3$
Input	$\mathbf{U}_k(\widetilde{\mathbf{x}}^{\text{low}})$	$\widetilde{\mathbf{x}}^{\text{low}}$

Table 2.1: An overview of the grid sizes of method A, where the input is up-sampled, and method B, where the target is down-sampled. The grid size and input are tabulated for the training phase and the processing phase, in which the final output is calculated. The grid size is in number of voxels. The processing grid size is the size of the output $\widetilde{\mathbf{x}}^{\text{final}}$ when the network is applied to $\widetilde{\mathbf{x}}^{\text{low}}$, and s is the number of additional input slices that the network takes as input.

which yields the set of parameters, $\hat{\varphi}$. Other norms, such as the L1 norm, could also be used.

Multiple deep network architectures can be used to improve the quality of reconstructions [81, 143, 156, 165]. One of these is the mixed-scale dense (MS-D) network, which is a neural network architecture specifically developed for scientific settings. The mixed-scale dense network has several properties that make it well-suited for our method. The MS-D network has relatively few parameters compared to other neural network structures, making it easier to train and less likely to overfit. This is especially useful if a small training set is available, and no part of the training set is sacrificed to serve as a validation set, as is the case here. Another advantage is that the MS-D network maximally reuses the intermediate images in each layer, thus requiring fewer intermediate images compared to other deep convolutional neural network architectures. Therefore, the MS-D network can transform large images using less memory and computation time compared to other popular convolutional neural network architectures [143]. Finally, the network has shown good results for removing noise and other artifacts from tomographic images when a large training set of similar objects scanned at high dose is available. This is described in [140], where the network was applied to tomographic images reconstructed from a dataset of parallel beam projections, rather than cone beam projections.

2.2.4 Improving Resolution

After training the network, a set of network parameters $\hat{\varphi}$ is obtained. The trained neural network $\text{Net}_{\hat{\varphi}}$ is applied to the initially reconstructed low-resolution volume $\widetilde{\mathbf{x}}^{\text{low}}$ to obtain a final volume representation $\widetilde{\mathbf{x}}^{\text{final}}$. This process consists of several steps. If the training input has been up-sampled using method A, then the reconstruction $\widetilde{\mathbf{x}}^{\text{low}}$ must likewise be up-sampled to $\mathbf{U}_k(\widetilde{\mathbf{x}}^{\text{low}})$, which is k times larger in every dimension. If method B has been used, no up-sampling of the reconstruction is necessary. The resulting voxel grid is processed by the network slice by slice. These slices must be collected in slabs containing s slices above and below the input slice. This influences the output size, since the top and bottom s slices of $\widetilde{\mathbf{x}}^{\text{low}}$ do not have enough surrounding slices to be processed by the network. This is usually not a problem because s is small ($s < 10$) and the low-resolution grid can be chosen large enough that any important features are at some distance from the top and bottom. Finally, the network can be applied to each of these slabs in turn, the result of which we collect into $\widetilde{\mathbf{x}}^{\text{final}}$. In summary, this process results in the final volume representation

$$\widetilde{\mathbf{x}}^{\text{final}} = \left[\text{Net}_{2s+1,\hat{\varphi}}\left(\left[U_K\left(\widetilde{\mathbf{x}}^{\text{low}}\right)_j\right]_{j=i}^{i+2s}\right)\right]_{i=1}^{N_{\text{final}}} \qquad \text{(Method A)},$$

$$\widetilde{\mathbf{x}}^{\text{final}} = \left[\text{Net}_{2s+1,\hat{\varphi}}\left(\left[\widetilde{\mathbf{x}}_j^{\text{low}}\right]_{j=i}^{i+2s}\right)\right]_{i=1}^{N_{\text{final}}} \qquad \text{(Method B)}.$$

For both methods, the size of the final grid $\widetilde{\mathbf{x}}^{\text{final}}$ is summarized in Table 2.1.

2.3 Results

We evaluate the accuracy of our method on simulated and experimental data, which is described below. Before going through the specifics of each experiment, we first provide details on the steps that all experiments have in common.

Data acquisition For each experiment, three projection datasets are acquired. The first projection dataset has the entire object in the field of view and has a low magnification factor α_{low}. The second and third projection datasets are acquired of a central and upper region of interest and have a higher magnification factor α_{high}. This is illustrated in Figure 2.5.

Reconstruction In all experiments, high-resolution reconstructions are computed of a central and upper region of interest to serve as target data for the *training set* and *test set*, respectively. A low-resolution reconstruction is computed of the entire object, which serves as input data for both training and testing. In all experiments, the low-resolution voxels are $k = 4$ times as large as the high-resolution voxels. FDK reconstructions were computed using the ASTRA toolbox [2].

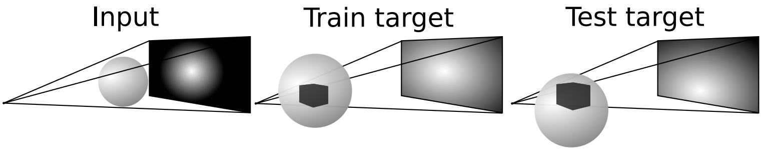


Figure 2.5: For each object, three projection datasets are acquired. The first dataset has the entire object in the field of view, and its reconstruction is used as input data for the learning step. The second dataset has a central region of interest in view (dark gray), the reconstruction of which is used as target data for the learning step. The third dataset has an upper region of interest in view (dark gray), the reconstruction of which is used as target data for evaluation.

Up- and down-sampling The up- and down-sampling of the reconstructed volumes can be performed in various ways. In the experiments, up-sampling in method A is carried out by nearest-neighbor up-sampling, where each voxel is repeated k times in each direction. Empirically, we found this to be sufficient, as cubic up-sampling did not improve the quality of the network output. Down-sampling of the target voxel grid in method B is carried out using three-dimensional cubic interpolation, where the image is interpolated to a coarser grid using polynomials of degree at most 3 determined by a window of $4 \times 4 \times 4$ voxels.

Neural network implementation For each experiment, three separate networks are trained using: (i) method A and a single input slice, (ii) method A and a slab of nine input slices, (iii) method B and a single input slice. The MS-D network is implemented in PyTorch [139]. Each trained network has 100 single-channel intermediate layers, and the convolution in layer i is dilated by $d_i = 1 + (i \mod 10)$, as is described in [143]. The network has 45,652 parameters when there is a single input slice, and it has 52,948 parameters when the input is a slab of nine slices.

Training procedure The network is trained on the central region of interest. Because we chose not to use a validation set, a criterion was set for early stopping. In all experiments, training finished after two days or 1000 epochs, whichever came first. All networks are trained from scratch in each experiment. The training procedure minimizes the mean square error between the output and target images, and the networks are trained using the ADAM algorithm [94] with a mini-batch size of one. The network output is evaluated on a test set containing the low- and high-resolution reconstructions of the upper region of interest. The training and test set are thus non-overlapping.

Metrics and evaluation On both datasets, the on-the-fly machine learning approach is evaluated using the structural similarity index (SSIM) [191] and the mean square error (MSE) metrics. These metrics are used to compare the output slices of the networks to target slices from the upper ROI reconstruction. To compare methods A and B on the same grid, the output volume of method B is cubically up-sampled before computing the MSE and SSIM metrics. The output slices of method A are not processed, since they are the same size as the target slices of the test set. Both metrics are computed slice by slice and averaged. To prevent the influence of reconstruction artifacts at the boundaries of the high-resolution reconstructions, all metrics are calculated on pixels that are at least eight pixels from the boundary of the volume. Likewise, all displayed images are cropped to remove eight pixels from all sides. The error metrics of our approach are compared to a baseline: a full-volume three-dimensional cubic up-sampling of the low-resolution input volume. We visually compare all methods on the central slice of the upper ROI reconstruction on both the simulated and the experimental data.

All computations were performed on a server with 192 GB of RAM and four Nvidia GeForce GTX 1080 Ti GPUs (Nvidia, Santa Clara, CA, USA) or on a workstation with 64 GB of RAM and one Nvidia GeForce GTX 1070.

	Foam	**Foam**	**Oatmeal**
	Pixel-Limited	**Optics-Limited**	
Detector			
Shape	1000 × 1000	1000 × 1000	1536 × 1944
Pixel size	0.0012	0.0012	75 μm
Number of angles	1500	1500	2000
Full object			
Grid shape	1064 × 1064 × 1064	1064 × 1064 × 1064	1647 × 2084 × 2084
Voxel size	0.0012	0.0012	68.26 μm
Central ROI			
Grid shape	888 × 664 × 664	888 × 664 × 664	1364 × 1304 × 1304
Voxel size	0.0003	0.0003	17.07 μm
Top ROI			
Grid shape	880 × 664 × 664	880 × 664 × 664	1364 × 1304 × 1304
Voxel size	0.0003	0.0003	17.07 μm
Training epochs			
Method A 9 slices	230	230	30
Method A 1 slice	260	250	40
Method B 1 slice	1000	1000	950

Table 2.2: A summary of the pixel and voxel grids used for the reconstructions of the full object and regions of interest (ROIs). For each dataset, the network was trained by up-sampling the input (method A) with a slab of nine slices and one slice as input, and by down-sampling the target (method B) with a slab of one slice as input.

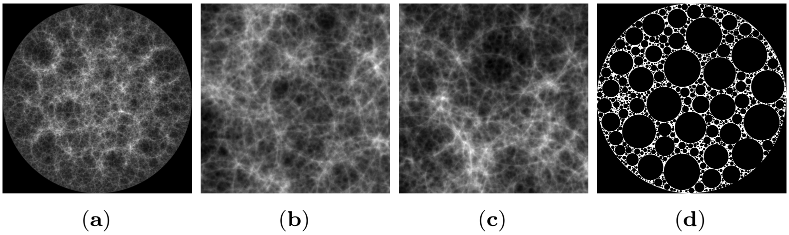
2.3.1 Simulations

First, we investigated the performance of the proposed on-the-fly machine learning technique on simulated tomographic data.

Simulation phantom A foam ball simulation phantom was generated by removing 90,000 randomly-placed non-overlapping spheres from a large sphere made of one material. The foam ball has diameter 1. All other dimensions in the simulation are relative to this unit length. The radius of the random spheres ranges between 0.0025 and 0.2. The central slice of this phantom is displayed in Figure 2.6.

Data simulation Using the foam phantom, three projection datasets were computed. For the first projection dataset, which has the entire foam ball in the field of view, the source-object distance is equal to the source-detector distance, yielding a magnification factor of $\alpha_{\text{low}} = 1$. In practice, having an equal source-object and source-detector distance is not possible, since the detector and object would share the same physical space. In simulations, however, this is both possible and natural, since it results in a minimal voxel size that is equal to the detector pixel size. The second and third projection dataset were acquired at a magnification factor of $\alpha_{\text{high}} = 4$.

To simulate how our method copes with optical phenomena, another set of projections was created by post-processing the projections using a Gaussian blur



(a) (b) (c) (d)

Figure 2.6: (**a**) A projection image of the foam ball; (**b**) a magnified projection image of the central region of interest (ROI); (**c**) a magnified projection image of the upper ROI; (**d**) the central cross-sectional slice of the phantom.

with standard deviation of two pixels. The Gaussian blur convolves the projection images with a 2D filter defined by $g(x) = \exp\left(-\|x\|^2/2\sigma^2\right)/\sqrt{2\pi\sigma^2}$, where σ is the chosen standard deviation. We refer to the blurred data as *optics-limited*, and to the non-blurred data as *pixel-limited*.

The projections were carried out using the GPU-accelerated *cone_ balls* software package, which we have made available as an open source package [67]. This package analytically computes the linear cone beam projection of solid spheres of constant density. For each dataset, 1500 projections were acquired over 360 degrees on a virtual detector with 1000 × 1000 pixels. For each detector pixel, four rays were cast through the phantom, and their projection values were averaged.

Processing The reconstruction and training of the foam phantom was performed as described before in Section 2.3. The details are summarized in Table 2.2. Methods A and B are evaluated on the upper ROI reconstruction, and compared below for both the pixel-limited and optics-limited case.

Evaluation For the simulated data, the MSE and SSIM metrics are computed between the network outputs and the original phantom data, rather than the high-resolution reconstruction. As the SSIM metric is designed to operate on images with a fixed intensity interval, we clip the network outputs such that all images have the same minimum and maximum intensity as the phantom data. Similarly, the images displayed in Figure 2.7 are clipped to the range $[0, 1]$. The MSE metrics are calculated on the unclipped data.

The quantitative results for the foam phantom are given in Table 2.3. Both

	Pixel-limited		**Optics-limited**	
Method	**MSE**	**SSIM**	**MSE**	**SSIM**
Method A: 9 slices in slab	**0.0044**	**0.9578**	**0.0143**	**0.8275**
Method A: 1 slice in slab	0.0089	0.9281	0.0154	0.8220
Method B: 1 slice in slab	0.0111	0.8065	0.0178	0.8135
Cubic up-sampling	0.0125	0.6893	0.0463	0.6038

Table 2.3: Foam ball phantom: comparison of the MSE and SSIM between the output of the three approaches described in this section and cubic up-sampling. For each dataset and metric, the best results are shown in bold.

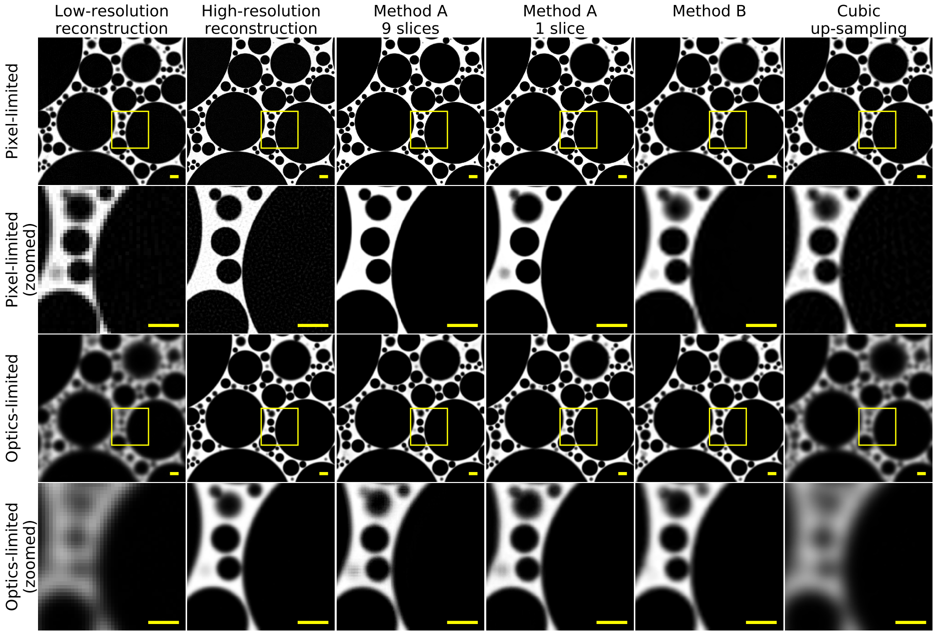


Figure 2.7: Results of applying the trained networks to the central slice of the upper region of interest. The low-resolution slice (shown left) was not part of the training set. The high-resolution slice is the result of reconstruction (Equation (2.12)). The images in the right-most four columns are computed from the input using our proposed methods and cubic up-sampling. Magnifications of the central yellow square are displayed in the even rows. The first row displays the results on the non-blurred foam phantom reconstructions, and the third row displays the results on reconstructions of blurred foam phantom projections. The output of method B is cubically up-sampled by a factor of 4 to be the same size as the other images.

variants of method A significantly outperform cubic up-sampling on both MSE and SSIM metrics. On the pixel-limited dataset, the SSIM score of method B is lower than the methods A but higher than cubic up-sampling. The fine scale features in the pixel-limited high-resolution image can simply not be represented on the coarser grid that method B operates on. On the optics-limited dataset, on the other hand, the SSIM score of method B is comparable to the SSIM scores of methods A, and is significantly higher than that of cubic up-sampling.

In Figure 2.7, the methods are visually compared on both the pixel-limited dataset and the optics-limited dataset. In the pixel-limited dataset, the low-resolution input suffers from partial volume effects, where some voxels contain more than one material. In this case, the foam and void contributions to a voxel are averaged, which leads to jagged edges. The high-resolution data are significantly sharper, but still has some non-smooth texture in the foam and voids. The output of method A with nine input slices makes the edges as sharp as in the high-resolution image, and removes the non-smooth texture in the foam and voids. Moreover, it does not introduce any voids where there are none in the target data. Method A with one input slice performs similarly, but has some difficulty with features that are rapidly introduced in the vertical direction. This is apparent in the magnified image in Figure 2.7, where the top or bottom of one void in the input data are not correctly removed. As discussed before, method B performs worse than method A when the high-resolution features cannot be represented on the coarse grid.

In the optics-limited dataset, both low- and high-resolution slices are less sharp than in the pixel-limited case. Due to the additional blur on the projection data, some of the smaller features in the low-resolution image cannot easily be distinguished. Nonetheless, the output of both methods A has significantly improved resolution and is visually similar to the target slice. With few exceptions, all voids in the output are round, and, without exception, all voids in the output are also present in the target. Method B produces output that looks similar to the output of methods A, although it fuses some voids that can still be distinguished in the output of methods A.

2.3.2 Experimental Data

To verify the practical applicability of our approach, we have investigated the performance of our method on experimentally acquired cone-beam CT data.

Sample A plastic jar filled with oatmeal was scanned. The oatmeal was specifically chosen for its structure, which is consistent throughout the entire object. A package of oatmeal contains thousands of flakes, which all have roughly the same dimensions and shape, but still exhibit fine-scale features, requiring high-resolution tomography to accurately capture. Projection and reconstruction images of the oatmeal are displayed in Figure 2.8.

Data acquisition The projection images were acquired using the custom-built and flexible CT scanner, FleX-ray Laboratory, developed by XRE NV and located at CWI [42]. The apparatus consists of a cone-beam microfocus X-ray point source that projects polychromatic X-rays onto a 1944 × 1536 pixels, 14-bit, flat detector panel. The data was acquired over 360 degrees in circular motion with 2000 projections distributed evenly over the full circle. The projections were

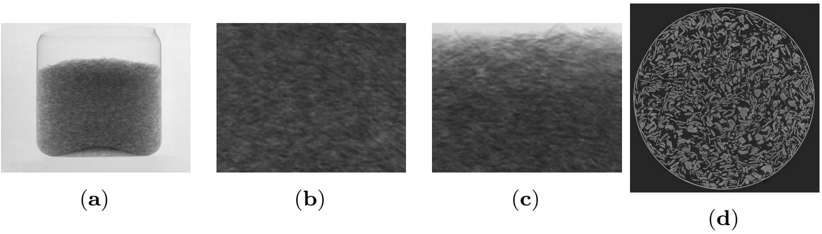

(**a**) (**b**) (**c**) (**d**)

Figure 2.8: (**a**) A projection image of the oatmeal sample; (**b**) a magnified projection image of the central ROI; (**c**) a magnified projection image of the upper ROI; (**d**) a central slice of the oatmeal sample.

	Emitted Beam Energy (J)	**Scanning Time (Minutes)**
Full low-resolution scan	$\eta\times$ 27,720	11
High-resolution ROI scan	$\eta\times$ 27,720	11
Full high-resolution scan (Estimate for tiling)	$\eta\times$ 443,520	176

Table 2.4: The scanning times and emitted beam energy of the experiments. The third row contains an estimate of the duration of a full high-resolution scan with a detector tiling strategy, where a virtual large projection image is created by stitching the images for several detector positions. To achieve similar resolution as the high-resolution ROI scan, 16 detector positions would have to be stitched together. The emitted beam energy is used as a proxy for the radation dose absorbed by the object. The efficiency of the source η is multiplied by the power $P = 42W$ applied to the X-ray tube, and the scanning time in seconds.

collected with 250 ms exposure time and the total scanning time was 11 minutes per acquisition. The tube voltage was 70 kV and the tube power was 42 W. This acquisition strategy was performed three times with magnification factors $\alpha = 1.09, 4.38$, and 4.38. Examples of the acquired projection images are displayed in Figure 2.8 and an overview of scanning time and emitted beam energy is given in Table 2.4. The object was centered on the detector for the first two scans, and the object was moved down to capture a region of interest above the center of the object for the final scan. These data are publicly available via [40].

Processing Reconstructions were computed of the full object, a central region of interest (serving as training set), and an upper region of interest (serving as test set). An example of the central slice of the oatmeal is displayed in Figure 2.8d. The voxels in the full-object reconstruction are four times larger in each dimension than the voxels in the region-of-interest reconstructions. The training step was also performed in the same way as for the simulated data. The details are summarized in Table 2.2. Up-sampling the low-resolution region-of-interest slices to 1304×1304 pixels takes roughly 0.56 s per slice and 13 min in total using method A with one input slice. Using method B, up-sampling a single slice takes 0.03 s, and the entire region of interest can be up-sampled in less than a minute.

Evaluation The attenuation coefficients of the voxels in the high-resolution reconstruction are contained in the range $[-0.039, 0.105]$. To enable comparison of the MSE and SSIM metrics of the experimental data with the simulated data, we rescaled the attenuation coefficients of the target volume to the range $[0, 1]$, and used the same scaling factors to rescale the values of all other volumes. In addition, before calculating the SSIM, all volumes are clipped to the range $[0, 1]$, similar to the simulated data. Likewise, the slices displayed in Figure 2.9 are rescaled and clipped to the unit range.

The experimental results on the oatmeal dataset are visually compared on the central slice of the upper region of interest. This low-resolution input slice, which is displayed in Figure 2.9, is thus not part of the training set. The low- and high-resolution slices of the oatmeal both suffer from some noise. The high-resolution slice contains significantly more visible fine-scale features. The output

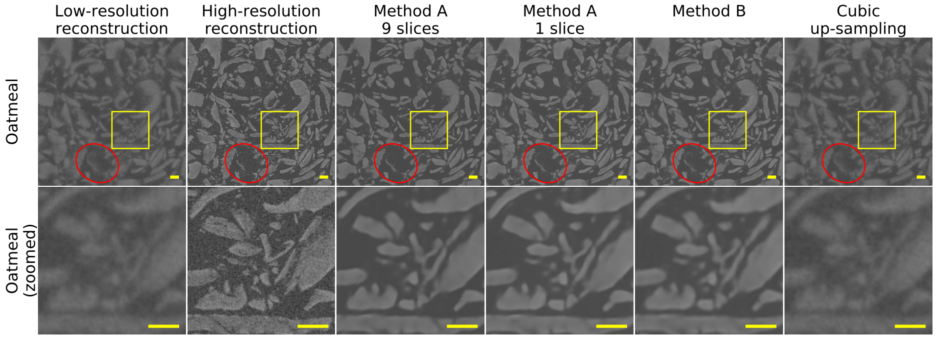


Figure 2.9: Oatmeal data: results of applying the trained networks to the central slice of the upper region of interest. The low-resolution slice (shown left) was not part of the training set. The high-resolution slice is the result of reconstruction (Equation (2.12)). The four images on the right are computed from the input using our proposed methods and cubic up-sampling. Magnifications of the central yellow square are displayed in the second row. The red ellipse highlights a flake that is partially visible in the input, and is correctly removed by all three methods. The output of method B is cubically up-sampled by a factor of 4 to be the same size as the other images.

Method	**MSE**	**SSIM**
Method A: 9 slices in slab	0.0027	0.4508
Method A: 1 slice in slab	0.0026	**0.4529**
Method B: 1 slice in slab	**0.0025**	0.4521
Cubic up-sampling	0.0035	0.4192

Table 2.5: Oatmeal: comparison of the MSE and SSIM between the output of the three approaches described in this section and cubic up-sampling. The best results are shown in bold for each metric.

of the three methods is virtually indistinguishable in the non-magnified images. In the magnified images, we see that not all fine details can be retrieved: some small features are removed and others are slightly deformed compared to the target. In the red ellipse, a large flake is visible in the low-resolution slice that appears larger than it really is due to partial volume effects. The high-resolution slice, on the other hand, shows only a small fragment of the large flake. All three networks are able to filter away a large part of the flake and retain the ridge that is visible in the high-resolution slice. The three methods significantly reduce the background noise that is present in the high-resolution target beyond what can be achieved using cubic up-sampling. Overall, all three methods sharpen features that are hard to distinguish in the input, significantly reduce the background noise, and do not introduce new features that are not present in the high-resolution slice. The quantitative results paint a similar picture, and are displayed in Table 2.5. The difference between the three proposed methods in terms of MSE and SSIM is small, whereas the cubic up-sampling performs worse on both metrics.

Comparing method A and method B, we observe a difference between simulation and experimental data. For the pixel-limited dataset, method A performs better on the MSE and SSIM metrics, and it outputs visually more appealing and accurate results. For the real-world data, however, there appears to be qualitatively little difference between method B and method A. In this case, the coarse voxel grid seems fine enough to represent almost all details that can be reconstructed.

2.3.3 Comparison with Other Network Structures

In this section, we compare the results of the MS-D network to the widely used U-net approach described in [156] on both the simulated and experimental data. The data acquisition and reconstruction procedures remain unchanged, and the U-net is trained on the low- and high-resolution data for two days or 1000 epochs, whichever comes first. As before, the metrics on the simulated data are computed based on the original phantom and the output of the network, and the metrics on the experimental data are computed based on the high-resolution target and the output of the network.

Neural network implementation The U-net network is implemented in PyTorch, and is based on a widely available open source implementation. We have provided our code as an open source package [68]. This implementation of the U-net architecture is almost identical to that described in [156]: the images are down-sampled four times using 2×2 max-pooling, the "up-convolutions" have trainable parameters, and the convolutions have 3×3 kernels. Like [38], this implementation uses batch normalization before each ReLU. Moreover, the smallest image layers are 512 channels instead of 1024 channels, and zero-padding is used instead of reflection-padding. An adaptation was made to allow the network to process images whose width or height was not divisible by 16: these images were padded to the right dimensions using reflection padding. All U-net networks are trained from scratch in each experiment.

Metrics and evaluation The quantitative results are given in Table 2.6, and the network outputs are visually compared in Figure 2.10. On the whole, the output of the U-net is visually similar to the MS-D network, and apart from a

		Method A		**Method A**		**Method B**	
		Slab: 9 Slices		**Slab: 1 Slice**		**Slab: 1 Slice**	
		MSE	SSIM	MSE	SSIM	MSE	SSIM
Pixel-limited	U-net	0.0057	0.4225	0.0092	0.9149	0.0111	0.6850
	MS-D	**0.0044**	**0.9578**	0.0089	0.9281	0.0111	0.8065
Optics-limited	U-net	0.0159	0.7436	0.0402	0.1339	0.0178	0.7959
	MS-D	**0.0143**	**0.8275**	0.0154	0.8220	0.0178	0.8135
Oatmeal	U-net	0.0030	0.4372	0.0029	0.4372	0.0037	0.4221
	MS-D	0.0027	0.4508	0.0026	**0.4529**	**0.0025**	0.4521

Table 2.6: Comparison of the MSE and SSIM between MS-D and U-net for all three datasets. For each dataset and metric, the best results are shown in bold.

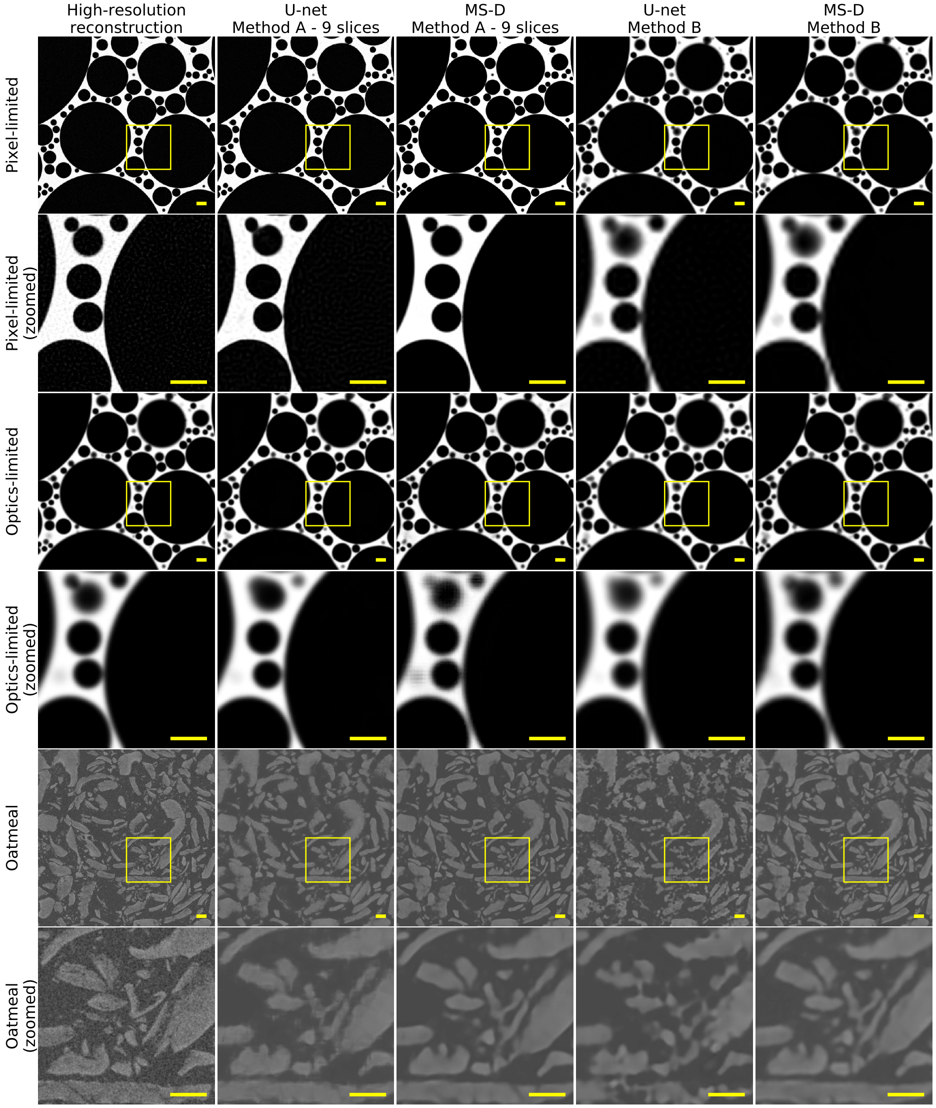


Figure 2.10: Comparison of U-net and MS-D on the central slice of the upper region of interest. The high-resolution slice (left-most column) is the result of reconstruction (Equation (2.12)). The four columns on the right are computed from a low-resolution reconstruction using a U-net or MS-D network. The first and third row display the results on simulated data, and the fifth row displays the results on experimentally acquired oatmeal data. Magnifications of the central yellow square are displayed in the even rows. The output of method B is cubically up-sampled by a factor of 4 to be the same size as the other images.

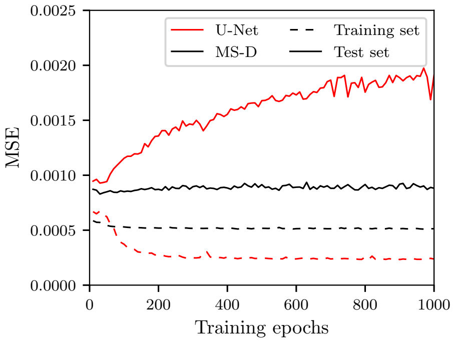


Figure 2.11: Oatmeal dataset, method B: the mean square error for MS-D and U-net calculated on the central slice of the training and test set. The training error decreases as the networks are trained longer. The test error of the U-net increases, whereas the test error of the MS-D network remains stable.

few negative outliers, the metrics of U-net are similar to those of the MS-D network. The lower SSIM metrics of the U-net could be explained by the fact that, on the pixel-limited dataset, the U-net appears to introduce the same non-smooth texture in the foam (method A—9 slices) and in the voids (method B) that is present in the high-resolution image. Likewise, the oatmeal flakes in the U-net outputs appear to have a distinct texture, especially for method B. It appears that some care must be taken to prevent the U-net from overfitting to the training data. In Figure 2.11, we compare the training and test error of the U-net and MS-D networks as training progresses. The mean square error metric is computed on the central slice of the training and test set of the oatmeal dataset. For both networks, the MSE decreases on the training slice. On the test slice, however, the MSE metric of the MS-D network remains consistently low, but the MSE of the U-net increases as training progresses, indicating that the network overfits to the training data.

2.4 Discussion

The experimental results demonstrate the feasibility of applying our combined acquisition scheme and on-the-fly machine learning technique to improve the resolution of tomographic reconstructions. Although our approach already achieves substantial improvement in image resolution, we believe our approach can still be improved. In this section, we discuss possible improvements to the network training procedure, and possible adaptations to less self-similar objects.

We believe that training the neural network can be accomplished faster and that the training procedure can be made less sensitive to small changes in alignment. In its current form, the computations of the learning step take considerably longer than the reconstruction step. In fact, the reconstruction for both the simulated data and the experimental data took less than fifteen minutes, while the learning step took up to two days. By using a validation set, for instance, the training can be cut short when the validation error does not improve. Sacrificing part of the training set for validation appears possible without making the training set too small for the network to learn a useful transformation from low- to high-resolution data. Based on the results in Figure 2.11, we expect that training for a considerably shorter period is possible, and will lead to results comparable to those reported here. The neural networks were trained using the mean square error, which is sensitive to small changes in alignment. Hence, the alignment of the low-resolution and high-resolution reconstructions is critical, but can in practice not always be guaranteed. Therefore, using an error metric that does not depend on the exact alignment of the input and target data could improve robustness. Resolving this problem has received considerable attention, for instance by learning the loss function in an adversarial setting [78, 106].

The experimental results indicate that super-resolution can be obtained using our method if objects are self-similar, thus providing a super-resolution approach that does not rely on a training set of similar objects. There are cases, however, where not all parts of the object have similar structure. For example, an object may consist of multiple parts with different characteristics. In that case, the network may be trained on multiple ROI reconstructions in these parts to ensure that it is able to improve the resolution of the various structures within the object. The areas where the local structure are different can usually be identified using just the low-resolution reconstruction.

2.5 Conclusions

In this chapter, we have presented a novel technique for improving the resolution of tomographic volumes using a custom scanning procedure combined with on-the-fly machine learning. The technique relies on combining high-resolution projection data of a small region of interest with low-resolution projection data of the entire object. Reconstructions of both projection sets are used to train a neural network, which is then able to improve the resolution of the low-resolution reconstruction of the entire object. The effectiveness of our approach was tested on simulated data and real-world experimental data. The proposed approach is able to recover a large fraction of high-resolution features without introducing additional artifacts. Moreover, it requires a limited increase in scanning time and radiation dose. We have proposed two variants of our approach (A and B). The first variant (A) performs better on synthetic data but is more computationally costly to compute. The second variant (B) is considerably cheaper to compute, results in smaller output images, and produces results qualitatively similar to method A on real-world experimental data. Our proposed approach has the added advantage of removing noise from reconstructions of experimental data. The results show that the proposed machine learning method is able to significantly improve resolution of tomographic reconstructions without requiring a training set of similar objects.

3

Noise2Inverse: Self-supervised denoising

"You only start to see it once you understand."

"Je gaat het pas zien als je het doorhebt."

Johan Cruijff,
Vrij Nederland, 8 Jan 1994

Reconstruction algorithms compute an image from indirect measurements. For a subclass of these algorithms, the relation between the reconstructed image and the measured data can be described by a linear operator. Such *linear reconstruction methods* are used in a variety of applications, including X-ray and photo-acoustic tomography, ultrasound imaging, deconvolution microscopy, and X-ray holography [10, 102, 118, 121, 123, 148, 167, 197, 198]. These methods are well-suited for fast, parallel computation [140], but are also generally sensitive to measurement noise, leading to errors in the reconstructed image [31, 167]. Controlling this error, i.e., *denoising*, is a central problem in inverse problems in imaging [16, 27, 34, 85, 123, 140, 174].

Supervised deep convolutional neural network (CNN)-based methods are able to accurately denoise reconstructed images in several inverse problems [16, 85, 123, 140, 174]. These networks are trained in a *supervised* setting, which amounts to finding the network parameters that best compute a mapping from noisy to clean

This chapter is based on:
A. A. Hendriksen, D. M. Pelt, and K. J. Batenburg. "Noise2inverse: Self-Supervised Deep Convolutional Denoising for Tomography". *IEEE Transactions on Computational Imaging* (2020), pp. 1–1.

reconstructed images on a dataset of example image pairs. However, the success of these supervised deep learning methods critically depends on the availability of such a high-quality training dataset of similar images [16, 112].

For photographic image denoising, recent work has shown that deep learning may be possible without obtaining high quality target images, by instead training on paired noisy images [107]. Nonetheless, such *Noise2Noise* training still requires additional noisy data. The feasibility of image denoising by *self-supervised* training, that is, training with *single* instead of paired noisy images, was demonstrated by [14, 100, 104]. These self-supervised training methods, such as Noise2Self, depend on the assumption that noise in one pixel is statistically independent from noise in another pixel.

In inverse problems, reconstructed images may exhibit coupling of the measured noise [85]. In CT, for instance, backprojection smears out the noise in a detector pixel across a line through the reconstructed image. Naturally, this causes the noise in one pixel to be statistically dependent on noise in other pixels of the reconstructed image.

In this chapter, we demonstrate that a straightforward application of Noise2Self to reconstructed CT images delivers substantially inferior results compared to results obtained on photographic images, for which it was developed. We analyze the cause of this apparent mismatch, and propose Noise2Inverse, a new approach that is specifically designed for linear reconstruction methods in imaging to overcome these limitations.

In the proposed Noise2Inverse approach, the training regime explicitly takes into account the structure of the noise in the inverse problem. In its simplest form, our method splits the measured data in two parts, from which two reconstructions are computed. We train a CNN to transform one reconstruction into the other, and vice versa. The properties of the physical forward model cause the noise in the reconstructed images to be statistically independent. This enables the CNN to perform *blind* image denoising on the reconstructed images. That is, our method does not assume a *known noise model.* We stress that our method can be applied to existing datasets without acquiring additional data.

In recent years, a range of deep learning approaches have been developed for denoising in imaging with limited training data. Several weight-regularized self-supervised methods exist that require a known Gaussian noise model [32, 126, 170, 200]. While such a model is often available in direct imaging modalities, the noise model for reconstructed images in an inverse problem setting is often more complex and hard to characterize by such a Gaussian model. Unsupervised approaches using the Deep Image Prior [37, 120, 177] have been proposed for image restoration and inverse problems [46, 80]. A key obstacle for the application of such techniques to large-scale 3D image reconstruction problems is their computational cost, as they involve training a new network for every 2D slice of the reconstruction. For inverse problems, approaches that rely on splitting the measurement data have recently been proposed for magnetic resonance imaging (MRI) [112, 196] and Cryo-transmission electron microscopy (Cryo-EM) [27] showing image quality improvement with respect to denoising applied on the reconstructed image. While these results are highly promising, a solid theoretical underpinning that allows analysis and insights into the interplay between the underlying noise model of the inverse problem and the obtained solution is currently lacking.

In this chapter — motivated by these promising results — we present a framework for generalizing the self-supervised denoising approach in the setting of linear reconstruction methods. Our framework pinpoints exactly the underlying theoretical properties that explain the differences in observed results of self-supervised approaches. We perform a qualitative and quantitative comparison to conventional iterative reconstruction and state-of-the-art image denoising techniques. We evaluate these methods on several simulated low-dose CT datasets, and include results on an existing experimentally acquired CT dataset, for which no low-noise data is available. In addition, we present a systematic analysis of the hyper-parameters of the proposed method.

This chapter is structured as follows. In Section 3.1, we introduce linear inverse problems and deep learning for image denoising, including self-supervised methods. In Section 3.2, we introduce the proposed Noise2Inverse method, and show its theoretical properties, which we use to develop an implementation for computed tomography. In Section 3.3, we perform experiments to compare the performance of Noise2Inverse, conventional reconstruction techniques, and Noise2Self-based methods on real and simulated CT datasets. In addition, we perform a hyper-parameter study of the proposed method. We discuss these results in Section 3.4.

3.1 Notation and concepts

As prerequisites for describing our Noise2Inverse approach, we first discuss deep learning methods for image denoising, including strategies for training neural networks when clean images are unavailable. In addition, we review linear inverse problems, where we discuss that denoising reconstructed images introduces additional difficulties.

3.1.1 Deep learning for image denoising

The goal of image denoising is to recover a 2D image $\mathbf{y} \in \mathcal{Y} = \mathbb{R}^m$ from a measurement $\tilde{\mathbf{y}} \in \mathcal{Y}$ that is corrupted by random noise ϵ, taking values in $\mathcal{Y}$. This problem is described by the equation

$$\tilde{\mathbf{y}} = \mathbf{y} + \epsilon. \tag{3.1}$$

It is common to assume that the entries of the noise vector ϵ are mutually independent. Many image denoising methods rely on this assumption [43, 100, 199]. In addition, these methods assume that the image exhibits some statistically meaningful structure that can be exploited to remove the noise. The popular BM3D algorithm [43], for example, exploits non-local self-similarity, i.e., the expectation that certain structures of the image are repeated elsewhere in the image. Note that

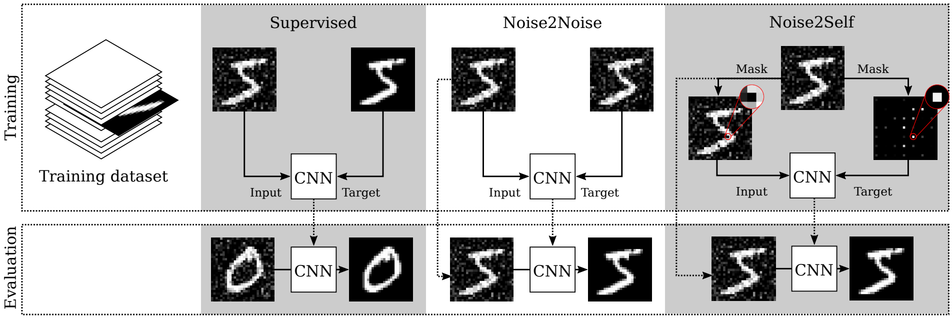


Figure 3.1: Three training regimes for CNN-based image denoising. Supervised training is performed with noisy and clean images, and the trained CNN is applied to unseen noisy data. Noise2Noise training is performed with pairs of noisy images. Noise2Self training is performed with just noisy images, which are split into input-target pairs. The loss is only computed where target pixels are non-zero. The red inset displays one of these locations. For Noise2Noise and Noise2Self, the trained CNN can be applied to the training data to obtain clean images.

it is also possible to include BM3D as a prior inside iterative algorithms for inverse problems using a plug-and-play framework [182].

Instead of relying on an explicit image prior, prior knowledge can be based on a range of example images, as is done in deep learning. In particular, deep convolutional neural networks (CNNs) have been recognized as a powerful and versatile denoising technique [199]. We briefly introduce three training schemes for denoising with CNNs: supervised[199], Noise2Noise [107], and Noise2Self [14].

The **supervised** training scheme has access to a *training dataset* containing pairs of noisy *input* and clean *target* images

$$(\tilde{\mathsf{y}}_i, \mathsf{y}_i) \sim (\mathsf{y} + \epsilon, \mathsf{y}), \quad i = 1, \ldots, N, \tag{3.2}$$

where y is a random variable taking values in $\mathcal{Y}$ that represents the clean images. The supervised training objective is to find the *regression* function

$$h^* = \arg\min_h \mathbb{E}_{\mathsf{y},\epsilon} \left[\| h(\mathsf{y} + \epsilon) - \mathsf{y} \|_2^2 \right], \tag{3.3}$$

that minimizes the *expected prediction error* [66]. The most common loss function is the pixel-wise mean square error, which we use here. Alternative training losses are also used, such as the L1 loss and perceptual losses [107]. Solving Equation (3.3) is usually intractable. Therefore, the expectation is estimated by the sample mean over the training dataset, which is minimized over neural networks $f_\varphi : \mathcal{Y} \to \mathcal{Y}$ with parameters φ. The training task is then to find the optimal parameters

$$\hat{\varphi} = \arg\min_\varphi \sum_{i=1}^{N} \| f_\varphi(\tilde{\mathsf{y}}_i) - \mathsf{y}_i \|_2^2, \tag{3.4}$$

which minimize the loss on the sampled image pairs. The trained network $f_{\hat{\varphi}}$ is applied to unseen noisy images to obtain denoised images, as displayed in Figure 3.1.

The regression function that minimizes the expected prediction error in Equation (3.3) is the conditional expectation

$$h^*(\tilde{y}) = \mathbb{E}\left[\mathsf{y} \mid \mathsf{y} + \epsilon = \tilde{y}\right]. \tag{3.5}$$

In practice, the trained neural network $f_{\hat{\varphi}}$ does not equal h^* and an approximation is obtained.

Noise2Noise training may be applied if no clean images are available, but one can measure *independent* instances of the noise for each image. The training dataset contains pairs of independent noisy images

$$(\mathrm{y}_i + \epsilon_i, \mathrm{y}_i + \delta_i) \sim (\mathsf{y} + \epsilon, \mathsf{y} + \delta), \quad i = 1, \dots, N, \tag{3.6}$$

where the noise δ is a random variable that is statistically independent of ϵ. The training task is to determine

$$\hat{\varphi} = \arg\min_{\varphi} \sum_{i=1}^{N} \| f_{\varphi}(\mathrm{y}_i + \epsilon_i) - (\mathrm{y}_i + \delta_i) \|_2^2, \tag{3.7}$$

and the trained neural network $f_{\hat{\varphi}}$ approximates

$$h^* = \arg\min_{h} \mathbb{E}_{\mathsf{y},\epsilon,\delta}\left[\| h(\mathsf{y} + \epsilon) - (\mathsf{y} + \delta) \|_2^2\right]. \tag{3.8}$$

If the noise δ is mean-zero, i.e., $\mathbb{E}\left[\delta\right] = 0$, the expected prediction error in Equation (3.8) is minimized by the same regression function h^* as in the supervised regime (Equation (3.5)). In practice, Noise2Noise and supervised training indeed yield trained networks with similar denoising performance.

Noise2Self enables training a neural network denoiser without any additional images. The training dataset contains only noisy images

$$\tilde{\mathrm{y}}_i \sim \mathsf{y} + \epsilon, \quad i = 1, \dots, N. \tag{3.9}$$

The method depends on the assumption that the noise is element-wise statistically independent and mean-zero, and that the clean images exhibit some spatial correlation.

Noise2Self training uses a masking scheme that ensures that the loss compares two statistically independent images. For simplicity, we describe a simplified version of Noise2Self training, and refer to [14] for a more in-depth explanation. In each training step, the noisy image is split into two *sub-images*: one sub-image — the target — contains non-adjacent pixels and the other sub-image — the input — contains the remaining surrounding pixels. The network is trained to predict the value of a noisy pixel from its surrounding noisy pixels, as is shown in Figure 3.1.

The division of pixels between the input and target image is determined by a partition $\mathcal{J}$ of the pixels such that adjacent pixels are in different subsets. We denote by $J \in \mathcal{J}$ the *target section*, and by J^C the *input section*, where J^C denotes the set complement of J, containing all pixel locations not contained in J. The input and target images $\mathbb{1}_{J^C}\tilde{\mathsf{y}}_i$ and $\mathbb{1}_J\tilde{\mathsf{y}}_i$ have non-zero pixels only in the *input* and *target section*, respectively. Here, $\mathbb{1}_J$ denotes the indicator function such that element-wise multiplication of $\mathbb{1}_J$ with an image retains pixel values in J and sets pixels to zero elsewhere. The training task is to determine the set of network parameters minimizing the training loss

$$\hat{\varphi} = \arg\min_{\varphi} \sum_{i=1}^{N} \sum_{J \in \mathcal{J}} \|\mathbb{1}_J f_\varphi(\mathbb{1}_{J^C}\tilde{\mathsf{y}}_i) - \mathbb{1}_J\tilde{\mathsf{y}}_i\|_2^2, \tag{3.10}$$

where the loss is only computed on the target sections.

The inference step is performed by the *section-wise combined network* $g_{\hat{\varphi}} : \mathcal{Y} \to \mathcal{Y}$,

$$g_{\hat{\varphi}}(\tilde{y}) := \sum_{J \in \mathcal{J}} \mathbb{1}_J f_{\hat{\varphi}}(\mathbb{1}_{J^C}\tilde{y}), \tag{3.11}$$

that computes the output in each target section by applying the trained network to the input section.

The piecewise-combined network is an approximation of the regression function

$$g^*(\tilde{y}) = \sum_{J \in \mathcal{J}} \mathbb{1}_J \mathbb{E}[\mathbb{1}_J \mathsf{y} \mid \mathbb{1}_{J^C}\,(\mathsf{y} + \epsilon) = \mathbb{1}_{J^C}\tilde{y}]. \tag{3.12}$$

This regression function computes the conditional expectation of the clean image in each target section using the surrounding noisy pixels.

Although aforementioned methods can produce accurately denoised photographic images in many cases [14, 107, 199], a subclass of these algorithms — Noise2Self in particular — has strong requirements on the element-wise independence of the noise. These requirements do not generally hold for solutions of linear inverse problems, as we discuss next.

3.1.2 Linear inverse problems

We are concerned with inverse problems that are described by the equation

$$\mathbf{A}\mathbf{x} = \mathbf{y}, \tag{3.13}$$

where $\mathbf{x} \in \mathcal{X} = \mathbb{R}^n$ denotes an unknown image that we wish to recover, and $\mathbf{y} \in \mathcal{Y} = \mathbb{R}^m$ denotes the indirect measurement. The linear forward operator $\mathbf{A} : \mathbb{R}^n \to \mathbb{R}^m$ describes the physical model by which the measurement arises from the image $\mathbf{x}$. As in the image denoising setting, these measurements are corrupted by element-wise independent noise ϵ, and we write

$$\tilde{\mathbf{y}} = \mathbf{A}\mathbf{x} + \epsilon. \tag{3.14}$$

Although noise in Equation (3.14) is modeled as an additive term, we note that this model also covers non-additive noise, such as Poisson noise, where the noise term typically depends on the signal intensity.

Reconstruction algorithms approximate the image $\mathbf{x}$ from measured data $\mathbf{y}$. A subclass of these reconstruction algorithms computes a linear operator $\mathbf{R} : \mathcal{Y} \to \mathcal{X}$. Examples of linear reconstruction algorithms include the filtered backprojection algorithm for tomography and Wiener filtering for deconvolution microscopy [31, 167]. We denote the reconstruction from a noisy measurement by

$$\tilde{\mathbf{x}} = \mathbf{R}\tilde{\mathbf{y}} = \mathbf{R}\mathbf{y} + \mathbf{R}\epsilon, \tag{3.15}$$

which can contain artifacts unrelated to the measurement noise, e.g., reconstruction and/or under-sampling artifacts. The reconstruction operator $\mathbf{R}$ may cause elements of the reconstructed noise $\mathbf{R}\epsilon$ to be statistically coupled, even if ϵ is element-wise independent [85]. That $\mathbf{R}\epsilon$ does not satisfy the element-wise independence property is unavoidable for all but the most trivial cases, since inverse problems are essentially defined by the intricate coupling of the unknown image with its indirect measurement.

This coupling of the noise seriously degrades the effectiveness of the Noise2Self approach, as we will see in Section 3.3.4. In the next section, we propose a self-supervised method that does take into account the properties of noise in inverse problems.

3.2 Noise2Inverse

In this section, we present the proposed Noise2Inverse method. First, we describe the assumed noise model, and give a general description of the method. In Section 3.2.1, we provide a theoretical explanation how and why the convolutional neural network learns to denoise. Here, we also discuss how these results can guide implementation in practice. In Section 3.2.2, we give a more practical description of the implementation for tomography, and discuss implementation choices with regard to the obtained theoretical results.

Suppose that we wish to examine several unknown images $\mathsf{x}_1, \ldots, \mathsf{x}_N \sim \mathsf{x}$, sampled from some random variable x. We obtain noisy indirect measurements

$$\tilde{\mathsf{y}}_i \sim \mathbf{A}\mathsf{x}_i + \epsilon, \quad i = 1, \ldots, N, \tag{3.16}$$

where we assume that the noise ϵ is element-wise independent and mean-zero conditional on the data, i.e.,

$$\mathbb{E}_{\mathsf{x},\epsilon}\left[\mathbf{A}\mathsf{x} + \epsilon \mid \mathbf{A}\mathsf{x} = \mathsf{y}\right] = \mathsf{y}. \tag{3.17}$$

As in Equation (3.14), we assume the noisy may be non-additive. Our goal is to recover the *clean reconstructions* that would have been obtained in the absence of noise, i.e., $\mathsf{x}_i^* = \mathbf{R}\mathsf{y}_i$ with $\mathsf{y}_i = \mathbf{A}\mathsf{x}_i, i = 1, \ldots, N$.

One approach is to compute noisy reconstructions, and use Noise2Self to remove the noise in the reconstructed images. Given the noisy reconstructions

$\tilde{\mathrm{x}}_i = \mathbf{R}\tilde{\mathrm{y}}_i, i = 1, \ldots, N$, the training task is to determine the network parameters minimizing the training loss

$$\hat{\varphi} = \arg\min_{\varphi} \sum_{J \in \mathcal{J}_x} \sum_{i=1}^{N} \| \mathbb{1}_J f_\varphi(\mathbb{1}_{J^C} \tilde{\mathrm{x}}_i) - \mathbb{1}_J \tilde{\mathrm{x}}_i \|_2^2, \tag{3.18}$$

where the target sections are contained in $\mathcal{J}_x$, a partition of the pixels of the reconstructed images. As discussed before, however, the noise in the input and target pixels of the reconstructed images are unlikely to be statistically independent.

The key idea of the proposed Noise2Inverse method is that it partitions the data in the measurement domain — where the noise is element-wise independent — but trains the CNN in the reconstruction domain. In each training step, the measured data is partitioned into an input and target component, and a neural network is trained to predict the reconstruction of one from the reconstruction of the other. After training, the neural network is applied to denoise the reconstructions.

The division of measured data between input and target is determined by the collection $\mathcal{J}$ of *target sections* $J \subset \{1, 2, \ldots, m\}$ that represent subsets of the measurement domain $\mathcal{Y} = \mathbb{R}^m$. We note that $\mathcal{J}$ can be chosen such that it contains structured subsets of the measurement domain, rather than *all* subsets. For each target section $J \in \mathcal{J}$, the measurement is split into input and target sub-measurements $\tilde{\mathrm{y}}_{i,J^C}$ and $\tilde{\mathrm{y}}_{i,J}$, where J^C denotes the set complement of J with respect to $\{1, 2, \ldots, m\}$. The input and target sub-reconstructions are computed by linear reconstruction operators $\mathbf{R}_J : \mathcal{Y}_J \to \mathcal{X}$ that take into account only the measurements in section $J \in \mathcal{J}$. We define

$$\tilde{\mathrm{x}}_{i,J^C} = \mathbf{R}_{J^C} \tilde{\mathrm{y}}_{i,J^C} \text{ and } \tilde{\mathrm{x}}_{i,J} = \mathbf{R}_J \tilde{\mathrm{y}}_{i,J}$$

to be the input and target sub-reconstructions of $\tilde{\mathrm{y}}_i$, respectively.

The training task is to determine the parameters

$$\hat{\varphi} = \arg\min_{\varphi} \frac{1}{|\mathcal{J}|} \sum_{J \in \mathcal{J}} \sum_{i=1}^{N} \| f_\varphi(\tilde{\mathrm{x}}_{i,J^C}) - \tilde{\mathrm{x}}_{i,J} \|_2^2, \tag{3.19}$$

that best enable the network $f_{\hat{\varphi}}$ to predict the target sub-reconstruction from the complementary input sub-reconstruction.

The final output is computed by the *section-wise averaged network*, which applies the trained network to each input sub-reconstruction, and computes the average, yielding

$$\mathrm{x}^*_{i,\mathrm{out}} = \frac{1}{|\mathcal{J}|} \sum_{J \in \mathcal{J}} f_{\hat{\varphi}} \left(\tilde{\mathrm{x}}_{i,J^C} \right). \tag{3.20}$$

In the next section, we show why the final result approximates the clean reconstruction.

3.2.1 Theoretical framework

In this section, we embed Noise2Inverse in a theoretical framework that explains why it is an accurate denoising method. In addition, we describe design considerations that enable it to operate successfully.

Below, we show that Noise2Inverse recovers an average clean reconstruction in theory. This result is founded upon Proposition 1, which shows that the expected prediction error is the sum of the variance of the reconstructed noise and the *supervised prediction error*, which is the expected prediction error that would have obtained if the target reconstructions were noise-free. Hence, the regression function that minimizes the expected prediction error also minimizes the loss with respect to the unknown clean reconstruction. Therefore, it predicts a clean sub-reconstruction when given a noisy sub-reconstruction.

As before, we represent the clean and noisy measurements by the random variables $\mathsf{y} = \mathbf{A}\mathsf{x}$ and $\tilde{\mathsf{y}} = \mathsf{y} + \epsilon$. The input and target sub-reconstructions are represented by random variables $\tilde{\mathsf{x}}_{J^C} = \mathbf{R}_{J^C}\tilde{\mathsf{y}}_{J^C}$ and $\tilde{\mathsf{x}}_J = \mathbf{R}_J\tilde{\mathsf{y}}_J$ for $J \in \mathcal{J}$. In this case, the trained network $f_{\hat{\varphi}}$ obtained in Equation (3.19) approximates the regression function

$$h^* = \arg\min_h \frac{1}{|\mathcal{J}|} \sum_{J\in\mathcal{J}} \mathbb{E}_{\mathsf{x},\epsilon} \| h(\tilde{\mathsf{x}}_{J^C}) - \tilde{\mathsf{x}}_J \|^2, \tag{3.21}$$

which minimizes the expected prediction error. We randomize the section J as well, representing it by J taking values uniformly at random in $\mathcal{J}$. The input and target sub-reconstructions become random in J as well, which is denoted by $\tilde{\mathsf{x}}_{\mathsf{J}^C} = \mathbf{R}_{\mathsf{J}^C}\tilde{\mathsf{y}}_{\mathsf{J}^C}$ and $\tilde{\mathsf{x}}_{\mathsf{J}} = \mathbf{R}_{\mathsf{J}}\tilde{\mathsf{y}}_{\mathsf{J}}$. The expected prediction error then becomes

$$\frac{1}{|\mathcal{J}|} \sum_{J\in\mathcal{J}} \mathbb{E}_{\mathsf{x},\epsilon} \| h(\tilde{\mathsf{x}}_{J^C}) - \tilde{\mathsf{x}}_J \|^2 = \mathbb{E}_\mu \| h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \tilde{\mathsf{x}}_{\mathsf{J}} \|^2,$$

where we replace the average over $J \in \mathcal{J}$ by the expectation with respect to J. We denote with μ the joint measure of x, ϵ, and J. Define the sub-reconstruction of the clean measurement

$$\mathsf{x}^*{}_{\mathsf{J}} = \mathbf{R}_{\mathsf{J}}\mathsf{y}_{\mathsf{J}}, \tag{3.22}$$

which describes the clean target reconstruction. Now the expected prediction error can be decomposed into two parts.

Proposition 1 (*Expected prediction error decomposition*)**.** *Let* $\tilde{\mathsf{x}}_{\mathsf{J}}, \tilde{\mathsf{x}}_{\mathsf{J}^C}, \mathsf{x}^*{}_{\mathsf{J}}$*, and* μ *be as above. Let* ϵ *be element-wise independent and satisfy* (3.17)*. Let* $\mathbf{R}_J$ *be linear for all* $J \in \mathcal{J}$*. Then, for any measurable function* $h : \mathcal{X} \to \mathcal{X}$*, we have*

$$\mathbb{E}_\mu \| h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \tilde{\mathsf{x}}_{\mathsf{J}} \|_2^2 = \mathbb{E}_\mu \| h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_{\mathsf{J}} \|_2^2 + \mathbb{E}_\mu \| \mathsf{x}^*{}_{\mathsf{J}} - \tilde{\mathsf{x}}_{\mathsf{J}} \|_2^2. \tag{3.23}$$

Proof. First, expand the squared norm [158, Lemma 3.12]

$$\begin{aligned} \| h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \tilde{\mathsf{x}}_{\mathsf{J}} \|^2 =& \| h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_{\mathsf{J}} + \mathsf{x}^*{}_{\mathsf{J}} - \tilde{\mathsf{x}}_{\mathsf{J}} \|^2 \\ =& \| h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_{\mathsf{J}} \|^2 + \| \mathsf{x}^*{}_{\mathsf{J}} - \tilde{\mathsf{x}}_{\mathsf{J}} \|^2 \\ &+ 2\langle h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_{\mathsf{J}}, \mathsf{x}^*{}_{\mathsf{J}} - \tilde{\mathsf{x}}_{\mathsf{J}} \rangle. \end{aligned}$$

Let $x \in \mathcal{X}$, $y = \mathbf{A}x$, and $J \in \mathcal{J}$. Then, from Equation (3.17), we obtain

$$\begin{aligned}\mathbb{E}_\mu\left[\tilde{\mathsf{x}}_\mathsf{J} \mid x, J\right] &= \mathbb{E}_\mu\left[\mathbf{R}_J \tilde{\mathsf{y}}_J \mid x, J\right] \\ &= \mathbf{R}_J\, \mathbb{E}_{\mathsf{x},\epsilon}\left[y_J + \epsilon_J \mid x\right] \\ &= \mathbf{R}_J\, y_J \\ &= \mathbf{x}^*_J, \end{aligned} \tag{3.24}$$

where we use that $\mathbf{R}_J$ is linear.

The noisy random variables $\tilde{\mathsf{x}}_{\mathsf{J}^C}$ and $\tilde{\mathsf{x}}_\mathsf{J}$ are independent conditioned on x and J, since domains of $\mathbf{R}_J$ and $\mathbf{R}_{J^C}$ do not overlap, and the noise ϵ is element-wise statistically independent. This independence condition allows us to interchange the order of the expectation and inner product [47, Proposition 2.3], which yields, using Equation (3.24),

$$\begin{aligned}&\mathbb{E}\left[\langle h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_\mathsf{J}, \mathsf{x}^*{}_\mathsf{J} - \tilde{\mathsf{x}}_\mathsf{J}\rangle \mid x, J\right] \\ &\quad= \langle \mathbb{E}\left[h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_\mathsf{J} \mid x, J\right], \mathbb{E}\left[\mathsf{x}^*{}_\mathsf{J} - \tilde{\mathsf{x}}_\mathsf{J} \mid x, J\right]\rangle \\ &\quad= \langle \mathbb{E}\left[h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_\mathsf{J} \mid x, J\right], 0\rangle \\ &\quad= 0.\end{aligned}$$

Using the tower property of expectation, we obtain

$$\begin{aligned}&\mathbb{E}_\mu \|h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \tilde{\mathsf{x}}_\mathsf{J}\|^2 \\ &\quad= \mathbb{E}\left[\mathbb{E}\left[\|h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \tilde{\mathsf{x}}_\mathsf{J}\|^2 \mid \mathsf{x}, \mathsf{J}\right]\right] \\ &\quad= \mathbb{E}\left[\mathbb{E}\left[\|h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_\mathsf{J}\|^2 + \|\mathsf{x}^*{}_\mathsf{J} - \tilde{\mathsf{x}}_\mathsf{J}\|^2 \mid \mathsf{x}, \mathsf{J}\right]\right] \\ &\quad= \mathbb{E}_\mu \|h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_\mathsf{J}\|^2 + \mathbb{E}_\mu \|\mathsf{x}^*{}_\mathsf{J} - \tilde{\mathsf{x}}_\mathsf{J}\|^2.\end{aligned}$$

□

Similar proofs can be found in [4, 14]. Proposition 1 states that the expected prediction error can be decomposed into the supervised prediction error, which depends on the choice of h, and the variance of the reconstruction noise, which does not depend on h. Therefore, when minimizing (3.23), the function h minimizes the difference between its output and the unknown clean target sub-reconstruction $\mathsf{x}^*{}_\mathsf{J}$. Note that the minimization of h occurs with respect to $\mathsf{x}^*{}_\mathsf{J}$ instead of the fully sampled reconstruction $\mathbf{x}^*$. When the target sections have been chosen such that $\mathbb{E}_\mathsf{J}\left[\mathsf{x}^*{}_\mathsf{J}\right] = \mathbf{x}^*$ holds, however, the difference is minimized.

The supervised prediction error, $\mathbb{E}_\mu \|h(\tilde{\mathsf{x}}_{\mathsf{J}^C}) - \mathsf{x}^*{}_\mathsf{J}\|_2^2$, is minimized [4] by the regression function

$$h^*(\tilde{x}) = \mathbb{E}_\mu\left[\mathsf{x}^*{}_\mathsf{J} \mid \tilde{\mathsf{x}}_{\mathsf{J}^C} = \tilde{x}\right]. \tag{3.25}$$

The section-wise averaged network, defined in Equation (3.20), therefore approximates the section-wise average of the regression function, defined by

$$g^*(\tilde{y}) = \frac{1}{|\mathcal{J}|} \sum_{J \in \mathcal{J}} \mathbb{E}_\mu\left[\mathsf{x}^*{}_\mathsf{J} \mid \tilde{\mathsf{x}}_{\mathsf{J}^C} = \tilde{x}_{J^C}\right], \tag{3.26}$$

where we write $\tilde{x}_{JC} = \mathbf{R}_{JC}\tilde{y}_{JC}$ for $\tilde{y} \in \mathcal{Y}$ and $J \in \mathcal{J}$.

Using these results, we can explain why the section-wise average obtains a denoised output. A noisy sub-reconstruction can be explained by different values of the clean reconstruction x^*. The expectation $\mathbb{E}_\mu \left[\mathsf{x}^*{}_\mathsf{J} \mid \tilde{\mathsf{x}}_{\mathsf{J}C} = \tilde{x}_{JC}\right]$ is the mean of noiseless reconstructed images consistent with the observed noisy reconstruction $\tilde{x}_{JC}$. Equation (3.25) therefore predicts that our method produces denoised images. In fact, our method computes the mean over all clean sub-reconstructions indicated by $J \in \mathcal{J}$.

The obtained results may be used to guide implementation in practice. Equation (3.26) explains how to choose subsets $\mathcal{J}$. First of all, the mean of the clean sub-reconstructions ${}^1\!/\!{}_{|\mathcal{J}|} \sum_{J\in\mathcal{J}} \mathsf{x}^*{}_J$ must resemble the desired clean image. This can be achieved by choosing $\mathcal{J}$ to be a partition of $\{1, \ldots, m\}$, or, by choosing $\mathcal{J}$ such that each measured data point is contained in the same number of overlapping subsets $J \in \mathcal{J}$. Not doing so introduces a systematic bias into the reconstruction.

Second, the sub-reconstructions should be homogeneously informative throughout the image. If the sub-reconstructions are very different, or contain limited information about large parts of the image, then many dissimilar clean images are consistent with the observed noisy reconstruction, and the average over all these images will become blurred.

We note that x^* denotes the clean reconstruction, rather than the unknown image. This has two consequences. First, the theory predicts that artifacts that are unrelated to the measurement noise, e.g. under-sampling artifacts and reconstruction artifacts, will not be removed by the proposed network. Second, if the reconstruction method also performs denoising operations, for instance by blurring, then the result of our method might become blurred. The same effect might occur when a non-linear reconstruction method is used, for which Proposition 1 does not generally hold. In this case, the regression function averages the bias introduced by the non-linear reconstruction of the noise. In the next section, we use the considerations discussed above to devise an approach for computed tomography.

3.2.2 Noise2Inverse for computed tomography

In this section, we describe our implementation of Noise2Inverse for 3D parallel-beam tomography, and discuss how the implementation relates to the theoretical considerations discussed before.

The 3D parallel-beam tomography problem may be considered as a stack of 2D parallel-beam problems. In 2D parallel-beam tomography, a parallel X-ray beam penetrates an object, after which it is measured on a line detector. The line detector rotates around the object while capturing the intensity of the attenuated X-ray beam, as illustrated in the top panel of Figure 3.2.

In practice, a finite number of N_θ projections are acquired on a line grid of N_p detector elements at fixed angular intervals. Hence, the projection data can be described by a vector $\tilde{\mathbf{y}} \in \mathcal{Y} = \mathbb{R}^m, m = N_\theta \times N_p$, which is known as the *sinogram.* Likewise, the two-dimensional imaged object is represented by a vector $\mathbf{x} \in \mathcal{X} = \mathbb{R}^n, n = N_x^2$. We can formulate 2D parallel-beam tomography as a discrete

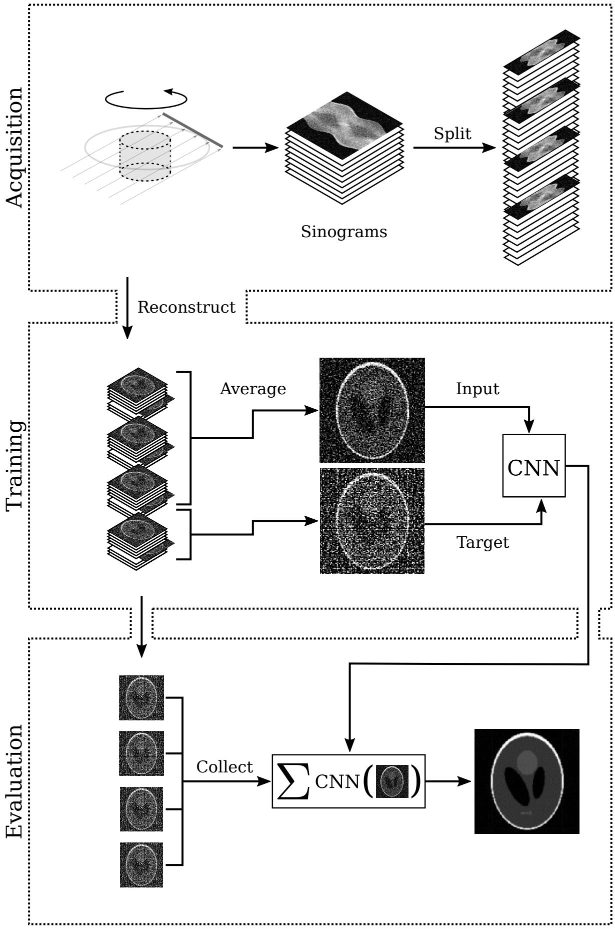


Figure 3.2: Noise2Inverse for computed tomography. First, 3D parallel-beam tomography obtains a stack of noisy sinograms by integrating over parallel lines at several angles. Next, the stack of sinograms is split along the angular axis. Then, the split sinograms are reconstructed to act as training dataset. During training, a dynamic subset of slices is averaged to form the input; the target is the average of the remaining slices. To obtain a low-noise result, the trained CNN is applied to all arrangements of input slices and averaged.

linear inverse problem, where $\mathbf{A} = (a_{ij})$ is an $m \times n$ matrix such that a_{ij} represents the contribution of object pixel j to detector pixel i. In 3D tomography, a sequence of 2D projection images of the 3D structure is acquired, which may be converted to a stack of 2D sinograms.

The imaged object can be recovered from the sinogram by a reconstruction algorithm, such as the filtered back-projection algorithm (FBP) [31]. FBP is an example of a linear operator that couples the measured noise in the reconstruction, as described in Equation (3.15). In addition, it is typically fast to compute, although its reconstructions tend to be noisy [34].

The Noise2Inverse method is well-suited to denoise this kind of problem. Suppose we have obtained a stack of 2D noisy sinograms $\tilde{\mathrm{y}}_1, \tilde{\mathrm{y}}_2, \ldots, \tilde{\mathrm{y}}_N$, acquired from a range of N_θ equally-spaced angles $\theta_1, \theta_2, \ldots, \theta_{N_\theta}$. Our approach follows the following steps.

First, we split each sinogram $\tilde{\mathrm{y}}_i$ into K sub-sinograms $\tilde{\mathrm{y}}_{i,1}, \ldots, \tilde{\mathrm{y}}_{i,K}$ such that each sub-sinogram $\tilde{\mathrm{y}}_{i,j}$ contains projection data from every Kth angle. The number of splits K is a hyper-parameter of the method.

Using the FBP algorithm, we compute sub-reconstructions

$$\tilde{\mathrm{x}}_{i,j} = \mathbf{R}_j(\tilde{\mathrm{y}}_{i,j}), \quad j = 1, \ldots, K. \tag{3.27}$$

For training, the division of the sub-reconstructions over the input and target is determined by a collection $\mathcal{J}$, which contains subsets $J \subset \{1, \ldots, K\}$. For $J \subset \{1, \ldots, K\}$, we define the mean sub-reconstruction as

$$\tilde{\mathrm{x}}_{i,J} = \frac{1}{|J|} \sum_{j \in J} \tilde{\mathrm{x}}_{i,j}. \tag{3.28}$$

As before, training of the neural network f_φ aims to find

$$\hat{\varphi} = \arg\min_{\varphi} \sum_{i=1}^{N} \sum_{J \in \mathcal{J}} \left\| f_\varphi\left(\tilde{\mathrm{x}}_{i,J^C}\right) - \tilde{\mathrm{x}}_{i,j} \right\|_2^2. \tag{3.29}$$

The final output, $\mathrm{x}^*_{i,\mathrm{out}}$, is computed slice by slice by section-wise averaging of the output of the trained network

$$\mathrm{x}^*_{i,\mathrm{out}} = \frac{1}{|\mathcal{J}|} \sum_{J \in \mathcal{J}} f_{\hat{\varphi}}\left(\tilde{\mathrm{x}}_{i,J^C}\right).$$

We identify two training strategies specifying $\mathcal{J}$:

X:1 Using this strategy, the input is the mean of $K - 1$ sub-reconstructions, and the target is the remaining sub-reconstruction, i.e.,

$$\mathcal{J}_{X:1} = \{\{1\}, \{2\}, \ldots, \{K\}\}. \tag{3.30}$$

1:X This is the reverse of the previous strategy: the input is a single sub-reconstruction, and the target is the mean of the remaining sub-reconstructions, i.e.,

$$\mathcal{J}_{1:X} = \{J^C \mid J \in \mathcal{J}_{X:1}\}. \tag{3.31}$$

In the 1:X strategy, the input is noisier than the target image, which corresponds to supervised training, where the quality of the target images is usually higher than the input images. The opposite is the case for the X:1 strategy, which corresponds more closely to Noise2Self denoising in its distribution of data between input and target, where more pixels are used to compute the input than to compute the target images. Note that other splits are possible, but we focus on these two strategies because they represent two extremes in the trade-off between input quality and target quality.

Our implementation of Noise2Inverse for tomography is consistent with the theoretical considerations discussed in the previous section. In both strategies, we prevent biasing the reconstructions, by ensuring that each projection angle occurs in reconstructions at the same rate. In fact, a property of FBP is that the full reconstruction is the mean of the sub-reconstructions. In theory, this means that training converges to the conditional expectation of the *full clean* FBP reconstruction. Furthermore, we use every Kth projection angle to compute the reconstructions. This ensures that the reconstructions are homogeneously informative throughout the image, and we prevent missing wedge artifacts, which occur when adjacent projection angles are used [127]. In addition, we use the FBP algorithm with the Ram-Lak filter[31], which does not blur the reconstructions to remove noise. Finally, we remark that our method is not geometry-specific, and can also be applied to non-parallel geometries, as is demonstrated in Section 3.3.3. In the next section, we describe the performance of this implementation in practice.

3.3 Results

We performed several experiments on tomographic reconstruction problems. These experiments were performed with the aim of assessing the performance of the proposed Noise2Inverse method, determining the suitability of Noise2Self denoising for tomographic images, and analyzing the impact of hyper-parameters on the performance of Noise2Inverse.

Comparison to denoising techniques Noise2Inverse is compared to tomographic reconstruction algorithms, an image denoising method, and an unsupervised deep learning method in Sections 3.3.1, 3.3.2, and 3.3.3. These sections describe a quantitative evaluation on simulated tomographic data, medical CT data with simulated noise, and a qualitative evaluation on an existing experimental dataset.

Noise2Self on tomographic images The experiments in Section 3.3.4 investigate a transfer of Noise2Self denoising to inverse problems. The Noise2Self method was evaluated on two datasets: one dataset with noise common to tomographic reconstructions and one with similar but element-wise independent noise. In addition, Noise2Inverse was compared to several variations of Noise2Self.

Hyper-parameters In Section 3.3.6, the impact on the reconstruction quality of several variables was investigated, specifically, the number of projection angles N_θ, the number of splits K, the training strategy $\mathcal{J}$, and the neural network architecture. In addition, we analyze the generalization performance of the Noise2Inverse

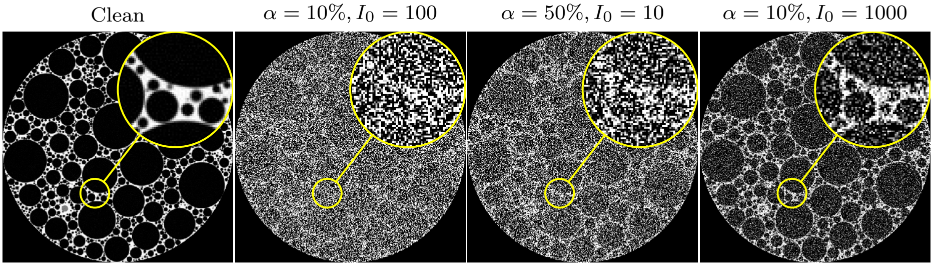

Clean
$\alpha = 10\%, I_0 = 100$
$\alpha = 50\%, I_0 = 10$
$\alpha = 10\%, I_0 = 1000$

Figure 3.3: Displays of the clean reconstruction (left) and low-dose reconstructions of the central slice of the foam phantom. Both α, the absorption of the phantom and I_0, the initial photon count per pixel, were varied. The yellow insets show an enlarged view of the reconstructions.

approach by training on progressively smaller subsets of the training dataset.

We first describe the simulated tomographic dataset and our implementation of Noise2Inverse. Both are used throughout the experiments.

Simulated data A cylindrical foam phantom was generated with 100,000 randomly-placed non-overlapping bubbles. Analytical projection images of the phantom were computed using the `foam_ct_phantom` package [140]. The value of each detector pixel was calculated by taking the average projection value of four equally-spaced rays through the pixel. Projection images were acquired from 1024 equally spaced angles.

The projection images of the foam dataset were corrupted with various levels of Poisson noise. The noise was varied by altering the average absorption of the sample α and the incident photon count per pixel I_0. The average absorption of the sample was calculated as the mean of the vector $1 - e^{-\mathbf{y}_i}$ for positions i where $\mathbf{y}_i$ was non-zero, and it was adjusted by modifying the intensity of the sinogram. The pixels in the noisy projections where sampled from $\tilde{\mathrm{p}}$, which for clean pixel value p was distributed as

$$I_0 e^{-\tilde{\mathrm{p}}} \sim \text{Poisson}\left(I_0 e^{-\mathrm{p}}\right).$$

i.e., a Poisson distribution on the pre-log raw data. As discussed in Section 1.1.3, this type of noise is typically mean-zero conditional on the clean projections, as described in Equation (3.17).

FBP reconstructions were computed on a 512^3 voxel grid with the Ram-Lak filter using the ASTRA toolbox [2]. On this grid, the radius of the random spheres ranged between 1.5 and 51 voxels. A reconstruction of the central slice of the foam phantom can be found in Figure 3.3, along with reconstructions of the noisy projection datasets.

Noise2Inverse We describe the Noise2Inverse implementation in terms of neural network architecture and training procedure.

The principal network architecture used throughout the experiments was the mixed-scale dense (MS-D) network [143], of which we used the `msd_pytorch` implementation [71]. The MS-D network has 100 single-channel intermediate layers, and the convolutions in layer i are dilated by $d_i = 1 + (i \mod 10)$. With 45,652 trainable network parameters, the MS-D architecture has considerably fewer parameters than comparable network architectures, reducing the risk of overfitting to the noise. The MS-D architecture is compared with other architectures in Section 3.3.6. The networks were trained for 100 epochs using the ADAM algorithm [94] with a mini-batch size of 12 and a learning rate of 10^{-3}.

3.3.1 Simulation study

In this section, Noise2Inverse is compared to two conventional iterative reconstruction techniques: the simultaneous iterative reconstruction technique (SIRT) [60] and Total-Variation Minimization (TV-MIN) [15]. In addition, we compare to the BM3D image denoising algorithm [43], the Deep Image Prior[177], and to supervised training. The reconstruction quality of these methods is assessed on a simulated foam phantom dataset with various noise profiles.

For Noise2Inverse, we used the X:1 training strategy with $K = 4$ splits. We show that this is a robust choice in Section 3.3.6.

Iterative reconstruction The hyper-parameters of SIRT and TV-MIN were tuned using the usually unavailable clean reconstructions. Therefore, the results of SIRT and TV-MIN might be better than what is achievable in practice, but they serve as a useful reference for comparison to Noise2Inverse. SIRT has no explicit hyper-parameters, but its iterative nature can be exploited for regularization: early stopping of the algorithm can attenuate high-frequency noise in the reconstructed image [60]. We selected the number of iterations (with a maximum of 1000) with the lowest Peak Signal to Noise Ratio (PSNR) on the central slice with respect to the clean reconstruction.

The FISTA algorithm [15] was used to calculate the TV-MIN reconstruction. TV-MIN has a regularization parameter λ that effectively penalizes steps in the gray value of the reconstructed image. As with SIRT, we selected the optimal number of iterations (with a maximum of 500) based on the PSNR of the central slice with respect to the clean reconstruction, and the value of the λ parameter maximizing the PSNR was determined using the Nelder-Mead method [185].

BM3D We used the BM3D implementation described in [116]. The BM3D algorithm was applied to the noisy FBP reconstructions and provided with the standard deviation of the noise, which was calculated from the difference image between the noisy and clean FBP reconstruction. The addition of a *prewhitening* step can improve denoising performance [163], but was not included as its computation becomes infeasible for large image sizes.

Supervised A separate training dataset was created to train MS-D networks with a supervised training approach. Here, the input and target images were noisy and clean reconstructions, respectively. The training parameters for supervised training — learning rate, batch size, network architecture — were exactly the same as for the Noise2Inverse network.

Deep Image Prior We used the Deep Image Prior implementation from [177]. The quality of the result can be improved by adding noise to the input and by

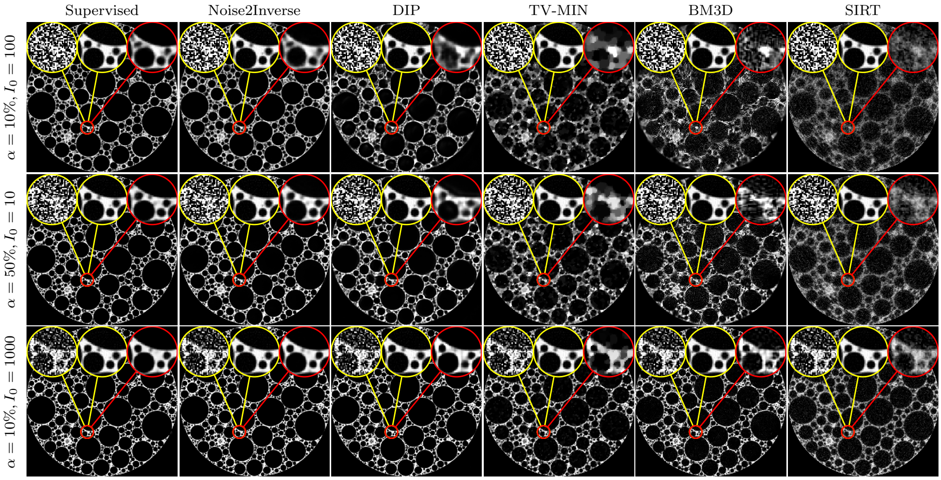

Supervised
Noise2Inverse
DIP
TV-MIN
BM3D
SIRT
$\alpha = 10\%, I_0 = 100$
$\alpha = 50\%, I_0 = 10$
$\alpha = 10\%, I_0 = 1000$

Figure 3.4: Results of supervised training, Noise2Inverse, Deep Image Prior (DIP), TV-MIN, BM3D, and SIRT on simulated foam phantoms with varying absorption α and photon count I_0. Results are shown on the central slice. The insets display the noisy and clean reconstructions (yellow) and the algorithm output (red).

			Full Volume		Central slice	
			PSNR	SSIM	PSNR	SSIM
α	I_0	Method				
10%	100	Supervised	20.01	0.83	20.02	0.80
		Noise2Inverse	**19.71**	**0.78**	**19.63**	**0.74**
		Deep Image Prior			17.98	0.59
		TV-MIN	16.89	0.46	16.78	0.40
		BM3D	14.79	0.38	14.81	0.33
		SIRT	15.56	0.36	15.54	0.32
50%	10	Supervised	21.77	0.86	21.71	0.83
		Noise2Inverse	**21.66**	**0.79**	**21.62**	**0.75**
		Deep Image Prior			19.75	0.67
		TV-MIN	18.08	0.53	17.99	0.48
		BM3D	16.65	0.49	16.74	0.45
		SIRT	16.53	0.42	16.50	0.37
10%	1000	Supervised	26.55	0.91	26.50	0.88
		Noise2Inverse	**26.25**	**0.89**	**26.24**	**0.87**
		Deep Image Prior			24.03	0.86
		TV-MIN	21.24	0.68	21.24	0.61
		BM3D	21.14	0.69	21.11	0.65
		SIRT	18.84	0.53	18.82	0.48

Table 3.1: On the full volume and on the central slice: comparison of PSNR and SSIM metrics for SIRT, TV-MIN, BM3D, Deep Image Prior, Noise2Inverse, and a supervised CNN at several noise profiles. Bold font is used to emphasize the best metrics, excluding supervised training, which serves as an oracle case for comparison.

employing an exponentially decaying average of recent iterations [37]. We used both techniques. To maximize the PSNR with respect to the ground truth, the training is stopped early with a maximum of 10000 iterations, and the σ parameter of the input noise is optimized using a line search.

Metrics and evaluation The output of each method was compared to the clean FBP reconstruction using two metrics: the structural similarity index (SSIM) [191] and the Peak Signal to Noise Ratio (PSNR). Because the reconstructed images did not fall in the $[0, 1]$ range, these metrics were computed with a data range that was determined by the minimum and maximum intensity of the clean reconstructed images. The metrics were calculated on the convex hull surrounding the object, which diminishes the importance of the background image quality. Due to the computational demands of deep image prior, we compute metrics on a single slice of the reconstruction rather than on the whole volume.

The top row of Figure 3.4 displays the output of Noise2Inverse for the central slice of the three simulated datasets. Denoising these datasets is challenging, as can be seen when comparing with SIRT and TV-MIN: these algorithms fail to recover several fine details. In contrast, our method achieves a much improved visual impression on all three datasets. As can be seen in Table 3.1, the PSNR and SSIM metrics of the Noise2Inverse method are considerably higher. The supervised network attains the best metrics, although by a slight margin compared to the Noise2Inverse method.

3.3.2 Medical CT

To assess the quality of reconstruction on medical data, we evaluate our method on simulated data from human abdomen CT scans from the low-dose CT Grand Challenge dataset [123]. This dataset contains full-dose reconstructions of 10 patients, consisting of a total of 2378 slices of 512×512 pixels. Following [5], sinograms were computed from these reconstructions by projecting onto a fan-beam geometry. Noise was applied, corresponding to a photon count of $10,000$ incident photons per pixel. Reconstructions are shown in Figure 3.5.

We compare the same methods as before. The dataset was split into a training

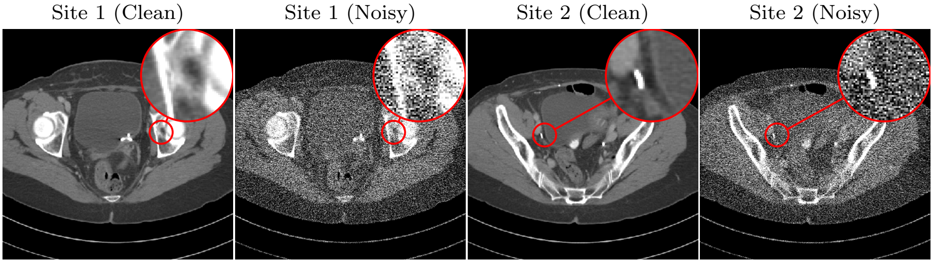


Figure 3.5: Original high-dose reconstructions of low-dose CT grand challenge (clean) and reconstructions with simulated noise (noisy).

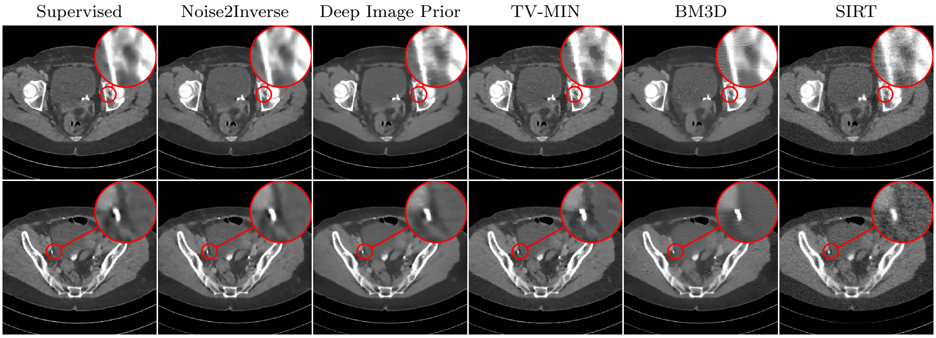


Figure 3.6: Results of supervised training, Noise2Inverse, Deep Image Prior, TV-MIN, BM3D, and SIRT on Low-dose CT grand challenge data with simulated noise. The red insets display the algorithm output.

	Full volume		Single slice	
	PSNR	SSIM	PSNR	SSIM
Method				
Supervised	46.34	0.99	46.29	0.99
Noise2Inverse	**45.06**	**0.99**	45.46	**0.99**
TV-MIN	44.91	**0.99**	**45.65**	0.98
Deep Image Prior			44.57	0.98
BM3D	43.84	**0.99**	43.97	0.98
SIRT	39.87	0.97	40.61	0.95

Table 3.2: Medical data: comparison of PSNR and SSIM metrics for SIRT, TV-MIN, BM3D, Deep Image Prior, a supervised CNN, and Noise2Inverse. Bold font is used to emphasize the best metrics, excluding supervised training, which serves as an oracle case for comparison.

dataset, consisting of nine patients, and a test set, containing the remaining patient. Both Noise2Inverse and the supervised CNN were trained on the training set. The optimal hyperparameters for SIRT, TV-MIN, and BM3D were determined on the training set. The Deep Image Prior, including its hyperparameters, was directly optimized with respect to the slices displayed in Figure 3.6. Metrics were calculated on the full volume of the test patient, and on the top displayed slice in Figure 3.6.

Results are shown in Figure 3.6 and Table 3.2. The Noise2Inverse method achieves similar results to TV-MIN, but without the staircasing artifacts. The difference between the methods is smaller in this experiment. For the SSIM metric, this is likely due to the low contrast of structures of interest compared to the full intensity range of the reconstructions. In general, compared to previous experiments, the noise has significantly lower intensity, and many different objects structures are present, each of which must be learned by the neural network.

3.3.3 Experimental data

The Noise2Inverse method was compared to SIRT and TV-MIN on an existing real-world experimental dataset from TomoBank [44]. The dataset, `Dorthe_F_002`, was acquired at the Advanced Photon Source at Argonne National Laboratory, and contained 900 noisy projection images of 960 × 600 pixels depicting a cylinder of glass beads that was scanned at experimental conditions designed to capture the dynamics of fast evolving samples. At 6 milliseconds per projection image, the exposure time was therefore much shorter than what is required for low-noise data acquisition [44]. The data was pre-processed with the TomoPy software package [64] and reconstructed with FBP[2], resulting in 900 2D slices of 960 × 960 pixels. We stress that no low-noise projection images were available.

For Noise2Inverse, an MS-D network was trained with the X:1 strategy and 4 splits for 100 epochs. The best parameter settings for SIRT and TV-MIN were determined by visual inspection. For SIRT, the best reconstruction was chosen from 1000 iterations on the central slice. For TV-MIN, the number of iterations was fixed at 500, and the optimal value of the regularization parameter was chosen from several values regularly spaced on an exponential grid. For BM3D, the best image was chosen from various values of the standard deviation parameter. We have omitted the Deep Image Prior since there was no ground truth with respect to which to perform early stopping.

After initial reconstructions, we found that the reported value of the center of rotation offset — 4.5 pixels from center — yielded unsatisfactory results. The reconstructions in Figure 3.7 were computed with a center of rotation that was shifted by 8.9 pixels. Results are shown for the central slice of the reconstructed volume. The FBP and SIRT reconstructions exhibit severe noise. The TV-MIN reconstruction improves on the level of noise, but contains stepping artifacts that reduce the effective resolution. Our method is able to remove the noise while retaining the finer structure of the image.

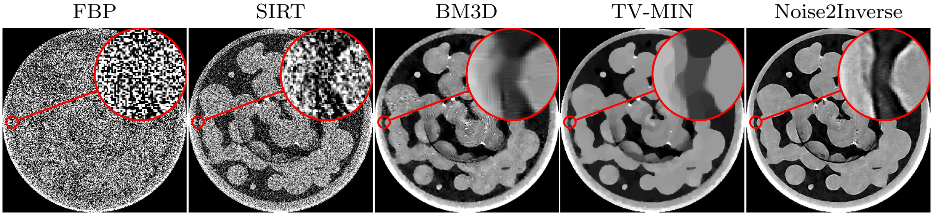


Figure 3.7: Reconstructions of cylinder containing glass beads [44] using: FBP, SIRT, BM3D, TV-MIN, and the proposed Noise2Inverse method. The red insets show an enlarged view of the algorithm output.

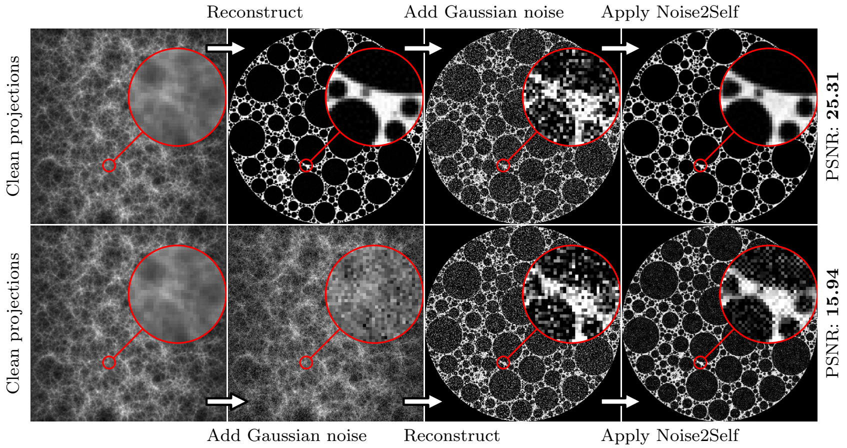


Figure 3.8: The effect of element-wise independence of the noise on the Noise2Self method. In the top row, Gaussian noise is added to a reconstruction, and Noise2Self is applied to remove it. In the bottom row, Gaussian noise is added to the projections before reconstruction, resulting in a reconstructed image with similar but coupled noise. Noise2Self achieves lower PSNR in the bottom row than in the top row.

3.3.4 Self-supervised image denoising for tomography

The performance of Noise2Self on tomographic images was evaluated in two experiments. The first experiment tested the element-wise independence requirement, by evaluating Noise2Self on images corrupted by element-wise independent noise and on images reconstructed from noisy projection data. The second experiment was a comparison of Noise2Inverse to Noise2Self, including variations of Noise2Self applied to projection and sinogram images.

Noise2Self We used the original implementation of Noise2Self [14], which obtains better performance than the simplified scheme discussed in Section 3.1.1. The training procedure was the same as for Noise2Inverse: an MS-D network was trained for 100 epochs as described at the beginning of Section 3.3.

Tomographic versus photographic noise Noise2Self was applied to images with coupled reconstructed noise and to similar but element-wise independent noise. In these experiments, the same foam phantom was used as before, and Gaussian noise was used throughout the comparison to strictly compare the independence properties of the noise. First, we confirmed that Noise2Self obtained denoised images when the noise satisfied the element-wise independence property. In this first case, a clean reconstruction was computed on a 512^3 voxel grid, and independent and identically distributed (i.i.d.) Gaussian noise was added to the reconstructed images. The PSNR of the noisy volume with respect to the clean reconstruction was 11.06. Then, Noise2Self was applied to obtain a denoised volume with significantly improved PSNR of 25.23. This process is displayed in the top row of Figure 3.8.

Next, the performance of Noise2Self on coupled reconstructed noise was inves-

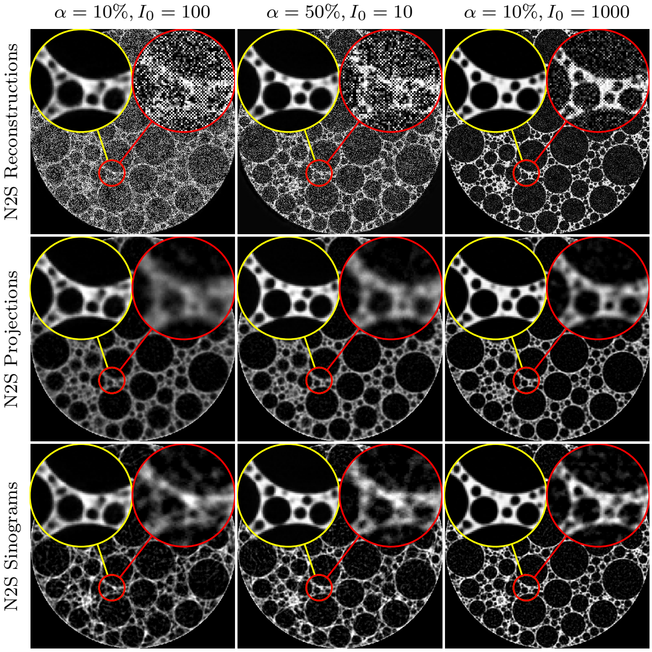

$\alpha = 10\%, I_0 = 100$
$\alpha = 50\%, I_0 = 10$
$\alpha = 10\%, I_0 = 1000$
N2S Reconstructions
N2S Projections
N2S Sinograms

Figure 3.9: From top to bottom, results on the central slice of the foam phantom of Noise2Self applied to reconstructed, projection, and sinogram images. For comparison, the insets show the output of Noise2Inverse (yellow) and Noise2Self (red).

tigated. Here, i.i.d. Gaussian noise was added to the projection images, and a reconstruction was computed afterwards. The PSNR of this noisy reconstruction with respect to the clean reconstruction was 11.59. When Noise2Self was applied to the noisy reconstruction, it obtained a PSNR of 16.14, substantially less than in the first case. This is displayed in the bottom row of Figure 3.8.

The results in Figure 3.8 demonstrate that the performance of Noise2Self is substantially degraded when the noise is not element-wise independent. Even though the starting PSNR in the bottom row is slightly higher, the PSNR improvement is only half of the top row. In the top row, the validation error continued to improve for 100 epochs, whereas in the bottom row, training started to overfit to the noise within the first 10 epochs of training, which could be caused by the statistical dependence between the input and target images.

Noise2Self on sinogram and projections To mitigate the effect of coupled noise, Noise2Self was also applied to images that do satisfy the pixel-wise independence property: the projection images and sinograms. In these cases, Noise2Self was first applied to denoise the raw images, and reconstructions were computed from the denoised projection images or sinograms.

As can be seen in Figure 3.9, the variations of Noise2Self did improve results, but not beyond Noise2Inverse. Although applying Noise2Self on the projection and

Absorption	I_0	Method	PSNR	SSIM
10%	100	N2S Reconstructions	6.37	0.27
		N2S Projections	16.43	0.44
		N2S Sinograms	*16.98*	*0.45*
		Noise2Inverse	**19.71**	**0.78**
50%	10	N2S Reconstructions	9.12	0.20
		N2S Projections	17.49	0.49
		N2S Sinograms	*18.06*	*0.51*
		Noise2Inverse	**21.66**	**0.79**
10%	1000	N2S Reconstructions	15.39	0.50
		N2S Projections	19.57	*0.62*
		N2S Sinograms	*20.62*	0.60
		Noise2Inverse	**26.25**	**0.89**

Table 3.3: Comparison of Noise2Self results on reconstruction, projection, and sinogram images.

sinogram images did accurately denoise the raw images, the resulting reconstructions of these denoised images exhibited some blurring (projections) and streaks (sinograms). As displayed in Table 3.3, the Noise2Self-based method with the best metrics, Noise2Self on sinograms, obtains PSNR on par with TV-MIN and SSIM worse than TV-MIN, see Table 3.1.

3.3.5 Noise2Inverse and missing wedge artifacts

The quality of tomographic reconstructions may be degraded due to artifacts other than measurements noise, such as missing wedge artifacts. These artifacts arise when projection data is acquired along an arc spanning less than 180°. The results in Section 3.2.1 predict that Noise2Inverse preserves these artifacts. To test this prediction, we apply Noise2Inverse to a foam dataset where the reconstructions are computed from 400 projection images along an arc of approximately 60°. Noise is applied consistent with an absorption of 10% and an incident photon count of 1000 photons per pixel. As can be seen in Figure 3.10, Noise2Inverse accurately denoises the reconstructed image, but leaves the missing wedge artifacts intact.

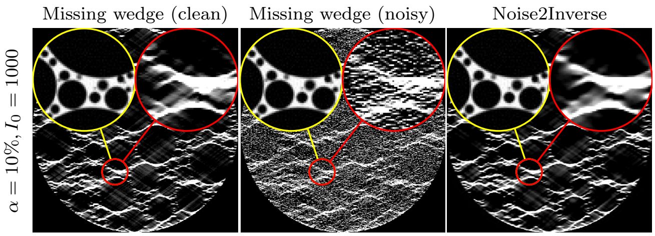


Figure 3.10: Noise2Inverse applied to a noisy dataset with missing wedge artifacts. The red and yellow insets show an enlarged view of the output and ground-truth, respectively.

3.3.6 Hyper-parameters

We analyzed how the effectiveness of Noise2Inverse was influenced by the number of splits, training strategy, number of projection angles, and neural network architecture. In addition, we tested the generalization by training on subsets of the data.

The same foam phantom was used, and noisy projection data were acquired from 512, 1024, and 2048 angles, of which the first and last acquisitions were undersampling and over-sampling the projection angles, respectively. For each dataset, the total number of incident photons remained constant: we used $I_0 = 400, 200, 100$ for $N_\theta = 512, 1024, 2048$, respectively. The average absorption was 23%, which is the default value of the `foam_ct_phantom` package.

Splits and strategy The performance of the Noise2Inverse method was evaluated with a number of splits $K = 2, 4, 8, 16, 32$, and with strategies X:1 and 1:X, see Equations (3.30) and (3.31). These experiments were performed with MS-D networks, which were trained for 100 epochs, and used the same training procedure as before.

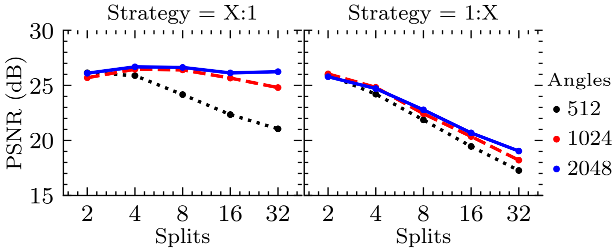


Figure 3.11: The PSNR metric for the Noise2Inverse method with the MS-D network applied on the foam phantom with varying number of splits, angles, and varying input-target splitting strategies. The X:1 strategy attains higher PSNR than the 1:X strategy.

The PSNR metrics are displayed in Figure 3.11. The figure shows that the X:1 strategy yields considerably better results than the 1:X strategy, except for $K = 2$, where they are equivalent. Setting the number of splits to $K = 2$ yields good results across the board, but the PSNR can be improved by setting K to 4 or 8, if the projection angles are not under-sampled. In general, the figure shows that increasing the number of acquired projection images can improve reconstruction quality without increasing the photon count. On the other hand, we note that reducing the number of projection images further can reduce the reconstruction quality as the artifacts arising from undersampling are not removed by the neural network.

Neural network architectures We compared three neural network architectures: the U-Net [156], DnCNN [199], and the previously described MS-D [143] network architectures, all of which were implemented in PyTorch [139].

The U-net is based on a widely available open source implementation[1], which

[1] `https://github.com/milesial/Pytorch-UNet/`

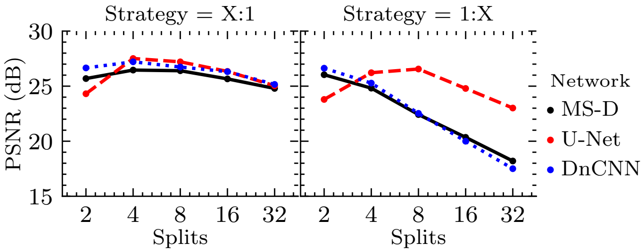


Figure 3.12: Comparison of the PSNR metric. The MS-D, U-Net, and DnCNN networks were trained for 100 epochs on the foam phantom with 1024 projection angles.

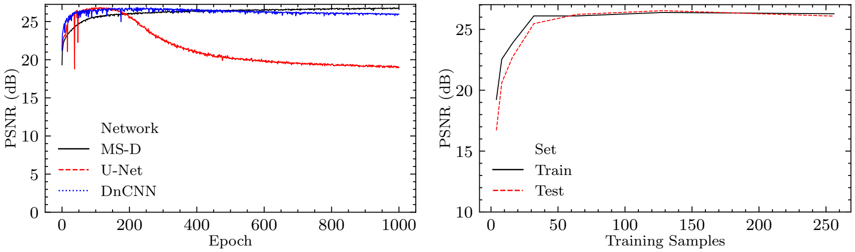
is a mix of the architectures described in [38, 156]. Like [156], the images are down-sampled four times using 2 × 2 max-pooling, the “up-convolutions” have trainable parameters, and the convolutions have 3 × 3 kernels. Like [38], this implementation uses batch normalization before each ReLU, the smallest image layers are 512 channels instead of 1024 channels, and zero-padding is used instead of reflection-padding. The resulting network has 14,787,777 trainable network parameters.

We used the DnCNN implementation from [14] with a depth of 20 layers, which is advised for non-Gaussian denoising [199]. The resulting network has 667,008 trainable network parameters.

The previous experiment was repeated on the dataset containing 1024 projection images. The networks were trained for 100 epochs, and used the same training procedure as before. The results are displayed in Figure 3.12. The figure shows that the U-net achieved overall highest performance using the X:1 strategy with 4 splits. In addition, the effect of the number of splits K is roughly the same across strategies and network architectures, except for U-net. In fact, the PSNR metric of the U-Net with the 1:X strategy initially increases when K is increased, which might be due to the large network architecture and number of parameters compared to the other two neural network architectures. Nonetheless, the X:1 strategy consistently attains higher PSNR than the 1:X for the U-net as well. We note that the U-Nets performed worse than the other networks with 2 splits, which suggests that training might have overfit the noise.

Overfitting We tested if the networks overfit the noise when trained for a long time. All three networks were trained for 1000 epochs using the X:1 strategy and $K = 4$ on the same foam dataset with 1024 projection angles. The resulting PSNR on the central slice as training progressed is displayed in Figure 3.13a. The figure shows that U-Net and DnCNN started to fit the noise, whereas the PSNR of the MS-D network continued to increase. This matches earlier results on overfitting [73, 140, 143]. If the training dataset had been larger, these effects could have been less pronounced.

Generalization We tested whether the network could be trained on fewer data samples and generalize to unseen data. We used the 1024-angle foam dataset, the MS-D network, 4 splits, and the X:1 strategy. The network was trained on the



(**a**) Overfitting.

(**b**) Generalization.

Figure 3.13: **a)** The PSNR on the central slice as training progressed. A U-Net, DnCNN, and MS-D network were trained with the X:1 strategy and number of splits $K = 4$ for 1000 epochs on the foam phantom reconstructed from 1024 projection angles.
b) An MS-D network was trained on subsets of the data. The PSNR on the training set (black) and test set (remaining data; red) are displayed.

first 4, 8, 16, 32, 64, 128, and 256 slices of the data. We report PSNR metrics on this *training set* and on the remaining slices, which we refer to as the *test set*. The number of epochs was corrected for the smaller dataset size, such that all networks were trained for the same number of iterations. When the training set exceeds 32 slices, the PSNR on the training and test set is comparable, as can be seen in Figure 3.13b.

3.4 Discussion and conclusion

The results show that the proposed Noise2Inverse method outperforms conventional reconstruction algorithms SIRT and TV-MIN by a large margin as measured in PSNR and SSIM. This improvement is accomplished despite optimizing the hyper-parameters of SIRT and TV-MIN on the clean reconstruction and without likewise optimizing the Noise2Inverse hyper-parameters. In addition, Noise2Inverse is able to significantly reduce noise in challenging real-world experimental data, improving on the visual impression obtained by SIRT and TV-MIN.

Extending the Noise2Self framework[14], we describe a general framework for denoising linear image reconstructions that provides a theoretical rationale for the success of our method. The framework shows that clean reconstructions may be recovered from noisy measurements *without observing clean measurements*, under the common assumption that the measured noise is element-wise independent and mean-zero. We remark that in low-noise situations, the trained network does not introduce additional artifacts in its output, as predicted by the theory.

We now focus on the comparison between the proposed Noise2Inverse approach and the existing Noise2Noise and Noise2Self approaches. As in Noise2Noise, the network is presented with two noisy images during training. In Noise2Inverse, however, these images are sub-sampled reconstructions, and since the artifacts arising from sub-sampling the data are correlated, the input and target images are not statistically independent — although the *reconstructed noise* in these images is statistically independent. Therefore, our results fall outside of the Noise2Noise framework. As in Noise2Self, Noise2Inverse trains a denoiser from unpaired measurements. The key difference is that the noise is element-wise independent in the measurement domain, rather than in the reconstruction domain, where denoising takes place. Therefore, the results from [14] do not carry over to the inverse problems setting. However, we are able to prove Proposition 1 using essentially similar arguments to those in [14].

The framework points the way to new applications of Noise2Inverse to linear image reconstruction methods. The implementation of Noise2Inverse for tomography shows that several aspects are worth considering. If reconstruction artifacts arise in the absence of noise, they will be preserved. In addition, if the reconstruction algorithm filters the noise at the expense of resolution, this will cause blurring in the output of our method. Moreover, splitting the measurement uniformly can avoid biasing the output of the method towards a particular subset of the measured data. Finally, the performance of the neural network can be improved by ensuring that the sub-reconstructions are homogeneously informative throughout the image.

Noise2Inverse is well-suited to imaging modalities that permit trading acquisition speed for measurement noise, as it aims to remove measurement noise but does not remove artifacts resulting from under-sampling, Whether this trade-off is possible, depends on the specifics of the imaging modality. Tomographic acquisition, for instance, permits acquiring the same number of projection images by lowering the exposure time at the cost of increased noise [123]. Magnetic Resonance Imaging (MRI), on the other hand, is usually accelerated by reducing the number of measurements, rather than by acquiring noisier measurements [96]. Examples of imaging modalities that permit trading speed for noise include ultrasound imaging [121], deconvolution microscopy [167], and X-ray holography [197].

The comparison of Noise2Inverse with Noise2Self demonstrates that the success of our method depends not only on considerations of statistical independence, but also on taking account of the physical forward model. Regarding statistical independence, we have demonstrated that a straightforward application of Noise2Self fails on noisy tomographic reconstructions due to coupling of the noise. Regarding the forward model, we have investigated a two-step approach, where Noise2Self is applied to projection or sinogram images — which *do* satisfy the element-wise independence requirement — before reconstructing. This approach performs worse than TV-MIN and Noise2Inverse in terms of visual impression and quality metrics. This matches earlier results [27], and could result from the fact that the consistency of the projection and sinogram images with respect to the forward operator is not necessarily preserved. These results suggest that taking into account the properties of the inverse problem — as Noise2Inverse does — significantly improves the quality of the reconstruction.

Several variables affect the performance of Noise2Inverse. Most importantly, the training strategy that reconstructs the input images from at least as many projection angles as the target images — the X:1 strategy — yields better results than vice versa. This conclusion holds regardless of network architecture, number of splits, or number of projection angles. This suggests that noise in the gradient is less problematic than noise in the input for neural network training, as was observed before [107]. Another variable that consistently predicts performance is the number of angles; acquiring more projections yields a small but consistent performance boost. The number of parts in which the measured data is split, however, deserves more nuance: when the projection angles are under-sampled, the results indicate that two parts yield the best results; otherwise, splitting into more parts yields better results. Finally, maximal performance can be obtained by tuning the neural network architecture and number of training iterations. When tuning is not an option, an MS-D network can be trained with limited risk of overfitting the noise. Finally, the object under study influences the comparative advantage of our method to conventional reconstruction techniques. When the aim is to retrieve low-contrast details from low-noise reconstructions, the difference may be minimal. When the object is self-similar and the noise has high intensity, on the other hand, our method can significantly outperform other methods.

In conclusion, we have proposed Noise2Inverse, a CNN-based method for denoising linear image reconstructions that does not require any additional clean or noisy data beyond the acquired noisy dataset. On tomographic reconstruction problems, it strongly outperforms both standard reconstruction techniques such as Total-Variation Minimization, and self-supervised image denoising-based techniques, such as Noise2Self. We also demonstrate that the method is able to significantly reduce noise in challenging real-world experimental datasets.

4 Noise2Inverse for synchrotron tomography

"That solution which appears the simplest, turns out to be the hardest in practice."

"Die op het oog simpele oplossing, blijkt in de praktijk het moeilijkst."

Johan Cruijff
Vrij Nederland, 21 Dec 1974

Synchrotron-based X-ray tomography is a powerful technique for investigating the internal structure of objects with applications in energy research, materials science, life sciences, and many other fields [146, 152, 173]. Thanks to the high photon flux available at synchrotrons, experiments can be performed at speeds and microscopic scales that push the envelope of scientific imaging [56, 77]. In these experiments, a rotating object is located in the path of an X-ray beam, and its projection is measured on a detector. A representation of the object is reconstructed from a series of its projection images. The projection images typically suffer from noise, which carries over to the reconstructed images. Noise can be reduced by increasing the exposure time or photon flux, resulting in a higher dose. In many cases, however, dose and exposure time are limited by unavoidable experimental constraints, such as fast sample dynamics, radiation damage, and alteration of system properties during prolonged exposure to high-flux synchrotron

This chapter is based on:
A. A. Hendriksen, M. Bührer, L. Leone, M. Merlini, N. Vigano, D. M. Pelt, F. Marone, M. di Michiel, and K. J. Batenburg. "Deep Denoising for Multi-Dimensional Synchrotron X-Ray Tomography Without High-Quality Reference Data". *Scientific Reports* 11.1 (2021).

X-ray beams [23, 29, 50, 56, 113, 157]. In such experiments, accurate denoising of the reconstructed images is a central problem.

Among techniques for removing noise from reconstructed images, deep convolutional neural network (CNN)-based methods have shown strong results, often outperforming conventional state-of-the-art techniques [5, 85, 140]. A substantial subset of these techniques are known as post-processing techniques. Here, a reconstruction is first computed using a fast reconstruction algorithm and a CNN is used to post-process the reconstructed image. Other approaches, such as learned iterative techniques [5] also exist, but are generally not as computationally efficient. Most proposed methods, however, require *supervised training* before they can be applied to the problem at hand. That is, the networks are trained by using paired examples of noisy input reconstructions and high-quality target reconstructions.

In practice, obtaining the set of paired noisy and high-quality reconstructions required for supervised training is a major obstacle [16]. First, obtaining low-noise reconstructions may not be possible, due to the aforementioned experimental constraints. Second, accurate pairing of two different datasets may require time-consuming manual labor to register the images, especially because the accuracy of registration directly impacts the output quality of the trained network [166]. Sub-pixel inaccuracies may cause blurring of the network output, for example. Finally, with fewer than 50 synchrotron facilities worldwide, beamtime is a scarce resource [190], and acquiring additional measurements may simply be too expensive.

To sidestep the issue of obtaining training data, CNN-based denoising techniques have recently been proposed that do not require the acquisition of high-quality images [14, 27, 97, 100, 107, 112, 149, 196]. However, many of these techniques rely on assumptions about the source of noise that are not correct in tomography, resulting in suboptimal denoising accuracy [75]. As a solution to these difficulties, we have proposed Noise2Inverse [75], which is a post-processing technique specifically designed for tomography and related inverse problems. Using *self-supervised* training, the acquisition process is exploited to create pairs of noisy reconstructions from a single tomographic dataset. Although no high-quality reconstructions are presented to the network, training still produces a denoising CNN whose accuracy is comparable to supervised CNN training [75]. To obtain a denoised reconstruction, the trained CNN can be applied both to the noisy training reconstruction and to similar data that is acquired in the future. The Noise2Inverse method can be applied in the aforementioned cases where collecting a training set is found to be an obstacle.

To obtain a denoising CNN, the reconstructed training pairs must satisfy certain conditions, which we describe in this chapter and refer to as the *Noise2Inverse conditions*. In previous work, analysis of the Noise2Inverse conditions was limited to single-slice (2D) parallel-beam tomography [75], treating multi-slice (3D) tomographic datasets as a stack of independent 2D problems. In the synchrotron community, however, 3D tomography is routine and many important research questions can only be answered using more advanced imaging techniques that make use of additional information in the time or spectral domain of the signal [10, 22, 23, 28, 29, 50, 56, 77, 87, 113, 157, 189]. In this chapter, we extend the analysis of the Noise2Inverse conditions to such techniques, including micro-tomography (space), dynamic micro-tomography (time), as well as X-ray diffraction computed tomography (diffractogram, spectrum-like). Taking into account the Noise2Inverse conditions, we tailor training procedures to each of these techniques. Our description of the training procedures for these imaging techniques may serve as a template for other use cases. In addition, we discuss how the acquisition can be refined to optimize the results of Noise2Inverse denoising. Finally, we apply Noise2Inverse to real-world datasets, compare it to commonly used reconstruction techniques, and investigate the possibility of reducing acquisition time without loss of image quality.

This chapter is structured as follows. First, we introduce the basics of CNN-based denoising. Next, we describe the Noise2Inverse method for single-slice (2D) parallel-beam tomography. Then, for each of the three domains (space, time, and diffractogram) and corresponding imaging technique, we describe a Noise2Inverse training procedure that exploits the acquisition process to obtain a denoising CNN. Next, we present results of the application of Noise2Inverse to real-world datasets of each of these techniques, and conclude with a discussion.

4.1 CNN-based denoising

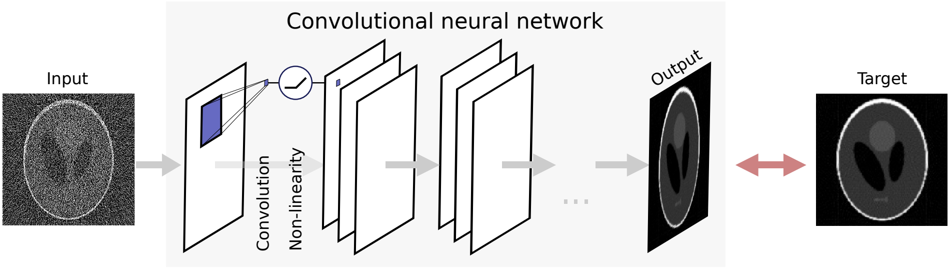


Figure 4.1: A prototypical convolutional neural network (CNN) in a supervised training arrangement for denoising. The CNN is composed of multiple layers. To compute an image in the next layer, images in the previous layer are convolved with a small filter (depicted in blue), after which a non-linear function is applied. To train a CNN to perform denoising, the output image is compared to a target image, and the convolution filters are updated accordingly.

This section briefly introduces convolutional neural networks (CNNs) for denoising. For a more detailed overview, we refer the reader to [122, 199]. CNNs are usually organized in layers: the input image is copied into the first layer, the output image is the final layer, and computations are performed in the intermediate layers. The CNN computes convolutions that are parameterized by thousands to millions of small filters (typically 3×3). In each intermediate layer, the images in the previous layer are convolved with these filters, after which a pixel-wise non-linear function is applied. A common non-linearity is the ReLU function, which is the identity for positive arguments and zero otherwise [79, 133]. A prototypical CNN is illustrated in Figure 4.1. The input to the network can consist of multiple 2D channels, for example when the input is a color image. Similarly, the intermediate layers and output may consist of multiple channels — regardless of the number of channels in the input.

The network can be prepared to perform denoising by supervised training. During training, the network is presented with noisy input images and noise-free target images from a training dataset. On each image pair, the parameters of the convolutions are optimized to minimize the training loss, i.e., the difference between the output and target image. The training loss is commonly quantified by the mean square error [66] and minimized using stochastic gradient descent [153].

If the number of parameters of a CNN is large compared to the amount of training data, there is a risk of overfitting. In such situations, the network starts to fit features particular to the training data which causes its accuracy to suffer on its intended use case [16]. A practical remedy is to monitor the results of the network on a separate validation dataset, and perform early stopping when the results start deteriorating. Early stopping may not be necessary, however, if the number of training pixels vastly exceeds the number of CNN parameters [66].

In tomography, the reconstructed data is often volumetric rather than a single 2D image. In the volumetric case, it has been observed that image quality can be improved by using 3D CNNs instead of 2D CNNs [201]. Here, the input and output are a 3D volume, and intermediate computations are performed using 3D convolutions. 3D CNNs, however, impose computational demands that can be prohibitive in practice. As a solution to this problem, 2.5D CNNs have been proposed. These networks can often achieve image quality similar to 3D CNNs at a reduced computational cost [201]. We refer to a 2D sub-image of a 3D volumetric dataset as a *slice*. The input to a 2.5D CNN is a stack of 2D slices, and its output is a 2D slice. In addition to the current input slice, the network input consists of context slices located above and below the current slice. This technique is applied to 3D tomographic datasets in Section 4.2.2.

As described before, the collection of paired image data can be a major obstacle to the use of supervised training. In the next section, we describe how a denoiser can be trained without such a training dataset.

4.2 Method

In this section, we discuss the Noise2Inverse method. First, we summarize the procedure for single-slice (2D) parallel-beam tomography and describe the conditions under which Noise2Inverse training is possible [75]. Next, we describe how Noise2Inverse can be extend in space, time, and spectrum-like domains so that it can be applied to imaging techniques in common use at synchrotron facilities.

4.2.1 Noise2Inverse on single-slice parallel-beam tomography

In single-slice parallel-beam tomography, a rotating object is located in the path of a planar beam, and its projection is measured on a line detector, as displayed in Figure 4.2. A 2D image, called the sinogram, is collected from readouts at a series of angles. From the sinogram, a 2D image (slice) of the object can be computed using a reconstruction algorithm [31].

The Noise2Inverse procedure for parallel-beam tomography was comprehensively introduced in [75]. First, the acquired sinogram is split into multiple *target sections* in the domain of the rotation angle. Each target section has a matching *input section* that covers the remainder of the sinogram. As displayed in Figure 4.2 (**b**), the target sections split the sinogram in such a way that adjacent measurements in the angular domain are in different target sections. During CNN training, the input and target images are reconstructed from the corresponding sections of the sinogram. The CNN thus learns to map noisy reconstructed images to each other. The training process is displayed in Figure 4.2.

After training, the reconstructed images are post-processed using the CNN. In this chapter, we describe a modification to the post-processing step described in the previous chapter. There, the CNN was applied to reconstructions of the input sections separately, resulting in several denoised intermediate images, of which the average formed the output of the algorithm. Here, on the other hand, the output is computed by directly applying the CNN to the reconstruction of the full sinogram. Empirically, this modification consistently improves results [14], although no theoretical explanation for its success has yet emerged.

To train a denoising CNN, the Noise2Inverse method requires that the reconstructed training pairs satisfy the following two **Noise2Inverse conditions**:

1. The noise in the reconstructed input image is statistically independent from noise in the reconstructed target image.

2. Every subset of the measurement is used in the reconstructed target images equally often.

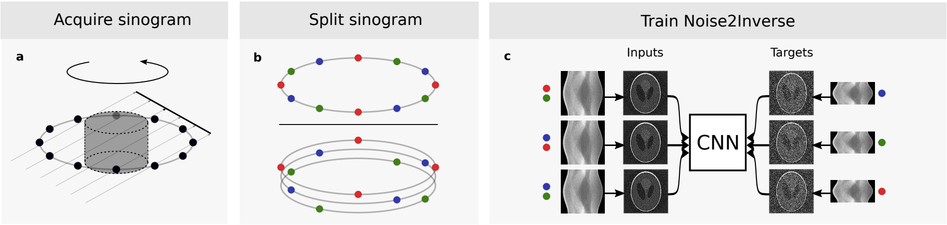


Figure 4.2: Noise2Inverse for single-slice parallel-beam tomography. (**a**), A sinogram is acquired of a rotating object. (**b**), The sinogram is split into sections (red, green, and blue) in the domain of the rotation angle. (**c**), The input and target images for training the network are reconstructed from distinct sections of the sinograms (indicated by the red, green, and blue dots). The input and target sections cover $^2/_3$ and $^1/_3$ of the sinogram, respectively.

The Noise2Inverse conditions are satisfied in the case of single-slice parallel-beam tomography. The noise in the reconstructed input and target images is independent ("uncorrelated"), because the images, and thus the noise, are reconstructed from distinct input and target sections of the sinogram. Because target sections from all parts of the sinogram are used during training, the second requirement is satisfied as well. This prevents biasing the result towards a specific subset of the measurements.

Formally, the first condition makes it possible to write the training loss as the sum of two terms. Consider a noisy measurement and its corresponding unknown noise-free sinogram. Split the measurement into J target sections and denote by $x_{j,\text{noise-free}}$ the reconstruction of the j-th target section of the noise-free sinogram. Denote by $x_{j,\text{noisy}}$ the reconstruction of the j-th target section of the noisy measurement and by $x_{\cdot \neq j,\text{noisy}}$ the corresponding input reconstruction. The training loss can then be decomposed as a sum of two terms [75]

$$\begin{aligned} &\overbrace{\sum_{j=1}^{J} \|\text{CNN}(x_{\cdot \neq j,\text{noisy}}) - x_{j,\text{noisy}}\|_2^2}^{\text{Noise2Inverse training loss}} \\ &\qquad \overset{\mathbb{E}}{=} \sum_{j=1}^{J} \underbrace{\|\text{CNN}(x_{\cdot \neq j,\text{noisy}}) - x_{j,\text{noise-free}}\|_2^2}_{\text{Loss w.r.t. noise-free reconstructions}} + \underbrace{\|x_{j,\text{noise-free}} - x_{j,\text{noisy}}\|_2^2}_{\text{Variance of the noise}}, \end{aligned} \tag{4.1}$$

where $\overset{\mathbb{E}}{=}$ indicates that the equality holds on average over the statistical distribution of the noise and the imaged objects. On the right hand side, the first term represents the difference between the CNN output and the noise-free reconstructions, and the second term represents the variance of the noise in the reconstructed images, which does not depend on the CNN. Minimizing the Noise2Inverse training loss therefore minimizes the loss with respect to the noise-free reconstructions, which is the goal of denoising.

Equation (4.1) holds in combination with any linear reconstruction algorithm. Most common reconstruction algorithms in synchrotron tomography fall into this category, e.g., FBP, GridRec, and SIRT [31, 60, 118]. An additional requirement is that the measurement noise is zero-mean. This is typically satisfied by sources of noise in X-ray synchrotron tomography, such as Poisson or additive Gaussian noise. An example of noise that is not zero-mean is salt and pepper noise, which randomly corrupts pixels by setting them to zero or a fixed high value.

Intuitively, Noise2Inverse and other self-supervised methods rely on the ability of CNNs to find correlations between images. During training, the content of both the input and the target reconstruction is determined by the structure of the object — resulting in a strong structural correlation between the images — and the noise, which is uncorrelated. Therefore, the network is rewarded when it makes a prediction using the structure of the object. If it picks up a correlation based on the noise, it may be punished when this spurious correlation is absent in later iterations. During training, the network thus learns the statistical distribution of noiseless reconstructions. This distribution encompasses all features of the reconstructed images that are not related to the measurement noise. Therefore, when systematic imperfections in the measurement carry over into the reconstructed images, the network will not learn to remove them. This can occur, for instance, when the angular sampling of the full sinogram has insufficient resolution, as described in [75].

If the Noise2Inverse conditions are not satisfied, then the output of the trained CNN can be sub-optimal. First, if the noise in the reconstructed input and target images is not independent, then the CNN will learn to reintroduce noise in its output. This can happen when the input and target sections of the measured data partially overlap, for instance. Second, when not every subset of the measurement is used in the target image equally often, then the CNN may be trained to overemphasize reconstruction of certain sections of the measured data, resulting in biased output.

4.2.2 Noise2Inverse on synchrotron X-ray tomography

In this section, we propose Noise2Inverse training strategies tailored to imaging techniques in common use at synchrotron facilities. We discuss extensions in the spatial domain (3D micro-tomography), time domain (dynamic micro-tomography), and spectrum-like domain (X-ray diffraction tomography). In addition, we discuss how the training strategies satisfy the Noise2Inverse conditions.

Static 3D micro-tomography In 3D micro-tomography, the beam is measured on a 2D detector instead of a line detector. In the parallel-beam case, this problem can be viewed as a stack of single-slice problems. Therefore, the strategy from the previous section could be pursued, using reconstructed image pairs from the entire stack to train a single 2D CNN [75]. As discussed in Section 4.1, however, the image quality can be further improved by using a 2.5D CNN [201]. This network's input consists of context slices located above and below the current slice, as illustrated in Figure 4.3 (**c**).

The Noise2Inverse strategy can be used with 2.5D CNNs. After acquisition of a stack of sinograms, we propose that each sinogram is split in the angular domain to obtain target sections and complementary input sections. In each training iteration, the target image is a single slice reconstructed from a target section. The input consists of an input slice and context slices reconstructed from the corresponding input section. The procedure is illustrated in Figure 4.3. This strategy preserves the Noise2Inverse conditions, as the noise in the input and target is reconstructed from distinct sections of the sinograms, and each part of the sinogram is used as target section.

Dynamic micro-tomography In dynamic tomography, a full scan of a dynamically evolving process is made at several time steps. In a single time step, movement of the object can result in motion artifacts, severely degrading the quality of the reconstructed image. To prevent such artifacts, the ideal scan time of a time step is bounded by the speed of the evolving process [29, 56]. Additionally, the number of projection images per time step may be limited by the maximum frame rate of the detector. Therefore, both the exposure time and the number of projection images per rotation may be limited, and a reconstruction of a single

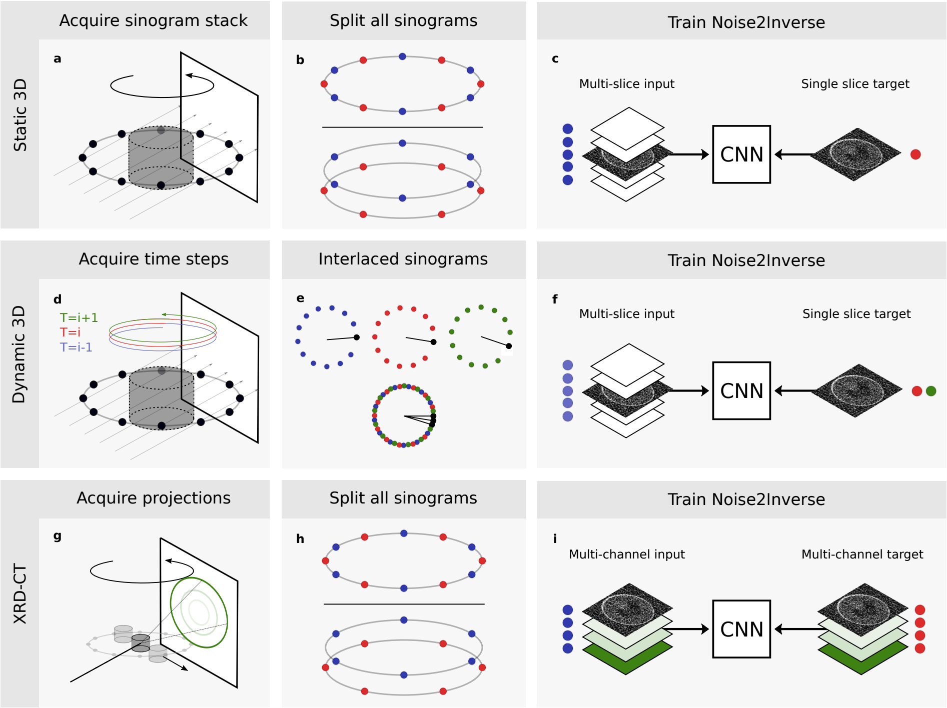


Figure 4.3: Noise2Inverse training procedures for several imaging techniques. **a–c**, Static 3D micro-tomography. Acquisition produces a stack of sinograms (**a**), each of which is split in the angular domain (**b**). A 2.5D-CNN is supplied with the current input slice and several context slices (shown in white) reconstructed from one part of the sinogram (blue dot). The target slice is reconstructed from a different part of the sinogram (red dot). **d–f**, Dynamic micro-tomography of an evolving process. Scans of multiple time steps are acquired (**d**). With interlaced acquisition, the angular sampling of each time step is slightly displaced with respect to the previous time step (indicated by the "clock's hands") (**e**). Denser angular sampling is achieved by combining the sinograms of multiple time steps (**e**). A 2.5D-CNN is trained on time steps where the dynamic process has not yet started (**f**). The input is a reconstruction of a single time step (blue dot), and the target is reconstructed from the sinograms of the other time steps (red and green dots). **g–i**, X-ray diffraction computed tomography (XRD-CT). A pencil beam probes a rotating object that is moved sideways in steps (**g**). Diffraction of the beam gives rise to rings on the detector (in green). Azimuthal integration along rings at different radii yields several sinograms for each slice of the object. These sinograms are split in the domain of the rotation angle (blue and red dots) (**h**). From these sinograms, a multi-channel reconstruction is computed, representing the diffractogram of the object (in shades of green) (**i**). Training is performed with multi-channel inputs and multi-channel targets, with inputs and targets reconstructed from distinct parts of the sinograms (blue and red dots).

time step may suffer from sparse angle artifacts [83].

When angular undersampling occurs in dynamic tomography, Noise2Inverse benefits from interlaced angular sampling [3]. Here, the angular sampling at each time step is displaced from the sampling at the previous time step. By combining sinograms of multiple time steps, this technique enables denser angular sampling to be achieved, as illustrated in Figure 4.3 (**e**). From the combined sinogram, an accurate reconstruction can be computed, possibly even avoiding motion artifacts where and when the object is static. As discussed in Appendix A.1, it is likely that a fast dynamic scan uses interlaced angular sampling, either deliberately or as a consequence of minimal mechanical inaccuracies.

Typically, the dynamic process does not start immediately during the acquisition. This creates a window of opportunity to train a CNN on the first few time steps. Here, interlaced sampling mitigates the effects of undersampling, and motion artifacts are avoided, since the dynamic process has not yet started. We propose that the sinogram stacks of these time steps are combined into a single sinogram stack. The sinogram stack is subdivided such that each input section covers a single time step, and the corresponding target section covers the remaining time steps. During training, the input and target are reconstructed from the input and target sections. This process is displayed in Figure 4.3 ($\mathbf{d} - \mathbf{f}$), where three time steps are combined into a single sinogram stack, and two time steps are used to reconstruct the target image in a round-robin fashion. The number of time steps should preferably be chosen such that the interlaced sampling distributes the angles of the projection images evenly over a 360° arc. As in the static 3D case, 2.5D CNN training can be used. After training, the network is applied to each reconstructed time step to obtain a sequence of denoised reconstructions. A similar approach has been described for dynamic magnetic resonance imaging (MRI) [112].

This training strategy preserves the Noise2Inverse conditions. The noise in the reconstructed input and target images is independent as the images have been reconstructed from distinct time steps. The second property is satisfied, as each time step occurs in the target images equally often. Note that in contrast to the previous cases, the target is reconstructed from *more* measurements than the input, since the target is computed using multiple time steps. This enables the target reconstructions to achieve denser angular sampling and causes the network to mitigate sparse angle artifacts.

X-ray diffraction computed tomography X-ray diffraction tomography (XRD-CT) makes it possible to non-destructively identify the crystallographic phases that compose a material, using the principles of X-ray powder diffraction [10, 22]. As illustrated in Figure 4.3 (**g**), the acquisition proceeds by probing a rotating object with a pencil beam, repeating the acquisition as the object is moved sideways in steps. The diffraction signal produced by the sample is recorded on a 2D detector, on which rings correspond to the scattering angle of the beam. By azimuthal integration along these rings, sinograms are computed for each scattering angle, forming 2D images indexed by the rotation angle and the sideways translation of the object. Performing the azimuthal integration with different radii yields several sinograms for one 2D horizontal slice of the object. From these sinograms, a multi-channel reconstruction is computed where each channel corresponds to a radius of the azimuthal integration. For each voxel in the 2D slice, these channels form a diffractogram, which is a non-spatial dimension analogous to a spectrum.

The Noise2Inverse strategy for X-ray diffraction tomography exploits the interdependence of the object in the domain of the diffractogram (across channels). The diffractogram at a location is determined by the specific crystallographic phase of a material at that location. Hence, if a material in a specific phase is present at multiple locations, its diffractogram — the values in the multi-channel reconstructed image — must correspond. The accuracy of denoising can be improved by exploiting this interdependence.

During Noise2Inverse training, we propose to split the obtained sinograms in the domain of the rotation angle to obtain input and target sections. Multi-channel reconstructions are computed from the input and target sections of the sinograms, which are used as input and target during training. This training strategy satisfies the Noise2Inverse conditions, as the input and target images are reconstructed from distinct sections of the sinograms, and the target sections cover the full sinogram.

In summary, Noise2Inverse can be adapted to make use of additional information in space, time, and spectrum-like domains. This enables its application to several imaging techniques in use at synchrotron X-ray facilities. In the case of static 3D micro-tomography, the 2D network input is extended with context slices to utilize 2.5D CNN training. In the case of dynamic 3D micro-tomography, the effects of sparse angular sampling are mitigated by reconstructing the target image from a combined sinogram of multiple time steps.

In the case of X-ray diffraction tomography, multiple channels represent the diffractogram of each voxel in a 2D slice of the object — not its vertical direction —, and thus serve a different purpose than in the previous cases.

4.3 Results

We applied the Noise2Inverse method to datasets acquired at two synchrotron beamlines. In particular, we compared the method to common reconstruction algorithms on a static and a dynamic micro-tomography dataset from the TOMCAT beamline at the Swiss Light Source (SLS). In addition, we investigated the possibility of accelerating the acquisition process using an X-ray diffraction tomography (XRD-CT) dataset from the ID15A beamline at the European Synchrotron (ESRF). The acquisition procedure for each dataset is described in Appendix A.1.

On each dataset, CNN training was performed using the same parameter-efficient mixed-scale dense network architecture (MS-D) [143] and the same hyper-parameters. Differences between the Noise2Inverse training procedures on each dataset are summarized in Table 4.1 and described in more detail in Appendix A.2.

Static 3D micro-tomography A static 3D micro-tomography dataset of a fuel cell was acquired. This dataset was part of a set of acquisitions aimed at imaging the water dynamics in the fuel cell. During operation, fuel cells generate water as

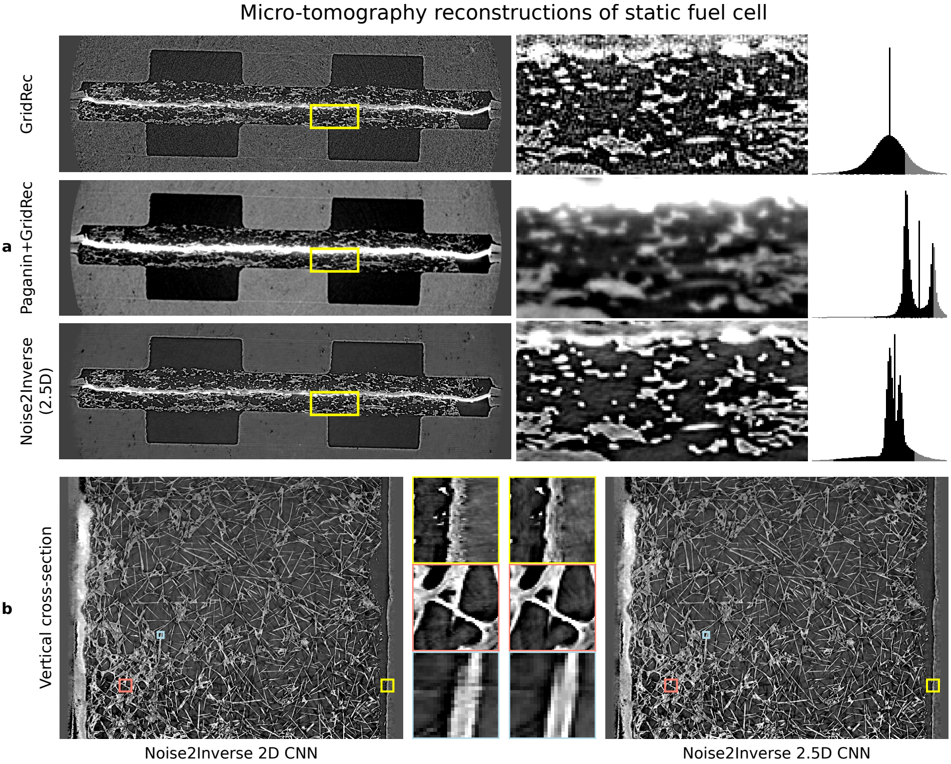


Figure 4.4: Comparison of reconstructions of a fuel cell using GridRec, Paganin+GridRec, and Noise2Inverse. **a**, Reconstructions of the static fuel cell. The yellow region of interest is magnified in the windows on the right. Gray level ranges have been adjusted to improve contrast and are indicated by the shaded region in the histograms on the right. **b**, A comparison of Noise2Inverse with a 2D CNN and a 2.5D CNN on a vertical cross-section of the reconstruction. The magnified regions of interest in the middle reveal discontinuities in the 2D CNN reconstruction.

a by-product. Suboptimal water management is currently a major limiting factor to sustained performance of the fuel cell at high current densities [29]. Accurate imaging of the water dynamics can inform improvements to the fuel cell design. This static dataset contained no water; a dynamic dataset of the operating fuel cell is discussed below.

We compared the Noise2Inverse reconstruction to reconstructions using the GridRec algorithm [118] with and without additional preprocessing. The preprocessing was performed using Paganin phase retrieval [137], and we refer to resulting reconstructions as Paganin+GridRec. For single material objects and monochromatic radiation, the Paganin algorithm [137] can be used to preprocess the projection images prior to tomographic reconstruction to obtain quantitative results free of edge-enhancement artifacts. In the synchrotron community, thanks to its robustness, the Paganin algorithm is also used when not all assumptions are strictly satisfied as a tool to boost contrast and decrease noise in tomographic reconstructions. In both static and dynamic experiments with polychromatic radiation and a multi-material object, the parameters for the Paganin algorithm were chosen so as to maximize contrast while limiting the degradation of spatial resolution. The sinogram stacks were computed using standard dark-field, flat-field, and log-correction from the raw projection data for the GridRec and Noise2Inverse reconstructions. In the Paganin+GridRec reconstruction, the dark-field and flat-field corrections were applied before Paganin phase retrieval, and the log-correction was applied afterwards. We emphasize that the GridRec and Paganin+GridRec reconstructions were computed from exactly the same measured data as the Noise2Inverse reconstruction. No additional measurements were acquired for the Noise2Inverse reconstructions.

The results are shown in Figure 4.4. The measurement noise was carried over into the GridRec reconstruction, and the Paganin+GridRec reconstruction was blurred. Image features in the Noise2Inverse reconstruction could be more easily distinguished, and the reconstruction did not suffer from noise or blurring. In addition, we trained a 2D CNN to illustrate the effect of not taking into account additional 3D information using a 2.5D CNN. Panel (**b**) displays a vertical cross-section of the resulting reconstruction that allows comparing between Noise2Inverse with a 2D CNN and with a 2.5D CNN. On this vertical reconstruction, we see that the 2D CNN reconstruction suffers from slight vertical discontinuities that are not present in the 2.5D CNN. The 2D CNN was not much faster to train than the 2.5D CNN.

Dynamic 3D micro-tomography A dataset was acquired of a fuel cell that was in operation. The speed of the water dynamics made it necessary to acquire a single time step 10 times faster than in the static case, resulting in a highly restricted exposure time and a restricted angular sampling frequency — due to the maximum frame rate of the detector. During the acquisition, the object was continuously rotating at a constant speed. As described in Appendix A.1, we observed a small deviation in the rotation speed as a result of a minor mechanical inaccuracy. This caused each time steps' 300 projection angles to be slightly displaced with respect to the previous time step, resulting in interlaced sampling. Training was performed on the first 3.6 seconds of the 18 s acquisition (36 out of 180 time steps).

A comparison of GridRec, Paganin+GridRec, and Noise2Inverse can be found in Figure 4.5. It shows a reconstruction of a horizontal slice, and several time steps of a magnified region of interest, in which the formation of a water bubble takes place. The GridRec reconstruction suffered severely from noise carried over into the reconstruction, and the Paganin+GridRec reconstruction was blurred, although the water dynamics were discoverable. The Noise2Inverse reconstruction was substantially sharper, and the water dynamics in the magnified views were clear. In panel (**b**), several crops show the difference in blurring between Noise2Inverse and Paganin+GridRec on a vertical cross-section.

XRD-CT An X-ray diffraction tomography dataset was acquired of an archaeological ceramic, whose fragments are kept at the University of Milan. The

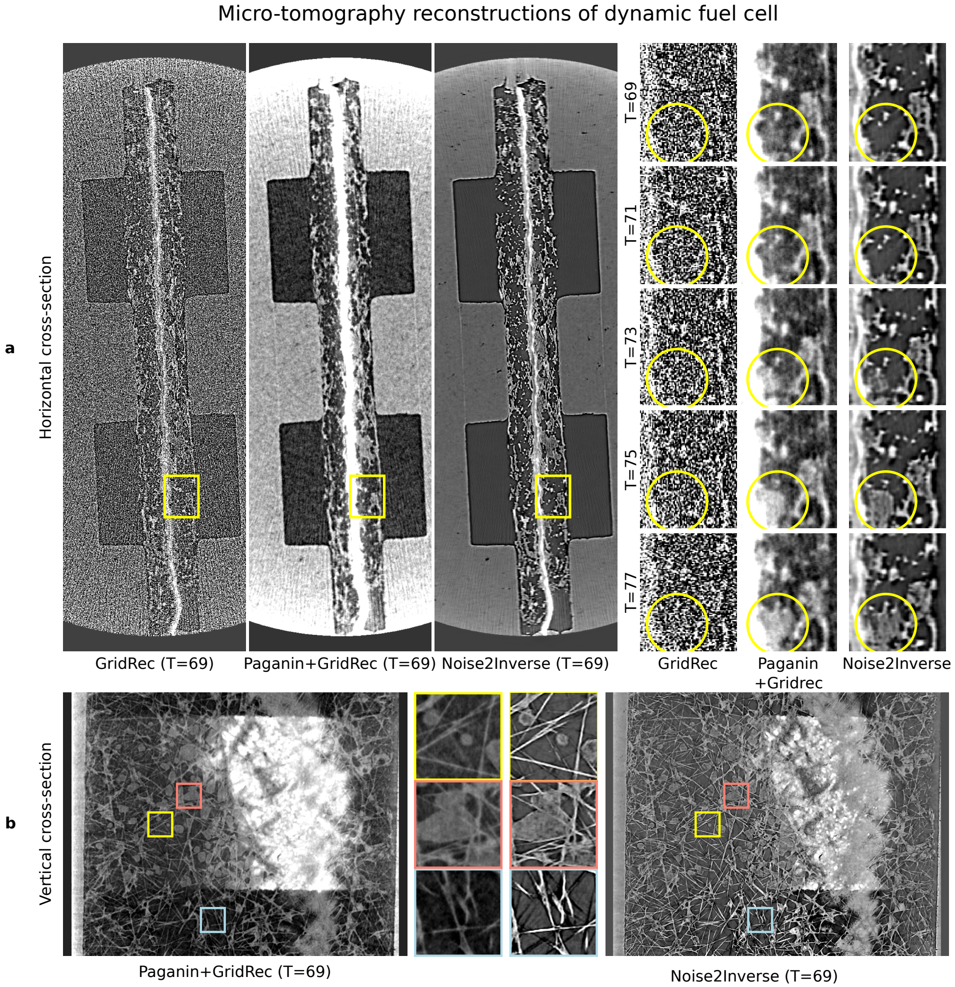


Figure 4.5: Comparison of reconstructions of an operating fuel cell using GridRec, Paganin+GridRec, and Noise2Inverse. **a**, Reconstructions of the fuel cell in operation. The yellow region of interest is magnified in the windows on the right, and shows the formation of a water bubble over several time steps. **b**, A vertical cross-section of the dynamic reconstruction is displayed with magnified regions of interest shown in the middle.

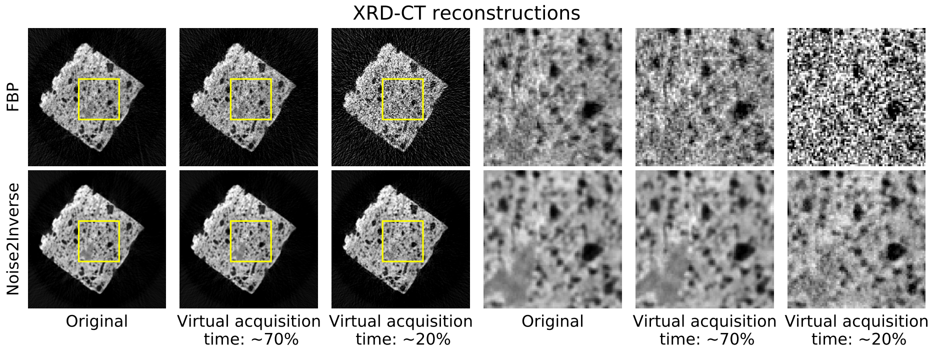

XRD-CT reconstructions
FBP
Noise2Inverse
Original
Virtual acquisition time: ~70%
Virtual acquisition time: ~20%
Original
Virtual acquisition time: ~70%
Virtual acquisition time: ~20%

Figure 4.6: X-ray diffraction tomography reconstructions of a single channel of a single slice of a ceramic fragment. The leftmost column shows the reconstruction of the originally acquired data, and the next two columns show reconstructions with synthetic noise. The rightmost three columns show magnifications of the yellow region of interest.

acquisition resulted in a dataset containing 3 horizontal slices with 11 channels each. Acquisition of each slice took 20 minutes. Two noisier datasets were obtained by applying synthetic noise to the sinograms. These were estimated to correspond to a virtual acquisition time of 70% and 20%, respectively, as described in Appendix A.3. The relative variance of the noise of the synthetic datasets is displayed in Figure A1.

The results are shown in Figure 4.6. As the virtual acquisition time was decreased, the quality of the FBP reconstruction suffered due to measurement noise that was carried over into the reconstruction. Compared to the FBP reconstructions, the degradation in image quality of the Noise2Inverse reconstructions was substantially less severe.

4.3.1 Training time, intermediate results, and overfitting

Differences of Noise2Inverse training between the three previous experiments are summarized in Table 4.1. The training times were realized on a system with 192 GB of RAM and four Nvidia GeForce GTX 1080 Ti GPUs. The reported training times are indicative, and are specific to the hardware. For instance, a networked storage bottleneck caused training on the dynamic dataset to take twice as long as on the static dataset, even though a similar number of training iterations was spent on both datasets.

Although training times were substantial for all three datasets, we note that training time did not increase linearly with the size of the training dataset. For instance, the dynamic dataset was 10, 000 times larger than the XRD-CT dataset, yet training time took less than 30 times longer. An important factor is that training time dominates reconstruction time: the static fuel cell took 20 hours to train, but inference took only 5 minutes for instance. As shown in Table 4.1, the CNNs had roughly the same number of parameters for all three datasets. That

	Static	Dynamic	XRD-CT
Noise2Inverse			
Channels (input, target)	11, 1	11, 1	11, 11
Relative section size (input : target)	1 : 1	1 : 5	2 : 1
Size			
Sinograms (32-bit float)	6.3 GB	342 GB	9.2 MB
Training volume size (voxels)	$6.86 \cdot 10^8$	$2.47 \cdot 10^{10}$	$2.46 \cdot 10^6$
CNN Parameters	$5.48 \cdot 10^4$	$5.48 \cdot 10^4$	$5.60 \cdot 10^4$
Duration			
Training iterations	220,000	171,600	30,000
Training + Reconstruction duration	∼ 20 hours	∼ 43 hours	∼ 90 minutes

Table 4.1: Details of Noise2Inverse training on the static and dynamic micro-tomography datasets of a fuel cell and X-ray diffraction tomography dataset of a ceramic. The relative section size reports the ratio of the size of the input section relative to the target section. The size of the training volumes varies widely, whereas the number of parameters in the CNN is stable. Reported training times are indicative, and are specific to the hardware.

may be the reason that the number of required training iterations does not scale with the size of the dataset, but reaches a point of diminishing returns. Therefore, training does not necessarily take much longer for larger datasets.

The number of training iterations was fixed in advance for each experiment. On intermediate results on the XRD-CT dataset, we observed that most of the improvement occurred in the first 20% of training, in which all noise was removed, but the images were slightly blurred. Small details evolved in the remaining 80% of training time. In [75], it is reported that prolonged training can obtain some improvement in image metrics. In our experience, however, training for longer periods had no substantial influence on image quality. Because no ground truth data was available, changes in the output of the network were difficult to quantify. On the XRD-CT dataset, an additional training session was continued for ten times as many iterations ($300,000$). As training progressed, no reintroduction of noise into the reconstructed images was observed, and some small details were better resolved, indicating that overfitting was unlikely to occur.

4.3.2 Comparison to Total-Variation Minimization

In this section, Noise2Inverse is compared to a traditional iterative reconstruction approach. We implemented Total-Variation Minimization (TV-MIN) using the Chambolle-Pock algorithm [168]. For each dataset, we used 500 iterations and determined the optimal regularization parameter λ visually, as described in Appendix A.4. Reconstructions for several values of the regularization parameter are displayed in Figure A2. The TV-MIN reconstruction was performed on a single slice of the reconstruction and did not take into account additional space, time, or diffractogram information that was present in the static micro-tomography, dynamic micro-tomography, or the XRD-CT dataset, respectively.

An indication of the reconstruction time is given in the top right corner of each

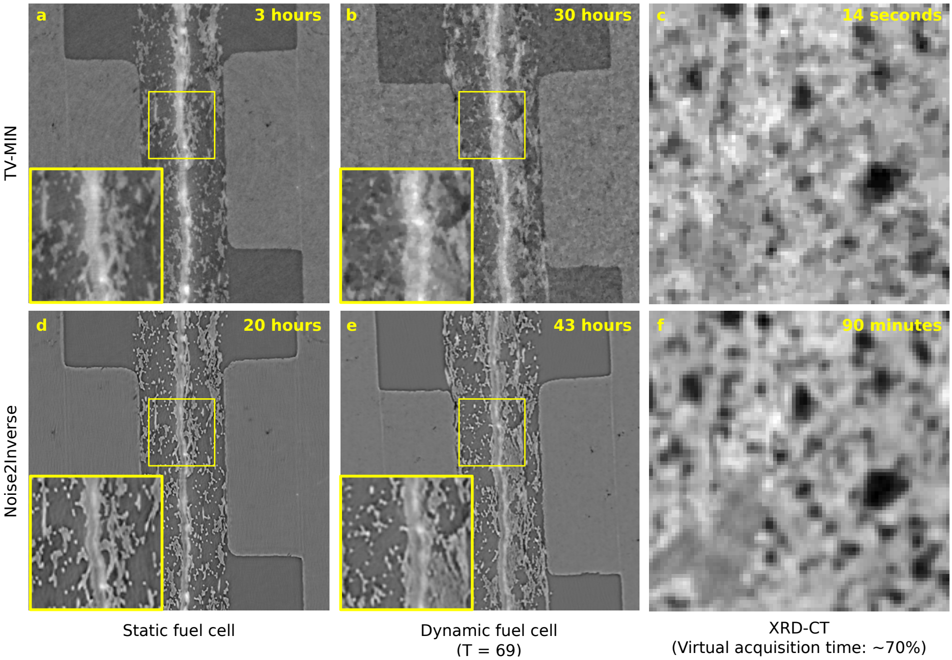


Figure 4.7: Comparison between Total-Variation Minimization reconstructions (**a** − **c**) and Noise2Inverse (**d** − **g**) reconstructions of the static and dynamic micro-tomography datasets of a fuel cell and X-ray diffraction tomography dataset of a ceramic. The displayed regions correspond to the reconstructions displayed in Figures 4.4, 4.5, and 4.6. Magnifications of the yellow regions of interest are displayed in the lower-left corner of the panels. In the top right corner of each panel, an indication of the reconstruction time of the full dataset is given using four Nvidia GeForce GTX 1080 Ti GPUs. The indicated time was realized for the Noise2Inverse reconstruction and estimated for the TV-MIN reconstruction.

panel in Figure 4.7. For the Noise2Inverse reconstruction, the realized reconstruction time is reported. For the TV-MIN reconstruction, an idealized reconstruction time is calculated by performing a single-slice reconstruction on a single GPU. To arrive at the reconstruction time for the full dataset, this number is multiplied by the total number of slices and divided by four to correct for the number of GPUs. We see that TV-MIN is faster in each case, but that the relative difference diminishes substantially for larger datasets.

The TV-MIN and Noise2Inverse reconstructions of the static fuel cell are visually comparable. In the other two cases, the TV-MIN reconstruction suffers from residual noise and stair-casing artifacts, which diminish the visibility of fine details. These results confirm the findings in [75], where it was found that Noise2Inverse could substantially outperform TV-MIN in terms of common image metrics. Image metrics cannot be computed in this case, because ground-truth images are not available.

4.4 Discussion and Conclusion

In this chapter, we have shown that Noise2Inverse can be extended in space, time, and spectrum-like domains. As a result, it can be effectively applied to tomographic imaging techniques in common use at synchrotron X-ray facilities. In the case of static and dynamic micro-tomography, we have shown that Noise2Inverse offers significant improvements in the quality of denoising over alternative reconstruction methods. In the case of X-ray diffraction tomography, we have shown that the acquisition time can be substantially reduced while maintaining image quality.

Using the Noise2Inverse conditions, we have discussed the considerations that enter into successful application of the method to several synchrotron imaging techniques. As noted in [75], the Noise2Inverse method benefits from high-resolution angular sampling, even at the cost of more measurement noise, but fares worse when angular sampling is sparse. We have shown that the effects of angular undersampling can be mitigated by exploiting interlaced angular sampling. We emphasize that Noise2Inverse does not assume a Gaussian noise model. The only assumption is that the noise does not result in a systematic upward or downward bias in the sinogram pixel intensities. Bias can result also from preprocessing, e.g., log correction and correcting for photon starvation, but for most realistic scenarios, the resulting bias is less than 1%, as discussed in Section 1.1.3.

Apart from measurement noise, vibrations and drifts of the sample constitute another substantial experimental uncertainty in dynamic micro-tomography. We note that the proposed approach does not deal with this issue directly: vibration of the sample can be visually identified between time steps in the Noise2Inverse reconstructions of the dynamic fuel cell dataset. It has been noted in previous work, however, that the improved visual reconstruction quality due to Noise2Inverse has enabled determining the correct center of rotation with greater precision [75]. Precise correction for sample vibration and drift as a preprocessing step is a topic for future work.

An open question is to what extent successful application to dynamic tomography depends on the similarity of the training time steps to later time steps. In our experiments, water was not present in the training data, but it was present in the rest of the sequence. We did not notice a deterioration of denoising accuracy for the water compared to other structures. This may be different in cases where the dynamics introduce more substantial changes in the sample structure. In general, we remark that the generalizability of the trained network mainly depends on the data that it is trained with and less so on whether it is trained in a supervised way or using Noise2Inverse.

In this chapter, we have demonstrated several strengths of the Noise2Inverse method. First of all, its versatility was shown on both large and small datasets that were obtained using different imaging techniques. In this work, it was applied to a dynamic tomography dataset hundreds of gigabytes in size, and an XRD-CT dataset of less than 10 MB. In addition, the method requires no CNN hyper-parameter tuning for training, which substantially simplifies its use in practice. We were able to achieve all results in this work using the same network architecture and the same CNN hyper-parameters, illustrating the sample-independence of the method. Finally, since the MS-D network architecture has a small number of parameters, it is unlikely to overfit to the noise. Therefore, prevention of overfitting through early stopping is not necessary, reducing the necessity for continuous observation of intermediate training results. All in all, the proposed method not only produces accurately denoised reconstructions, but can also be used in a “launch and forget” style, approaching the convenience of conventional reconstruction methods in a way that is uncommon for CNN-based methods. To scientists at synchrotron facilities, the sample-independence of the method could be appealing, as it opens up the possibility of blind application of the method to a variety of different samples.

5

Tomosipo: Flexible tomography in Python

“The difference between right and wrong often lies in less than five meters.”

“Het verschil tussen goed en fout ligt vaak in niet meer dan vijf meter.”

Johan Cruijff,
Vrij Nederland, 21 Dec 1974

Tomographic imaging enables the examination of the internal structure of an object. The object is typically placed between a source and detector, and its structure is reconstructed using projection images from a range of different positions. Collectively, the position information of the source, object, and detector determine the acquisition geometry.

Most common tomographic techniques rely on a selection of standard acquisition geometries, such as circular cone beam or single-axis parallel beam [31]. In recent years, several scientific and industrial applications have emerged whose needs are not met by the standard selection of paths. Such scientific applications include diffraction contrast tomography (DCT) [184] and X-ray scattering tensor tomography (XSTT) [93]. These techniques measure X-ray effects other than absorption, which necessarily give rise to more complex acquisition geometries. Complex geometries also arise in industrial applications like automotive and aerospace testing [52,

This chapter is based on:
A. A. Hendriksen, D. Schut, W. J. Palenstijn, N. Viganó, D. M. Kim Jisoo Pelt, T. van Leeuwen, and K. J. Batenburg. “Tomosipo: Fast, Flexible, and Convenient 3D Tomography for Complex Scanning Geometries in Python”. *Optics Express* (2021).

101], as objects may be too large to fit in conventional scanners. Instead, a robot arm moves the source and detector along an irregular path around the object.

Efficient reconstruction algorithms exist for many common acquisition geometries [31, 89]. Such filtered backprojection (FBP)-type algorithms are typically fast to compute [138], but require the source and detector to follow a regular path. Algorithms that permit flexible acquisition geometries, such as SIRT [60] and total variation minimization (TV-MIN) [168], typically follow an iterative reconstruction scheme. As iterative algorithms tend to be more computationally demanding than FBP-type algorithms, they benefit more from an efficient implementation.

Software packages for computing reconstructions can be roughly subdivided by their target audience. For application scientists in electron tomography [119] and synchrotron tomography [64, 130, 186, 188], software exists that provides pre-processing and reconstruction capabilities. For scientists developing new reconstruction algorithms, packages exist that integrate tomography in optimization methods [150, 193] and neural networks [175], or implement tomographic primitives on the graphics processing unit (GPU), such as the TIGRE and ASTRA Toolbox [1, 2, 20].

Existing tomography software is typically limited in its ability to represent, create, visualize, and reconstruct using complex acquisition geometries. Software for application scientists usually includes optimized reconstruction routines for a selection of acquisition geometries, but generally does not provide the flexibility to represent arbitrary acquisition geometries. Some software packages providing tomographic primitives, like the ASTRA Toolbox, can represent arbitrarily complex acquisition geometries, but do not provide effective tools to create them. In fact, the positions and orientations of the object and detector are usually computed using trigonometric formulas, requiring tedious and error-prone handwork [1]. In addition, limited facilities are included to visualize geometries, making it difficult to validate the computed geometry. Therefore, defining unconventional acquisition geometries requires extraordinary attentiveness. The lack of validation capabilities can also be problematic when processing data from advanced experiments, as it can be difficult or impossible to determine whether certain reconstruction artifacts are caused by an incorrect modeling of the acquisition geometry, or are due to other common sources of artifacts (e.g., sample motion, beam stability, etc). This may lead to sub-optimal reconstruction results and could prohibit further analysis of the data.

In this chapter, we introduce the `tomosipo` Python package[1], which is designed to alleviate the problems in defining complex acquisition geometries for tomography. Specifically, the package provides convenient primitives for the representation, creation, visualization, and reconstruction of complex acquisition geometries, as described below.

Representation and creation. Tomosipo allows the user to assemble increasingly complex acquisition geometries by composing geometric transforms and applying them to primitive acquisition geometries. Several standard geometric

[1] Tomósipo is pronounced with the stress on the second syllable.

transformations can be defined, such as rotation, translation, scaling, and reflection. Tomosipo's representation of the acquisition geometries is flexible. Therefore, the result of applying a geometric transform, e.g., rotation, to an acquisition geometry can be represented in tomosipo. In addition to flexible geometries, tomosipo provides convenience methods to create standard acquisition geometries, such as circular cone beam and single-axis parallel beam geometries.

Visualization. To aid in validation and communication, visualization of the resulting geometry is crucial. With tomosipo, the defined geometry can be viewed in a 3D environment or a Jupyter notebook [145], and saved to disk as a video or scalable vector graphic (SVG).

Reconstruction. Tomosipo provides a concise and efficient application programming interface (API) for computing reconstructions. Its design is similar to Matlab's Spot operators [21] and the computations are powered by the ASTRA Toolbox. In addition, tomosipo integrates with several packages for GPU computing, such as PyTorch [139] and CuPy, enabling the user to implement reconstruction algorithms without moving intermediate results to and from the GPU, yielding immediate speed benefits. These speed benefits are observed both in iterative and FBP-type reconstruction methods, as implemented in the separate `ts_algorithms` package[2].

This chapter provides an overview of the design of tomosipo and case studies of possible applications. First, the tomography problem is introduced in Section 5.1. In Section 5.2, key concepts of the package are described. In Section 5.3, these concepts are demonstrated on two simple absorption contrast tomography examples and two complex acquisition schemes exploiting X-ray diffraction and scattering. In Section 5.4, reconstructions are shown of experimental data using several algorithms. In Section 5.5, the use of tomosipo on the GPU is demonstrated and its speed is compared to existing reconstruction algorithms in the ASTRA Toolbox. We conclude with a discussion in Section 5.6.

5.1 Standard tomography problem

Common tomography setups expose a sample to a beam of high energy particles, e.g., photons, electrons, or neutrons, which are collected on a detector. Contrast in the measured projection images is generated by differences in attenuation, refraction index or scattering of the object (e.g. phase and diffraction, respectively), or the emission of secondary signals (e.g. X-ray fluorescence, Compton, Auger). Many of these problems can be modeled as a collection of line integrals through space where the ith measurement $y_i \in \mathbb{R}$ is obtained as a line integral

$$y_i = \int_{\mathbb{R}} x(\mathbf{s}_i + t\boldsymbol{\eta}_i) \, \mathrm{d}t \tag{5.1}$$

through a point $\mathbf{s}_i \in \mathbb{R}^3$ with direction $\boldsymbol{\eta}_i \in \mathbb{R}^3$. The canonical case is absorption contrast tomography, which we describe here.

[2] `https://github.com/ahendriksen/ts_algorithms`

In absorption contrast tomography, the reconstruction problem can be posed as a linear discrete inverse problem. Suppose measurements $\mathbf{y} \in \mathbb{R}^{N_\theta \times N_p^2}$ are acquired from N_θ positions using a square detector that is divided into N_p^2 pixels. Define the cubic reconstruction volume $\mathbf{x} \in \mathbb{R}^{N_v^3}$ on a voxel grid and let $\mathbf{A}$ denote the projection matrix such that $\mathbf{A}_{ij}$ describes the absorption by object voxel j of the ray to measurement i. The goal is to determine the value of $\mathbf{x}$ that gave rise to the measurement

$$\mathbf{A}\mathbf{x} = \mathbf{y}. \tag{5.2}$$

The computation of the linear operator $\mathbf{A}$ depends strongly on the geometry of the acquisition. This includes the direction of the rays, the position and orientation of the reconstruction volume, and the position and orientation of the detector.

5.2 Framework concepts

Three concepts are essential to the tomosipo package. These are geometries, geometric transformations, and the projection operator $\mathbf{A}$. Geometries represent the position of the source, sample, and detector at each time step. The sample's position and orientation is represented by a volume geometry, and the X-ray source and flat panel detector are represented by a projection geometry, which can model both point sources (cone beam geometry) and parallel box beams (parallel beam geometry). All geometries have two representations: a simple representation that defines a standard trajectory, and a flexible representation that permits arbitrary movement and orientation. Volume and projection geometries are discussed in Section 5.2.2.

Geometries can be manipulated using geometric transforms, as well as split and joined using subsampling and concatenation. In this way, complicated acquisition geometries can be assembled from simple geometries. This is described in Sections 5.2.3 and 5.2.5.

Together, a volume and projection geometry define the projection operator $\mathbf{A}$. In tomosipo, the computation using $\mathbf{A}$ is GPU-accelerated using the ASTRA Toolbox. Most tomosipo geometries have an ASTRA counterpart, except for the flexible volume geometry whose movement and orientation is compensated for by exploiting the flexibility of ASTRA's projection geometries. The creation of projection operators is discussed in the next section, and the integration with the ASTRA Toolbox and Python array libraries in Section 5.2.4. The main concepts of tomosipo and their relation to the ASTRA toolbox and the physical geometry are summarized in Figure 5.1.

5.2.1 Tomographic projection

In this section, we describe the creation and use of the projection operator $\mathbf{A}$. Tomosipo provides a convenient representation of the projection operator $\mathbf{A}$, offering

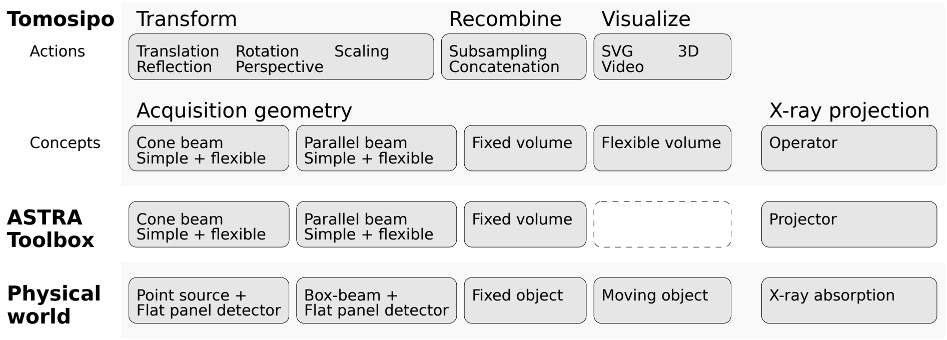


Figure 5.1: The relation between tomosipo, the ASTRA Toolbox, and the physical world. Tomosipo can be roughly divided in actions and concepts. The concepts describe the acquisition geometry and the X-ray projection and can be directly mapped onto ASTRA primitives, except for the flexible (moving) volume, which has no ASTRA counterpart. The actions provide the means to transform, recombine, and visualize tomosipo's geometry primitives.

an API that is similar to the opTomo Spot operator in the ASTRA Toolbox [21]. Given a volume geometry `vg` and projection geometry `pg`, the linear operator $\mathbf{A}$ from Equation (5.2) can be obtained as follows:

```
import tomosipo as ts
vg = ts.volume([...])          # Argument details are described
pg = ts.parallel([...])        # in next section
A = ts.operator(vg, pg)
```

The operator `A` is a stand-alone object. It has `domain_shape` and `range_shape` properties that facilitate the creation of data of the right shape in its mathematical domain and range, i.e., image space and sinogram space.

```
x = np.ones(A.domain_shape, dtype=np.float32)
```

It can be applied to data as follows:

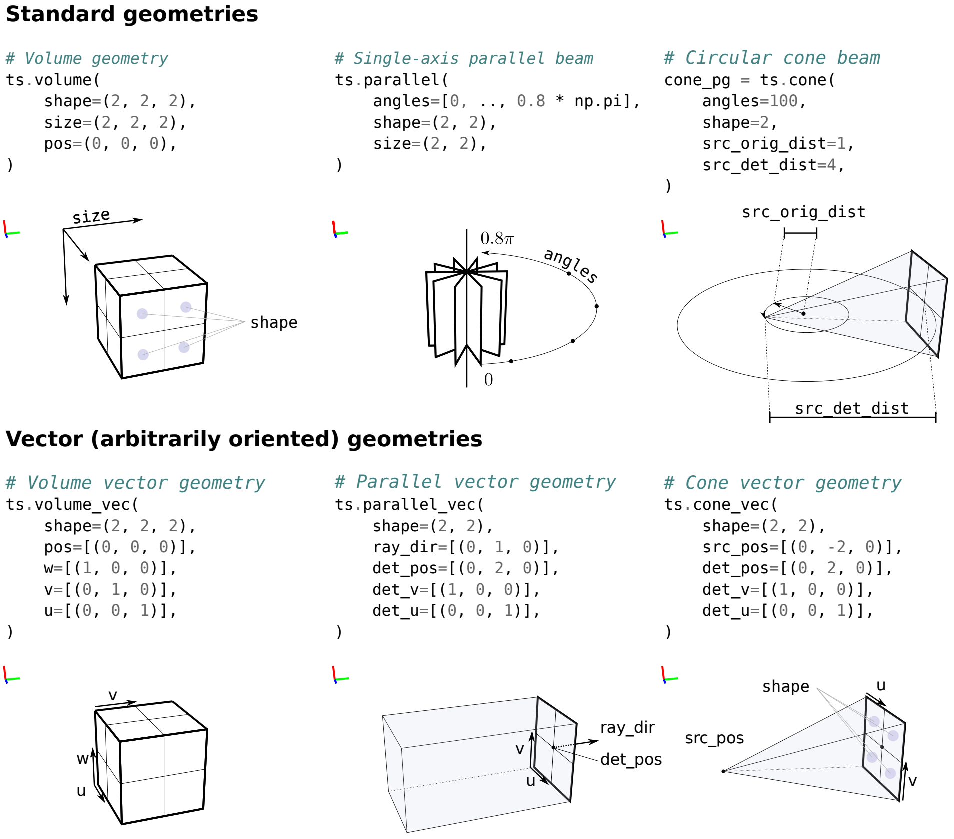
```
y = A(x)
backprojection = A.T(y)
```

The computation is performed on the GPU, and is handled by the ASTRA Toolbox.

The operator `A` can be used to solve the inverse problem posed in Equation 5.2. In the code below, this is demonstrated by computing a simple Landweber iteration [105] with step size `eta`.

```
x_rec = np.zeros(A.domain_shape, np.float32)
for i in range(num_iterations):
    x_rec = x_rec + eta * A.T(y - A(x_rec))
```

In the next section, we describe how to define the volume and projection geometries that are required to create a projection operator.



Standard geometries

```
# Volume geometry
ts.volume(
    shape=(2, 2, 2),
    size=(2, 2, 2),
    pos=(0, 0, 0),
)
```

```
# Single-axis parallel beam
ts.parallel(
    angles=[0, .., 0.8 * np.pi],
    shape=(2, 2),
    size=(2, 2),
)
```

```
# Circular cone beam
cone_pg = ts.cone(
    angles=100,
    shape=2,
    src_orig_dist=1,
    src_det_dist=4,
)
```

Vector (arbitrarily oriented) geometries

```
# Volume vector geometry
ts.volume_vec(
    shape=(2, 2, 2),
    pos=[(0, 0, 0)],
    w=[(1, 0, 0)],
    v=[(0, 1, 0)],
    u=[(0, 0, 1)],
)
```

```
# Parallel vector geometry
ts.parallel_vec(
    shape=(2, 2),
    ray_dir=[(0, 1, 0)],
    det_pos=[(0, 2, 0)],
    det_v=[(1, 0, 0)],
    det_u=[(0, 0, 1)],
)
```

```
# Cone vector geometry
ts.cone_vec(
    shape=(2, 2),
    src_pos=[(0, -2, 0)],
    det_pos=[(0, 2, 0)],
    det_v=[(1, 0, 0)],
    det_u=[(0, 0, 1)],
)
```

Figure 5.2: Creation of typical tomographic geometries. From left to right: a volume geometry, single-axis parallel beam geometry, and a circular cone beam geometry. Below, arbitrarily oriented vector geometries are shown. The parameters are specified using keyword-only arguments [179]. The `pos` parameter, for instance, determines the position of a volume, other parameters have accompanying labels in the diagrams.

5.2.2 Acquisition geometry primitives

Tomosipo provides three standard geometries: the fixed volume geometry, the single-axis parallel beam geometry, and the circular cone beam geometry. In addition, these geometries have a flexible counterpart that permits arbitrary orientation and movement. The flexible geometries are known as vector geometries, following the terminology of [1]. All geometry primitives are defined in the ASTRA Toolbox as well, except for the volume vector geometry that can represent an arbitrarily oriented moving reconstruction grid. The geometries are illustrated in Figure 5.2 with accompanying code.

In contrast to the standard projection geometries, whose movement is parameterized by the rotation angle, vector geometries move arbitrarily in time. We therefore refer to the state of the acquisition geometry at a specific time as a time step. Furthermore, geometries have a `num_steps` property that describes in how many time steps their movement is discretized.

Standard geometries

Volume geometry. A volume geometry describes the position and size of an axis-aligned voxel grid on which the object is reconstructed. A volume geometry can be created with `size`, `pos`, and `shape` parameters, which define its physical size, center position, and the number of voxels in each direction. By default, the volume is centered on the origin, and if the size is not specified, it is set to equal the shape, causing the voxel size to equal 1. Other parametrizations, such as in terms of the volume's extents, are described in the documentation.

Single-axis parallel beam. In the parallel beam geometry, X-rays run along parallel lines and are collected on a flat panel detector that rotates around a single axis on the origin. It can be created with `size`, `shape`, and `angles` parameters, which define the detector's physical size, the number of pixels in each dimension, and the rotation angles. If an integer argument is provided for `angles`, equispaced rotation angles in the interval $[0, \pi)$ are used. Otherwise, a provided array is interpreted as containing the rotation angles in radians.

Circular cone beam. Like the parallel beam geometry, the flat panel detector of a cone beam geometry rotates around an axis located on the origin and the `angles`, `shape`, and `size` parameters behave similarly. In contrast to the parallel beam geometry, the rays in a cone beam geometry are emitted from a point source, and the source-to-origin distance and source-to-detector distances can be specified using the `src_orig_dist` and `src_det_dist` parameters. Also, when `angles` is provided as an integer, a rotation is performed along a full arc $[0, 2\pi)$ as opposed to $[0, \pi)$.

Flexible vector geometries

Any geometry `g` can be converted to a vector geometry by calling `g.to_vec()`. Vector geometries can also be created directly as described below.

Volume vector geometry. In contrast to a volume geometry, which is static, a volume vector geometry may move over time and the reconstruction grid may be arbitrarily oriented. It can be created by providing the shape of the voxel grid and 3 vectors describing the local frame of reference of the grid at each point in time. In practice, a vector volume geometry is easier to obtain by applying a geometric transformation to a standard volume geometry.

The ASTRA Toolbox, tomosipo's computational back end, does not support non-axis-aligned volume geometries. Internally, tomosipo aligns the volume to the origin and moves the projection geometry with it. The transformed geometries are handed to ASTRA, causing the projection operation to be performed in the frame of reference of the object.

Arbitrarily oriented parallel beam. In a parallel vector geometry, the detector can be arbitrarily oriented and positioned. In addition, the direction of the incoming rays can be adjusted to a direction that is not necessarily orthogonal to the detector plane. It can be created by specifying a fixed detector shape and varying ray directions, detector positions, and detector orientations at each time step. The orientation is determined by parameters `det_u` and `det_v` that specify the vector from detector pixel $(0, 0)$ to $(0, 1)$ and $(0, 0)$ to $(1, 0)$, respectively. An example is the dual-axis parallel beam geometry, which is common in electron tomography [128].

Arbitrarily oriented cone beam. In a cone vector geometry, the detector can be arbitrarily oriented and the source can be placed in an arbitrary location. For instance, this geometry can represent a helical cone beam acquisition, as we show in Section 5.3.1. It can be created like the parallel vector geometry: instead of a ray direction, however, a source position must be provided for each time step. In the next section, we describe in more detail how vector geometries can be obtained as transformations of simple geometries.

5.2.3 Geometric transforms

Tomosipo defines geometric transforms that can rotate, translate, scale, and reflect the previously introduced geometries. In addition, the package provides a perspective transform to switch between different frames of reference. The transforms are stand-alone objects instead of functions that act on geometries directly. We first discuss the internal representation of the transforms and then we introduce the built-in functions to create transforms.

Representation. Internally, homogeneous coordinates [154] are used so that a 4×4 matrix $\mathbf{M}$ describes a single time step of a transformation. An orientation vector $\mathbf{v} = (v_1, v_2, v_3)$ is represented in homogeneous coordinates by $\mathbf{v} = (v_1, v_2, v_3, 0)$, whereas a position $\mathbf{p} = (p_1, p_2, p_3)$ is represented by $\mathbf{p} = (p_1, p_2, p_3, 1)$. This way, application of a geometric transform — notably translation — to points and vectors can be performed by matrix multiplication. That is, in homogeneous coordinates, the transformed vector equals $\mathbf{Mv}$ and the transformed point equals $\mathbf{Mp}$. In code, a vector and point in Euclidean coordinates are transformed as follows

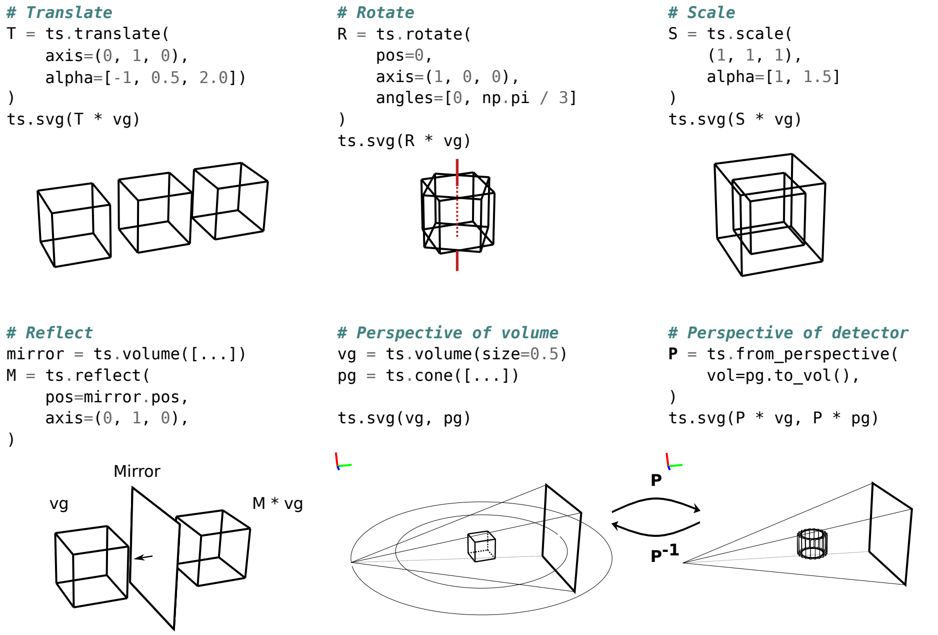
```
transformed_v = T.transform_vec(v)
transformed_p = T.transform_point(p)
```

Application of a transform to a geometry is expressed in code as

```
transformed_vg = T * vg
```

In the internal representation, the composition of two transforms is also computed by matrix multiplication. The matrix representation of the composition $T = T_1 \circ T_2$ of two transforms T_1, T_2 represented by matrices $\mathbf{M}_1, \mathbf{M}_2$ is equal to the matrix product of the matrices, i.e., $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$. In code, this is expressed as

```
T = T1 * T2
```



```
# Translate
T = ts.translate(
    axis=(0, 1, 0),
    alpha=[-1, 0.5, 2.0])
)
ts.svg(T * vg)
```

```
# Rotate
R = ts.rotate(
    pos=0,
    axis=(1, 0, 0),
    angles=[0, np.pi / 3]
)
ts.svg(R * vg)
```

```
# Scale
S = ts.scale(
    (1, 1, 1),
    alpha=[1, 1.5]
)
ts.svg(S * vg)
```

```
# Reflect
mirror = ts.volume([...])
M = ts.reflect(
    pos=mirror.pos,
    axis=(0, 1, 0),
)
```

```
# Perspective of volume
vg = ts.volume(size=0.5)
pg = ts.cone([...])

ts.svg(vg, pg)
```

```
# Perspective of detector
P = ts.from_perspective(
    vol=pg.to_vol(),
)
ts.svg(P * vg, P * pg)
```

Figure 5.3: Overview of geometric transforms in tomosipo. From left to right, translation, rotation, scaling, and reflection. In the two panes in the bottom right, a typical cone beam acquisition is shown from two perspectives: a static volume with the source and detector rotating around it and a static source and detector with a volume rotating in between. A perspective transform $\boldsymbol{P}$ allows switching between the two frames of reference. The vector illustrations are created using the `ts.svg()` function.

Composition of transforms is demonstrated in Section 5.3.1, where a helical cone beam geometry is created.

Rigid and scaling transforms. Tomosipo provides functions to create a translation, rotation, scaling, or reflection transform. These are illustrated in Figure 5.3. A transform may change over time, i.e., at each time step it can define a different geometric transformation. The functions that create the transforms are designed to facilitate defining transforms that vary over time.

A translation transform is parameterized by an `axis` and an array `alpha`. The displacement vector at time step `i` is defined by `alpha[i] * axis`.

A rotation transform is created using the axis angle representation. The `axis`, `pos`, and `angles` parameters describe the orientation and location of the rotation axis, as well as the angle of rotation. Each of these parameters may be provided as an array to define the rotation at multiple time steps. The angles are expressed in radians and the direction of rotation is right-handed by default.

A scaling transform describes a scaling operation centered on a position. The scaling is not necessarily isotropic: some directions can be scaled more than others. An `alpha` parameter can be used to modulate the scaling at each time step.

A reflection transform describes a reflection in a plane that is parameterized by a position `pos` and a normal vector `axis`. Both can be specified as an array, defining a reflection in a moving plane at several time steps.

Perspective. The `ts.from_perspective` function creates a perspective transform. This function takes a volume and returns the transform that moves the volume back to the origin and rotates it back into a single axis-aligned orientation. All projection geometries have a `to_vol()` method that describes the frame of reference of the detector at each time step. This makes it easy to create a transform that converts to the detector's frame of reference. In the case of a circular cone beam trajectory, for example, the source and detector rotate around the volume, from the volume's perspective. From the perspective of the detector, on the other hand, the volume rotates. This change in perspective is illustrated in the last two panes of Figure 5.3. Both perspectives yield the same projection operator $\mathbf{A}$.

5.2.4 Interoperability and GPU-acceleration

In this section, we discuss tomosipo's interoperability with NumPy arrays [65] and GPU-accelerated Python packages. In addition, we discuss the performance benefits of using GPU-accelerated arrays and also some trade-offs in favor of CPU arrays.

Projection operations are calculated using the ASTRA Toolbox. We have extended the ASTRA Toolbox API to enable direct operation on NumPy arrays. Before any ASTRA operation, the input arrays are automatically linked to the ASTRA runtime, and unlinked afterwards. This represents a substantial ergonomic improvement over the existing API. Apart from NumPy arrays, array types from other Python packages can also be linked. Out of the box, tomosipo interoperates with PyTorch and CuPy [136, 139]. More integrations can be added though an API, which can be used in the future to add interoperability with a variety of array libraries through the currently developing Python array API standard[3].

Integration with GPU array libraries can enable substantial performance improvements. In the snippet below, a NumPy array and a PyTorch array are forward projected. The NumPy array is located in RAM attached to the CPU, and the PyTorch array is located on the GPU.

```
y_numpy = A(np.ones(A.domain_shape, dtype=[...]))
y_torch = A(torch.ones(A.domain_shape, device="cuda"))
```

On line 1, the data is first moved to the GPU, the forward projection is calculated, and the data is moved back to the CPU. On line 2, no data movement takes place: the forward projection is calculated on the GPU. In iterative algorithms, where the forward and backprojection are repeatedly executed substeps of the algorithm, the latency imposed by CPU-GPU communication can dominate the computation time, as we demonstrate in Section 5.5. Note that PyTorch arrays can also be created on the CPU. In that case, the computation of the forward projection goes through exactly the same steps as a NumPy array would.

[3] `https://data-apis.org/array-api/latest/purpose_and_scope.html`

There are cases where it is beneficial to keep data on CPU. When data is too big to fit in GPU memory, the ASTRA Toolbox automatically splits data residing on CPU and performs the computation on the GPU in a streaming fashion. In this case, the user does not have to split up the data manually. When multiple GPUs are present on the system, they can be used automatically. In the code below, the ASTRA Toolbox is instructed to use four GPUs on line 1. The computation of the forward projection on line 2 is distributed over the four GPUs.

```
astra.set_gpu_index([0, 1, 2, 3])
y_numpy = A(np.ones(A.domain_shape, dtype=[...]))
```

5.2.5 Splitting and joining geometries

The ability to split and join geometries in tomosipo's API allows users to customize their design easily. Tomosipo allows subsampling a geometry to obtain a subgeometry. In addition, it allows joining the time steps of sequences of geometries into a single geometry. First, we demonstrate subsampling of projection geometries. Subsampling of volume geometries and geometric transforms works similarly and is described in the documentation. A projection geometry can be subsampled to obtain a geometry describing a subset of the detector surface. In the code below, the detector surface is cropped, removing a border of 100 pixels from each side. On the next line, the detector surface is subsampled, selecting every other row and column of pixels. Subsampling induces a slight shift in the detector's center, which is taken into account and described in detail in the documentation.

```
pg_cropped = pg[:, 100:-100, 100:-100]
pg_subsampled = pg[:, ::2, ::2]
```

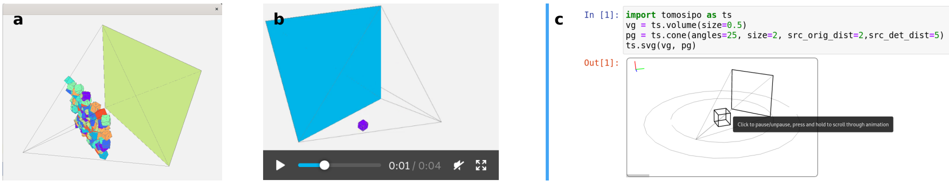
Subsampling the angular dimension is also possible. In this dimension, subsampling supports both slicing as well as Boolean masks [65]. In the code below, the angular direction is subsampled, obtaining a geometry that contains every other projection angle. In the line below, angles are selected when a `condition` array equals `True`.

```
pg_even_angles = pg[::2]
pg_boolean = pg[condition == True]
```

In Section 5.3.3, Boolean masking is demonstrated in the case study of X-ray diffraction tomography, where diffraction occurs in a subset of projection angles.

In addition to indexing, tomosipo also includes functionality to concatenate geometries and transforms. The concatenation of multiple projection geometries combines their time steps into a single geometry. In the code below, two projection geometries are combined. In the next line, a rotation `R` is repeatedly composed with different translations `T1, T2, T3`.

```
pg_combined = ts.concatenate([pg1, pg2])
TR_combined = ts.concatenate([T1 * R, T2 * R, T3 * R])
```


a
b
0:01 / 0:04
c
In [1]:
Out[1]:
Click to pause/unpause, press and hold to scroll through animation

```
import tomosipo as ts
vg = ts.volume(size=0.5)
pg = ts.cone(angles=25, size=2, src_orig_dist=2,src_det_dist=5)
ts.svg(vg, pg)
```

```
anim = animate(*vg, pg)
anim.window()
```

```
anim = animate(vg, pg)
anim.save("video.mp4")
```

```
svg = ts.svg(vg, pg)
svg.save("fig.svg")
```

Figure 5.4: Visualization options in tomosipo: (**a**) interactive 3D environment, (**b**) video, (**c**) interactive animation in a Jupyter Notebook. A single-particle Cryo-EM setup [17] is shown in panes (**a**) and (**b**), and a circular cone beam acquisition is shown in pane (**c**). Code snippets demonstrate how visualizations are created.

The concatenation of transforms is demonstrated in the case study of X-ray scattering tensor tomography in Section 5.3.4, where it is used to define a repeated rotation at several tilt angles.

5.2.6 Visualization

Tomosipo provides extensive support for visualizing geometries. Animations can be saved as a video or as a scalable vector graphic (SVG). In addition, geometries can be investigated in a 3D-accelerated environment on the desktop, allowing the user to zoom, pan, and rotate the view. Finally, an interactive SVG animation can be shown in an online Jupyter notebook, allowing for quick inspection of intermediate results. These options are illustrated in Figure 5.4. All other illustrations in this chapter have been generated using tomosipo. They were saved in the SVG format and extended using Inkscape.

5.3 Case studies

In this section, the concepts developed in the previous section are put into practice. We describe two simple examples and two complex acquisition schemes that are in use at synchrotron tomography beamlines. The first example demonstrates how geometric transforms can be composed to create a helical cone beam geometry. The second example models single-axis parallel beam tomography with a non-standard center of rotation in the frame of reference of the laboratory. In the first case study, we describe X-ray diffraction contrast tomography, which demonstrates the use of reflection and subsampling using a Boolean mask. In the second case study, we describe X-ray scattering tensor tomography, which demonstrates the use of concatenation. The case studies demonstrate that X-ray diffraction and scattering can be modeled using tomosipo's projection operators.

```
t = np.linspace(-1, 1, 100)   # Time t = -1.0, -.98, ..., 1
s = 2 * np.pi * t             # Angle
radius = 2                    # Radius of helix
h = 1.0                       # Vertical "speed"

vg = ts.volume()
pg = ts.cone(src_orig_dist=radius, src_det_dist=2 * radius)

R = ts.rotate(pos=0, axis=(1, 0, 0), angles=s)
T = ts.translate(axis=(1, 0, 0), alpha = h * s / (2 * np.pi))
H = T * R

ts.svg(vg, H * pg.to_vec())
```

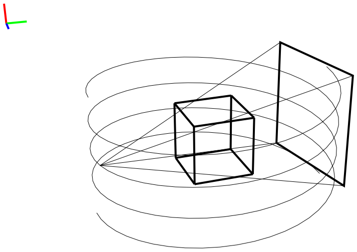


Figure 5.5: A helical cone beam geometry can be obtained as a composition of translation and rotation. The volume and cone beam geometries are defined to be non-moving. At each time step, the helical transform H applies a rotation R and then a translation T to the cone beam geometry.

5.3.1 Basic example: Helical cone beam geometry

As a demonstration of the composition of two primitive transforms, we define the helical cone beam geometry [88]. Here, the source and detector follow a helical path around the object. Using the notation of [88], we describe the helical geometry as a composition of translation and rotation in Figure 5.5. First, a static volume and a static cone beam geometry are defined. Next, a rotation R and translation T are defined, which rotate around and translate along the z-axis. The helical transform H is defined such that it applies the rotation R_i and then a translation T_i at time step i. When it is applied to the cone beam geometry, the resulting trajectory of the source and detector is helical. We note that the helical trajectory could have been obtained as a translation of a non-static cone beam geometry, effectively hiding the rotation in the cone beam geometry.

5.3.2 Basic example: Parallel beam in the lab frame

Acquisition using the single-axis parallel beam geometry is common at synchrotron beam lines. The detector is often positioned at a fixed location and the sample is mounted on a movable rotation stage. Typically, it is assumed that the center of rotation and the center of the detector coincide. In many cases in practice, however, it is difficult to achieve this with the described setup. Therefore, the offset between the center of rotation and the center of the detector has to be taken into account in order to achieve an accurate reconstruction. This is possible in tomosipo by positioning the detector, volume, and rotation axis independently from each other.

In Figure 5.6, the acquisition geometry is defined in the frame of reference of the laboratory. First, a static detector is translated from the origin to its final position by a transform `T`. Next, a static volume geometry is created at the initial position of the sample. A rotation is defined with a specific position of the rotation axis. Finally, the rotation is applied to the static volume, obtaining a rotating volume whose center rotates around the rotation axis.

There is an advantage to this formulation. In existing tomography packages, the position of the volume is commonly chosen to coincide with the rotation axis. However, this causes the reconstructed images to be translated when a different

```
# Static detector at custom position
T = ts.translate(det_pos)
static_pg = ts.parallel(angles=1, shape=det_shape)
pg = T * pg_static.to_vec()

# Static volume at custom position
vg_static = ts.volume(pos=vol_pos, shape=vol_shape)

# Rotate the volume
R = ts.rotate(pos=rot_axis_pos, axis=z_axis, angles=angles)
vg = R * vg_static.to_vec()

A = ts.operator(vg, pg)
```

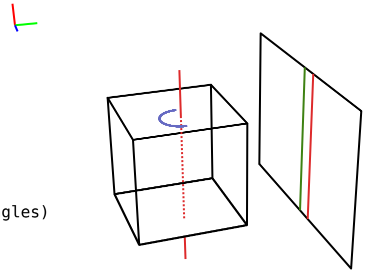


Figure 5.6: A single-axis parallel beam acquisition with a custom center of rotation. An object, whose changing position during rotation is indicated in blue, is rotated around a non-standard axis of rotation (in red). The center of the detector is indicated in green.

center of rotation is provided. This can cause problems when the determination of the correct center of rotation is based on the reconstructed images. In contrast, the proposed formulation opens up the possibility of determining the correct center of rotation by maximizing the auto-correlation in the reconstruction at several values of the center of rotation.

5.3.3 Complex case study: Diffraction contrast tomography

X-ray diffraction contrast tomography (DCT) [184] is an imaging technique used to investigate the internal structure of poly-crystalline materials. The crystal lattice is divided into grains, homogeneous regions where the lattice has a similar orientation. The orientation, size, shape, and arrangement of individual grains strongly influence macroscopic properties of the material. Therefore, mapping the orientation of grains is important to characterize a poly-crystalline material [124]. Here, we take as an example the three-dimensional DCT acquisition geometry as described in [184] to demonstrate specific features of tomosipo.

The goal of DCT is to reconstruct a vector field representing the intra-granular orientation of the crystal lattice. This is achieved by discretizing the orientation space on a regular grid that can be represented by unit vectors $\hat{\mathbf{o}}_1, \ldots, \hat{\mathbf{o}}_{N_o}$. For each orientation $\hat{\mathbf{o}}_k$, a scalar field, i.e., a volume, is reconstructed that represents the diffraction “intensity” at that orientation. A variational reconstruction algorithm ensures that neighboring voxels have similar orientations. A crystal lattice reflects an incoming X-ray beam at specific incidence directions, characterized by the so-called Bragg angles. When the diffraction geometry of the material under investigation is known beforehand, the intra-granular orientation of the crystal lattice can be recovered from those projection images at which Bragg diffraction is expected to occur.

As shown in Figure 5.7, the acquisition uses a monochromatic parallel box beam and the diffracted signal of the sample is measured on a flat-panel detector. As the sample is rotated, the reflection of the incoming beam in a voxel with local orientation $\hat{\mathbf{o}}$ forms a figure of eight on the detector. Bragg diffraction only

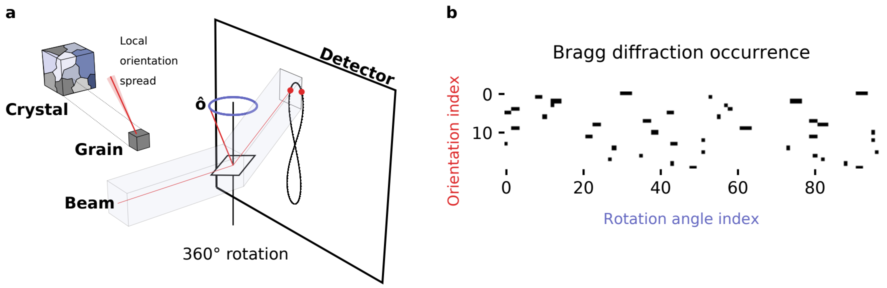


Figure 5.7: (**a**) In X-ray diffraction contrast tomography, a crystal sample is illuminated by a monochromatic X-ray box beam. The crystal sample is divided into grains, which have a minimal spread in local orientation. As the sample is rotated, the incoming beam is diffracted when its incident angle with the local lattice plane equals the Bragg angle. The intersection points of the reflected beam with the detector form a figure of eight, on which Bragg diffraction occurs only twice (marked in red) as the sample is rotated. (**b**) For a random sample of 20 orientations, the occurrence of Bragg diffraction at a rotation angle is indicated in black. Here, diffraction occurs in just 3.3% of the orientation-rotation combinations.

occurs in the instances where the beam and local lattice are in Bragg condition. Occurrence of the Bragg condition is relatively rare, as shown in Figure 5.7 (**b**). Both the reflection and its intermittent nature can be modeled in tomosipo.

Bragg diffraction (reflection). The diffraction, i.e., reflection, of an incident X-ray beam can be represented in tomosipo. As shown in Figure 5.7, a parallel bundle of rays remains parallel after it has been reflected. Therefore, the measurement of a diffracted parallel beam can be modeled using a standard parallel-beam geometry with altered ray direction.

The code below models the reflection of the incoming beam by a rotating crystal lattice. The orientation of the lattice is represented by a plane normal vector. First, the plane normal of the crystal lattice is rotated. Then, a reflection M is created in the rotating plane normal. The position of the reflection is arbitrary, as it is used to transform the direction of the beam and not its location. Finally, a static parallel beam geometry is modified such that the ray direction corresponds to that of the beam reflected in the rotating plane normal. The position and orientation of the detector remain static.

```
# Rotation of the rotation stage
R = ts.rotate(pos=0, axis=(1, 0, 0), angles=rot_angles)

def diffracted_pg(pg_static, plane_normal, R):
    rotated_plane_normal = R.transform_vec(plane_normal)
    M = ts.reflect(pos=0, axis=rotated_plane_normal)
    return ts.parallel_vec(
        shape=pg_static.det_shape,
        ray_dir=M.transform_vec(pg_static.ray_dir),
        det_pos=pg_static.det_pos,
        det_v=pg_static.det_v,
        det_u=pg_static.det_u,
    )
```

Bragg condition (Boolean masking). The occurrence of Bragg diffraction can be considered as a Boolean mask, an example of which is shown in Figure 5.7 (**b**). It is computed in the code below. First, the plane normal is rotated. Then, the Bragg condition is determined at each rotation angle.

```
bragg_mask = np.empty((num_orientations, num_angles), dtype=bool)

for i in range(num_orientations):
    rotated_normal = R.transform_vec(plane_normals[i])
    for j in range(num_angles):
        bragg_mask[i, j] = in_bragg_condition(
            rotated_normal[j], incoming_ray_dir, bragg_angle
        )
```

The created Boolean mask is used to select a subset of each projection geometry. For each orientation, the code below creates an operator that computes the forward projection only at rotation angles where Bragg diffraction occurs.

```
vg = R * ts.volume(shape=100).to_vec()
diffracted_pgs = [
    diffracted_pg(pg_static, normal, R) for normal in plane_normals
]
# Compute an operator per orientation
operators = [
    ts.operator(vg[bragg_mask[i]], diffracted_pgs[i][bragg_mask[i]])
    for i in range(num_orientations)
]
```

Multi-orientation tomography (sums of masked operators). The forward projection computes the diffraction pattern of $\mathbf{x} \in \mathbb{R}^{N_o \times N_v^3}$, representing all discretized plane orientations at N_v^3 locations, onto $\mathbf{y} \in \mathbb{R}^{N_\theta \times N_p^2}$, representing the N_p^2 pixel intensities at N_θ rotation angles. The operation is a linear combination of the masked operators defined above.

```
x = np.zeros((num_orientations, *vg.shape), dtype=np.float32)

def fp(x):
    y = np.zeros((N_p, N_angles, N_p), dtype=np.float32)
    for x_oriented, A, mask in zip(x, operators, bragg_mask):
        y[:, mask] += A(x_oriented)
    return y

y = fp(x)
```

In the interest of space, the backprojection operation is omitted. The full code listing can be found in the Supplemental materials of [74]. Implementing the variational reconstruction algorithm described in [184] is outside of the scope of this manuscript. We have shown how the DCT geometry can be succinctly expressed using tomosipo's rotation and reflection transformations. In addition, we have used subsampling with a Boolean mask to limit the forward projection to the few instances where Bragg diffraction occurs.

5.3.4 Complex case study: X-ray scattering tensor tomography

X-ray scattering tensor tomography (XSTT) is an imaging technique used to investigate materials with micro- and nano-scale structures over an orders of magnitude larger volumetric field of view, compared to conventional tomographic modalities [110, 117]. Here, we take the XSTT acquisition geometry that is described in [93] as an example to demonstrate specific features of tomosipo.

The goal of XSTT is to reconstruct a vector field representing the directional scattering intensity of a sample. This is achieved by reconstructing $N_{\hat{\mathbf{s}}} \geq 6$ scalar fields that represent the squared scattering coefficient along unit vector $\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_{N_{\hat{\mathbf{s}}}}$ at each voxel. After reconstruction, the directional scattering intensities are fine-tuned using per-voxel PCA (principal component analysis) [187]. XSTT has various biological and industrial applications [93]. As an example, the recovered local directional scattering intensities can be used to predict macroscopic properties of fibrous materials. These properties depend on the local fiber arrangement. Fibers scatter X-rays the least along their primary fiber orientation. Therefore, the local fiber orientation can be recovered from the shortest principal axis (smallest scattering magnitude) of the fitted scattering ellipsoid. The possibility to investigate these local structures over large enough volumes is valuable for the research and development of new materials.

We describe the acquisition process to obtain one of the scalar fields $\mathbf{x}_{\hat{\mathbf{s}}}$, representing the squared scattering coefficient along a unit vector $\hat{\mathbf{s}}$. First, we discuss the forward model at a single orientation of the sample, i.e., without any rotation or tilting. Let $\mathbf{x}_{\hat{\mathbf{s}}} \in \mathbb{R}^{N_v^3}$ represent the sample's squared scattering coefficient along a vector $\hat{\mathbf{s}}$. The sample is illuminated by a monochromatic parallel X-ray beam. Before they are measured on a detector, the X-rays travel through a panel that is etched with a periodic array of multi-circular gratings [93], generating a reference pattern. The panel is placed at a fixed propagation distance from the detector to maximize the visibility of the patterns. Different types of gratings require different

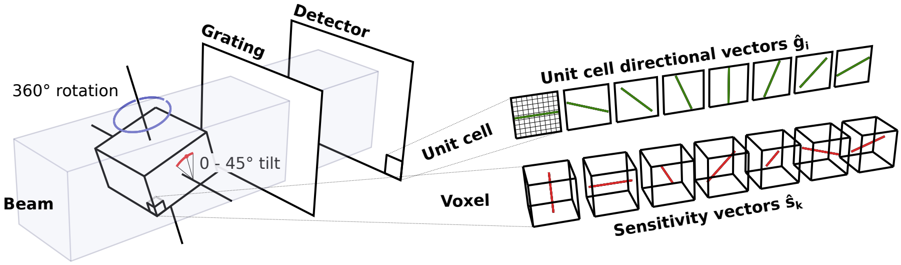


Figure 5.8: In X-ray scattering tensor tomography, a sample is illuminated by a box beam. The sample is repeatedly rotated at several tilt angles. The scattered signal passes through an array of gratings before being measured on a detector. The detector pixels are grouped into 9 × 9 pixel unit cells. In each unit cell, a directional intensity is measured along vectors $\hat{\mathbf{g}}_i$. In each voxel, scattering coefficients for multiple scattering sensitivity vectors $\hat{\mathbf{s}}_k$ are reconstructed.

acquisition geometries. The acquisition discussed in this case study is specifically geared to circular gratings.

With the use of circular gratings, the pixels of the detector are grouped into 9×9 pixel unit cells. In each unit cell, a 2D directional intensity is measured along multiple unit vectors $\hat{\mathbf{g}}_i$. The measured intensity $\mathbf{y}_i$ along the vector $\hat{\mathbf{g}}_i$ on the detector for a single beam direction $\hat{\mathbf{b}}$ is computed by scaling the forward projection with the scalar $\nu_{\hat{\mathbf{b}},\hat{\mathbf{s}},\hat{\mathbf{g}}_i}$ [117], defined by

$$\left(\left|\hat{\mathbf{b}} \times \hat{\mathbf{s}}\right| \langle \hat{\mathbf{s}}, \hat{\mathbf{g}}_i \rangle\right)^2 \mathbf{A}_{\hat{\mathbf{b}}}\, \mathbf{x}_{\hat{\mathbf{s}}} = \nu_{\hat{\mathbf{b}},\hat{\mathbf{s}},\hat{\mathbf{g}}_i} \mathbf{A}_{\hat{\mathbf{b}}}\, \mathbf{x}_{\hat{\mathbf{s}}} = \mathbf{y}_i. \tag{5.3}$$

The scaling is the same for each unit cell on the detector and varies as the sample (and thus $\hat{\mathbf{s}}$) is rotated.

Rotation and tilt (concatenation). When using circular gratings, the sample must be measured with multiple tilted rotation axes [93, 109]. In the XSTT acquisition described in [93], the sample stage is rotated, while the stage is tilted in steps. At each step, the stage makes a full rotation, as illustrated in Figure 5.8. Each step can be represented in tomosipo by composing a single tilt operation with a full rotation. In the code below, the full motion is computed by concatenating each step.

```
tilt = ts.rotate(pos=0, axis=(0, 0, 1), angles=tilt_angles)
rotate = ts.rotate(pos=0, axis=(1, 0, 0), angles=rotation_angles)
# For each tilt angle, perform a full rotation:
TR = ts.concatenate([tilt_single * rotate for tilt_single in tilt])
```

At each tilt and rotation angle, the scaling $\nu_{\hat{\mathbf{b}},\hat{\mathbf{s}},\hat{\mathbf{g}}_i}$ from Equation 5.3 is calculated as follows:

```
def calculate_nu(b, s, g, TR):
    nu = np.zeros(TR.num_steps)
    for j, s_rot in enumerate(TR.transform_vec(s)):
        nu[j] = (norm(np.cross(b, s_rot)) * np.dot(s_rot, g)) ** 2
    return nu
```

Because the calculation is performed in the lab frame, the vector $\hat{\mathbf{s}}$ is rotated rather than the beam direction $\hat{\mathbf{b}}$ or sensitivity vector $\hat{\mathbf{g}}$,

Scaled linear combinations. After $\nu \in \mathbb{R}^{N_{\hat{\mathbf{s}}} \times N_{\hat{\mathbf{g}}} \times N_{\text{tilt}} N_{\text{rot}}}$ is calculated for all values of $\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_{N_{\hat{\mathbf{s}}}}$, all $\hat{\mathbf{g}}_i$, and all tilts and rotations, then the full projection can be calculated. Here, the measurement along $\hat{\mathbf{g}}_i$ is the sum of the contributions of the $N_{\hat{\mathbf{s}}}$ scalar fields representing the scattering coefficients of the sample, as calculated below. In the interest of space, the backprojection operation is omitted. The full code listing can be found in the Supplemental materials of [74].

```
def fp(x, nu):
    y = torch.zeros(num_g, *A.range_shape, device=x.device)
    for k in range(num_s):
        for i in range(num_g):
            y[i] += nu[k, i][None, :, None] * A(x[k])

    return y
```

Data size. The reconstruction problem considered in [93] fits in memory on modern GPUs. The measured data consists of 46 tilt angles, 50 rotation angles, and 100 × 144 unit cells. Measuring along 8 $\hat{\mathbf{g}}_i$ vectors, the total number of measured unit cells equals $46 \times 50 \times 100 \times 144 \times 8 \approx 256 \times 10^6$, which requires approximately 1 GB when stored in 32 bit precision. The reconstruction volume consists of 44 × 71 × 71 voxels, repeated for each of $N_{\hat{\mathbf{s}}} = 7$ scattering directions. In total, it requires roughly 6 MB to store in 32 bit precision. The size of the scaling matrix ν is negligible in comparison. Modern data center GPUs range in memory size from 16 GB to 80 GB. Therefore, it is possible to run an iterative SIRT reconstruction of the full problem on GPU. Benchmarks comparing the performance on GPU versus CPU are provided in Section 5.5.

5.4 Experimental data

In this section, we show reconstructions of experimental data acquired using the standard circular cone beam and single-axis parallel beam trajectories, as well as a reconstruction of an X-ray scattering tensor tomography dataset. The reconstructions have been computed using the algorithms implemented in the separate ts_algorithms package.

Circular cone beam. A laboratory micro-CT dataset of a bell pepper was acquired at the FleX-ray laboratory[42] at the CWI, Amsterdam, The Netherlands. A polychromatic microfocus X-ray point source with tube voltage and power of 90 kV and 49.5 W was used. The data consisted of 3600 projection images of 1512 × 1912 pixels, acquired over a 360° rotation. A reconstruction was computed on a grid of 1512 × 1912 × 1912 voxels using FDK, a backprojection-type algorithm [53]. An axial slice of the reconstruction is shown in Figure 5.9 (**a**).

Single axis parallel beam. A 3D micro-tomography dataset of a fuel cell from the publicly available TomoBank [44] was used. This dataset (#81) was acquired at the TOMCAT beamline at the Swiss Light Source (SLS) at the Paul Scherrer Institut (PSI), Villigen, Switzerland [28]. The first 3600 projection images of 1100 × 1440 pixels were used to compute a reconstruction on an axial slice of 1400 × 1400 pixels. The reconstructions were computed using FBP (Ram-Lak filter), SIRT (200 iterations), and TV-MIN (500 iterations with $\lambda = 2 \times 10^{-7}$), as shown in Figure 5.9 ($\mathbf{b} - \mathbf{d}$).

X-ray scattering tensor tomography. The same validation sample was used as in a previous publication [93], which was also acquired at the TOMCAT beamline. It consisted of a $4 \times 4 \times 4\,\text{mm}^3$ plastic box containing three orthogonally oriented bundles of carbon fiber with a 12 µm diameter. The pixel and resulting unit cell size was $11 \times 11\,\mu\text{m}^2$ and $99 \times 99\,\mu\text{m}^2$, generating the dataset size described at the end of Section 5.3.4. An illustration of the validation sample and its reconstruction using tomosipo is shown in Figure 5.10. The reconstructions show the orientation of the fibers after post-processing using PCA and a similar thresholding strategy as in [93]. Thresholding causes noise in the background to be suppressed, as the X-ray scattering induced by plastic container is known to be negligible.

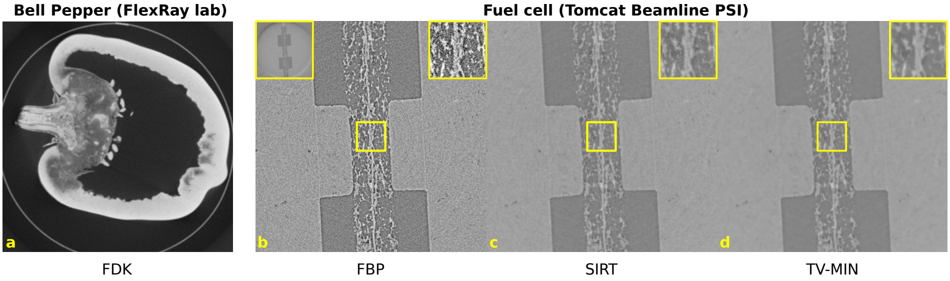


Figure 5.9: Reconstructions of experimental data acquired using laboratory micro-CT (**a**) and synchrotron micro-tomography (**b** − **d**). The yellow insets in the top-right corner show a magnified region of interest. The yellow inset in the top-left of pane (**b**) displays a full view, showing field-of-view artifacts due to the truncated projection images.

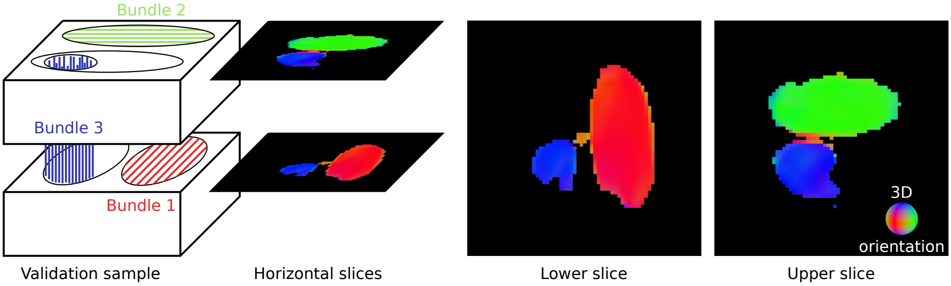


Figure 5.10: X-ray scattering tensor tomography reconstruction of a validation sample. The sample contains three orthogonally oriented carbon fiber bundles. A reconstruction of the orientation map is shown in two axial slices.

5.5 Benchmarks

In this section, we give a demonstration of the computational speed of tomosipo. First, we compare an implementation of SIRT in tomosipo to the built-in implementation in the ASTRA Toolbox. Using the tomosipo implementation, we also investigate the impact of storing intermediate data on the CPU rather than on the GPU. This comparison is run on the examples from Section 5.3 with data sizes that fit on a single GPU. We exclude DCT, as its reconstruction algorithm is out of the scope of this manuscript. We also benchmark a non-iterative algorithm on a circular cone beam dataset that does not fit on the GPU. Here, we compare the speed of the built-in FDK implementation of the ASTRA Toolbox to the FDK implementation in ts_algorithms, tomosipo's accompanying reconstruction algorithms package.

We describe the algorithms, data size, and benchmark methodology. The SIRT reconstructions were computed in 50 iterations. The implementation in tomosipo used PyTorch and the ASTRA implementation used the `SIRT3D_CUDA` algorithm. The FDK benchmark compared the FDK implementation provided

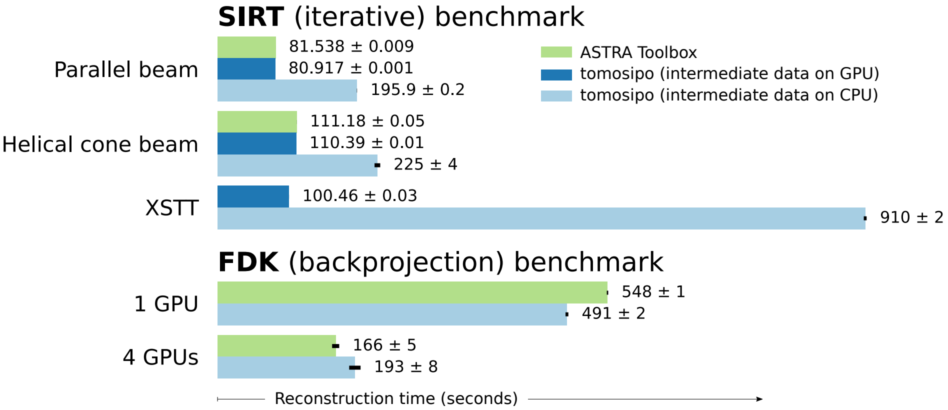


Figure 5.11: Comparison of reconstruction times using SIRT on a GPU-sized problem and using FDK on a lab-CT-sized problem. The SIRT implementations are compared on a parallel, helical and X-ray scattering tensor tomography (XSTT) acquisition geometry. The XSTT reconstruction cannot be implemented using the built-in ASTRA SIRT API. Because the FDK dataset is too large, intermediate data cannot be stored on the GPU, and the ASTRA implementation is compared to a tomosipo implementation that performs the filtering step on the CPU.

by ts_algorithms to ASTRA's built-in `accumulate_FDK` implementation. The dataset of the parallel beam and helical cone beam cases consisted of 768×768 pixel projection images acquired over 512 angles and was reconstructed on a 512^3 voxel volume. The sizes of the XSTT and circular cone beam dataset are described in Sections 5.3.4 and 5.4, respectively. The benchmarks were conducted on a dual-socket system containing 8-core Intel Xeon Silver 4110 CPUs at 2.10 GHz (Intel, Santa Clara, CA, USA) with 192 GB of RAM and four Nvidia GeForce GTX 1080 Ti GPUs (Nvidia, Santa Clara, CA, USA). Each benchmark was run once without measurement to minimize startup and caching effects. The mean and standard deviation of three trials are reported.

SIRT on GPU-sized problems. The results of the SIRT benchmark are shown in Figure 5.11. The ASTRA Toolbox and the Tomosipo implementation with intermediate data on the GPU are close in performance. They are are $2 - 9\times$ faster than the tomosipo implementation with intermediate data located on CPU memory. This indicates that CPU-GPU communication latency is non-negligible and that reconstruction algorithms benefit from being completely computed on the GPU. We note that in all three implementations the forward and backprojection are computed on the GPU using the ASTRA Toolbox. The native ASTRA SIRT implementation does not have an option to store intermediate data on CPU memory.

FDK on a lab-CT-sized problem. The FDK dataset is too big to fit in GPU memory. In tomosipo's implementation, the filtering step is performed on the CPU and the computation of the backprojection on chunks of projection data is distributed over multiple GPUs. The FDK implementation in the ASTRA Toolbox, on the other hand, first distributes chunks of projection data over available GPUs and performs the filtering and backprojection in a single step on each GPU.

Figure 5.11 shows the results of the FDK benchmark using one and four GPUs. Using one GPU, tomosipo's implementation of FDK is faster than ASTRA's. This can be attributed to fast filtering on the CPU, which is implemented using the Fast Fourier Transform provided by PyTorch and is approximately as fast as filtering on a single GPU. Using four GPUs, the run times of both implementations are reduced, but the ASTRA implementation comes out ahead. When four GPUs are available, the ASTRA implementation distributes the computation of the filter step over four GPUs, whereas the tomosipo implementation still computes the filtering step on the same amount of CPU cores.

The results show that a naive implementation of an iterative algorithm in tomosipo is not necessarily slower than a native implementation in the ASTRA Toolbox. In addition, the results illustrate the substantial negative impact that CPU-GPU communication has on reconstruction speed. Finally, the FDK results illustrate the benefits of interoperability with fast array libraries, but highlight the need for effective APIs to address multi-device streaming computation.

5.6 Discussion and conclusion

In short, tomosipo provides the expressive power to quickly and naturally define complex geometries, thereby unlocking the flexibility provided by the ASTRA Toolbox. We have demonstrated the ease of making common adjustments to an acquisition geometry, such as changing the center of rotation. In addition, the design and implementation of more complex geometries, such as the demonstrated X-ray diffraction and scattering setups, is made considerably easier by using tomosipo, especially compared to entering the formulas for all directional vectors manually. Reconstructions of real-world data from synchrotron and laboratory micro-CT sources are shown, computed using several common reconstruction algorithms. Finally, bechmarks demonstrate that the package enables the user to write fully GPU-accelerated reconstruction algorithms in Python whose speed is on par with native implementations. Because of tomosipo's interoperability with GPU-accelerated Python array libraries, intermediate results can remain on the GPU, avoiding the latency imposed by CPU-GPU communication.

The tomosipo package follows best practices. It has a comprehensive unit test suite, it is installable through the Anaconda package manager, it follows semantic versioning, it is developed in the open on GitHub, and it has extensive documentation.

Future developments are expected to go hand in hand with improvements in the ASTRA Toolbox. This includes support for curved detectors and more fine grained control of streams on the GPU, allowing for concurrency through pipelining. In addition, we intend to extend the interoperability of tomosipo's projection operator to more optimization packages. We note that the integration of tomosipo's projection operator in deep learning-based reconstruction methods using PyTorch is possible and is described in the documentation.

Compared to existing tomographic software packages, two features set tomosipo apart. First, the facilities to manipulate geometries significantly simplify defining complex acquisition geometries such as those in the described case studies. Although other packages including the ASTRA Toolbox and the Core Imaging Library (CIL) [82] can represent these acquisition geometries, they do not provide tools to define them. Specifically, the geometric transforms, subsampling, concatenation, and visualization features are not provided by the ASTRA Toolbox. Second, the extensible integration of tomosipo with GPU-accelerated Python array libraries provides two advantages. It enables the user to write custom reconstruction algorithms in Python that are comparable in computational efficiency to a native implementation. In addition, it enables integrating tomographic operators in deep learning-based reconstruction methods. This is technically possible using the ASTRA Toolbox, but the APIs that it exposes are designed to be wrapped by a user-friendly library, such as tomosipo.

We stress that tomosipo aims to be a building block in a larger system. Therefore, other software packages may be preferable for many purposes. Facilities for loading of various file formats, preprocessing of tomographic data, or post-processing of reconstructed images are present in TomoPy, Savu, and CIL [64, 82, 188]. Packages such as PyLops, CIL, and JUDI [82, 150, 193] provide building blocks and built-in optimization algorithms that enable rapid prototyping of variational reconstruction methods, among others. The reconstruction algorithms show-cased in this manuscript, on the other hand, are implemented in a separate package [70]. An advantage of the focused scope of tomosipo, is that it has only two required dependencies (NumPy and the ASTRA Toolbox), making it easy to install on various platforms, but contains several integrations with third-party packages, making it easy integrate into an existing system.

In summary, tomosipo provides scientists with an excellent tool to model and visualize complex tomographic acquisition geometries while maintaining and extending the fast reconstruction capabilities of the ASTRA Toolbox.

6 Conclusion and outlook

"When the press wants to burn me, I can have peace with that. But then I want to be burned for my own vision."

"Wanneer de pers me wil afbranden heb ik daar vrede mee, maar dan wil ik op mijn eigen visie afgebrand worden."

Johan Cruijff,
Voetbal International, 14 Oct 1989

The main goal of the research presented in this thesis was to develop practical deep learning techniques to improve tomographic reconstruction when acquiring a dataset of additional measurements is not feasible. To support this aim, a subordinate goal was to develop software that supports both the concise expression of complex tomographic acquisition geometries and the development of computationally efficient reconstruction algorithms. In this chapter, we summarize the contributions and limitations of this thesis and suggest directions for future research.

6.1 Contributions and limitations

The contributions of the first three chapters of this thesis can be categorized along two axes. The first is *what* factor that determines image quality is improved, i.e., resolution or noise. The second is *how* the problem of limited training data is circumvented. We propose two ways: the first is by changing the scanning protocol and the second is by changing the neural network training scheme. In Chapter 2, we propose to improve the resolution of reconstructed images by using a custom scanning protocol to create a dataset of low- and high-resolution images. In Chapters 3 and 4, we propose to remove noise from reconstructed images by training a neural network using a custom neural network training scheme.

The contributions of the final chapter cannot be categorized in this way. Instead, its contribution is a software package that makes it substantially easier to (1) compute reconstructions with a complex scanning protocol, and (2) incorporate tomographic operators in neural networks. In this way, it supports the aims of the previous chapters by enabling the custom scanning protocol and custom training scheme. This section is organized as follows. First, we describe the contributions in each chapter. Next, we describe remaining practical limitations.

In **Chapter 2**, we proposed a novel technique for improving the resolution of tomographic reconstructions. To enable application on single objects, a custom scanning procedure is developed that enables obtaining a low-resolution and a high-resolution reconstruction of a small region of the object. On this region, a data-efficient neural network is trained to transform low-resolution images to high-resolution images. The trained network can then be applied to the full low-resolution reconstruction. On both simulated and experimental data, the results demonstrate that the proposed technique is able to significantly improve the resolution of tomographic reconstructions of a single object.

The topic of **Chapter 3** was the removal of noise from reconstructed images. It consisted of an investigation of existing methods and the development of a custom training strategy. First, self-supervised deep learning methods for photographic image denoising were investigated. Such methods depend on the assumption that noise in one pixel is not correlated to noise in another pixel. An experiment was conducted in which a photographic image denoising method was applied to images with Gaussian noise and images with comparable tomographic noise. The results show that the method performs substantially worse on tomographic noise than on Gaussian noise. This failure can be explained by the fact that tomographic noise violates the no-correlation assumption.

Therefore, Noise2Inverse was developed, a method for training deep neural networks to denoise reconstructed images using only noisy training images. Starting from a single noisy tomographic acquisition, multiple reconstructions are computed from mututally non-overlapping subsamples of the projection images. Images from these separate noisy reconstruction volumes serve as input and target to train a neural network. The noise in each reconstructed volume is uncorrelated to noise in the other volumes, thereby enforcing non-correlation of the noise in the input and target images. We developed a Bayesian argument that shows that therefore such a training regime should result in a denoising neural network. Results on a diverse set of simulated and real-world data confirm that this is indeed the case: the denoising accuracy of the proposed method approaches that of a network trained using ground truth images. In addition, a comparison to state-of-the-art image denoising methods and conventional reconstruction methods, such as total-variation minimization was performed. Here, Noise2Inverse demonstrated an improvement in key image metrics, such as peak signal-to-noise ratio and structural similarity index.

In **Chapter 4**, we investigated the applicability of Noise2Inverse to tomographic imaging techniques in common use at synchrotron X-ray facilities. Specifically, we investigated applications to static and dynamic micro-tomography and X-ray diffraction tomography (XRD-CT). To obtain optimal results, we extended the training strategy in space (micro-tomography), time (dynamic micro-tomography), and spectrum-like domains (XRD-CT). In the case of static and dynamic micro-tomography, results show that Noise2Inverse offers significant improvements in the quality of denoising over alternative reconstruction methods. In the case of XRD-CT, results suggest that the acquisition time can be substantially reduced while maintaining image quality.

In **Chapter 5**, we presented the tomosipo software package. It supports the development of reconstruction algorithms for data from advanced experiments. Developing these algorithms can be slowed down by (1) the difficulty of defining the acquisition geometry, i.e., the trajectory of the source and detector, and (2) the inefficient interoperability of GPU-accelerated tomographic primitives with GPU-accelerated array libraries. This is exactly where tomosipo excels, as it allows the user to specify complex geometries in an intuitive and convenient framework. We have demonstrated the ease of making common adjustments to an acquisition geometry, e.g., changing the center of rotation, as well the ease of designing and implementing more complex geometries, e.g., X-ray diffraction and scattering setups. In addition, bechmarks demonstrate that tomosipo enables the user to overcome existing performance challenges in reconstruction algorithm development. The package enables the user to take full advantage of the GPU, achieving parity with existing optimized implementations and providing a 2–9$\times$ speed up compared to CPU-based implementations. Several common reconstruction algorithms are already implemented and demonstrated on real-world data from synchrotron and laboratory micro-CT sources.

The results in this thesis demonstrate that deep learning can be practically applied to improve resolution and noise in reconstructed images. Specifically, the developed techniques enable deep learning to be applied in situations where limited training data would previously have hampered its adoption. Some practical limitations still bear consideration though.

One challenge that becomes more prominent as a result of our work is the long training times of neural networks. As we show in Chapter 4 for instance, applying Noise2Inverse takes almost twice as long ($\sim$ 43 hours) as using a variational reconstruction algorithm ($\sim$ 30 hours) in the best case. The majority of this time is spent training the network and only a minor fraction of time is spent on applying the trained network to obtain the final result. Compared to other works that propose to train a network once and incorporate the trained network in a reconstruction algorithm, our methods are in the atypical position that network training can be part of the reconstruction algorithm itself. Therefore, a reduction in training times would be a major improvement in practice.

An issue that is not analyzed in detail is *generalization*, i.e., to which extent networks must be retrained when reconstructions of new structures are encountered. This issue is discussed in Chapter 4 in relation to scanning of additional objects under possibly different acquisition conditions. A more subtle issue arises in the super-resolution techniques of Chapter 2, where the local structure may vary throughout the object, contrary to the assumption that the structure in the region of interest is similar to the structure of the rest of the object. This may require attentiveness on the part of the user to identify changing structures and acquire a training dataset containing reconstructions of multiple regions of interest that more accurately reflect the heterogeneous structure of the object.

Another challenge is the calibration of the acquisition geometry, e.g., the center of rotation. When reconstructed images are extremely noisy, or blurred as a result of denoising, then artifacts resulting from improper calibration of the acquisition geometry are not immediately visible. In the results of Noise2Inverse, however, these artifacts are often noticeable. This could be too late, however: it may not feasible to repeat the computationally expensive neural network training with different calibration values for the acquisition geometry. Therefore, Noise2Inverse would benefit greatly from accurate and computationally efficient methods to detect subtle misconfigurations of the acquisition geometry.

Finally, there is the issue of completeness. We have proposed techniques to deal with insufficient detector resolution and measurement noise, but not angular undersampling. Measuring fewer projection images is one of the easiest ways to speed up tomographic acquisition and lower the total radiation dose. In addition, it requires fewer bits and bytes to store measured data. The data size may be a limitation in extremely fast experiments where the transmission of measurement data is bandwidth-limited [131].

6.2 Outlook

Several promising research directions are unfolding to improve image quality and reduce training time. In this section, we discuss these in turn. First, image quality can be improved by forcing the neural network output a more "likely" image, or output a distribution of several likely images. In addition, image quality can be substantially improved by proper calibration of the acquisition geometry. Finally, we discuss several approaches to reduce the training time of neural networks.

The use of the mean-square error objective during training can lead to blurred results. This is a consequence of the fact that the trained neural network approximates the conditional mean (Equation (1.15)). For instance, if an observed low-quality image is equally likely to have resulted from two different high-quality images, then traing will force a neural network to output the average of the two images [106]. In super-resolution of photographic images, methods have been developed to work around this issue using generative adversarial networks (GANs) [7, 58]. GAN-based approaches force the trained network to output an image that is likely to occur, rather than the average of likely images [84, 106]. In practice, however, this technique should be approached with some caution, as it can output a single very realistic looking result when the measurement supports multiple equally likely reconstructions. It could therefore lead a user to become overconfident in the results of a measurement.

To prevent overconfidence in a single result, multiple results can be sampled instead. Sampling approaches based on diffusion models have been demonstrated for improving resolution [160], denoising [91], and linear inverse problems such as deblurring [90], for instance. In these cases, output generation starts with Gaussian noise and repeatedly refines the noisy image using a neural network while noise of diminishing intensity is injected in the image. This process continues for up to a 100 steps [160], and under some regularity conditions the resulting image can be shown to be sampled from the posterior distribution [171]. In practice, however, applying a neural network a thousand times might be too computationally demanding.

The importance of accurate calibration of the acquisition geometry will increase as a result of faster and finer-scale acquisition and the demands of dynamic quantitative analysis. As discussed in Chapter 4, fast acquisition can cause vibrations in the measured object as a result of the high rotation speed. When tracking the internal structural dynamics of the object, it is essential to accurately overlay the 3D volumes at multiple time steps [115]. When vibration is an issue, the correct alignment of two time steps requires not just the correction of some rigid movement between the time steps, but also the correction of orientation changes between the projection images within a single time step. Iterative approaches that incorporate such calibration in the reconstruction, such as TIMBIR [3], have already been developed, but in terms of computational demands it may be desirable to develop preprocessing methods that do not require the simultaneous reconstruction of the object. Preprocessing-type methods may also be easier to combine with techniques such as Noise2Inverse and other deep learning-based methods.

Finally, to reduce the problem of long network training times, we discuss three approaches. These are based on (1) network modifications to speed up training (2) dataset modifications to improve generalization, and (3) transfer learning to jump-start network training. An instance of network modification to speed up training, is Noise2Filter [103], which uses a very small network that is adapted to tomography [144] to reduce training times to a minute or less. A drawback of these methods so far has been that the resulting image quality lags behind what can be achieved using convolutional neural networks. In Cryo-electron microscopy, a recent approach has been to train a single “super network” on a very large and diverse set of data that can ideally be applied to a variety of new datasets [18]. As this method gains more popularity, a more accurate assessment can be made whether it generalizes to unexpected datasets in practice. Finally, a hybrid between training a network from scratch and using a super network is transfer learning. Here, a network is first trained on a large and diverse dataset and the resulting trained network is retrained on new datasets for specific tasks. These techniques have been used in medical image classification [51, 63].

Bibliography

[1] W. van Aarle, W. J. Palenstijn, J. Cant, E. Janssens, F. Bleichrodt, A. Dabravolski, J. De Beenhouwer, K. Joost Batenburg, and J. Sijbers. "Fast and Flexible X-Ray Tomography Using the ASTRA Toolbox". *Optics Express* 24.22 (2016), p. 25129 (cit. on pp. 92, 96, 152).

[2] W. van Aarle, W. J. Palenstijn, J. De Beenhouwer, T. Altantzis, S. Bals, K. J. Batenburg, and J. Sijbers. "The ASTRA Toolbox: a Platform for Advanced Algorithm Development in Electron Tomography". *Ultramicroscopy* 157 (2015), pp. 35–47 (cit. on pp. 33, 59, 64, 92).

[3] K. Aditya Mohan, S. V. Venkatakrishnan, J. W. Gibbs, E. B. Gulsoy, X. Xiao, M. De Graef, P. W. Voorhees, and C. A. Bouman. "TIMBIR: a Method for Time-Space Reconstruction From Interlaced Views". *IEEE Transactions on Computational Imaging* 1.2 (2015), pp. 96–111 (cit. on pp. 81, 119).

[4] J. Adler and O. Öktem. "Deep Bayesian Inversion". *CoRR* (2018). arXiv: `1811.05910 [stat.ML]` (cit. on pp. 12, 54).

[5] J. Adler and O. Öktem. "Learned Primal-Dual Reconstruction". *IEEE Transactions on Medical Imaging* (2018), pp. 1–1 (cit. on pp. 11, 62, 74).

[6] A. Andersen. "Simultaneous Algebraic Reconstruction Technique (SART): a Superior Implementation of the ART Algorithm". *Ultrasonic Imaging* 6.1 (1984), pp. 81–94 (cit. on p. 10).

[7] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 214–223 (cit. on pp. 14, 118).

[8] S. R. Arridge. "Optical Tomography in Medical Imaging". *Inverse Problems* 15.2 (1999), R41–R93 (cit. on p. 21).

[9] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. "Solving Inverse Problems Using Data-Driven Models". *Acta Numerica* 28 (2019), pp. 1–174 (cit. on p. 12).

[10] G. Artioli, T. Cerulli, G. Cruciani, M. C. Dalconi, G. Ferrari, M. Parisatto, A. Rack, and R. Tucoulou. "X-Ray Diffraction Microtomography (XRD-CT), a Novel Tool for Non-Invasive Mapping of Phase Development in Cement Materials". *Analytical and Bioanalytical Chemistry* 397.6 (2010), pp. 2131–2136 (cit. on pp. 45, 74, 81).

[11] G. Ashiotis, A. Deschildre, Z. Nawaz, J. P. Wright, D. Karkoulis, F. E. Picca, and J. Kieffer. "The Fast Azimuthal Integration Python Library: pyFai". *Journal of Applied Crystallography* 48.2 (2015), pp. 510–519 (cit. on p. 152).

[12] S.-M. Bak, Z. Shadike, R. Lin, X. Yu, and X.-Q. Yang. “In Situ/operando Synchrotron-Based X-Ray Techniques for Lithium-Ion Battery Research”. *NPG Asia Materials* 10.7 (2018), pp. 563–580 (cit. on p. 4).

[13] K. J. Batenburg and J. Sijbers. “DART: a Practical Reconstruction Algorithm for Discrete Tomography”. *IEEE Transactions on Image Processing* 20.9 (2011), pp. 2542–2553 (cit. on p. 23).

[14] J. Batson and L. Royer. “Noise2Self: Blind Denoising by Self-Supervision”. In: *Proceedings of the 36th International Conference on Machine Learning.* Vol. 97. PMLR, 2019, pp. 524–533 (cit. on pp. 46, 48–50, 54, 65, 69–71, 74, 77).

[15] A. Beck and M. Teboulle. “Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems”. *IEEE Transactions on Image Processing* 18.11 (2009), pp. 2419–2434 (cit. on pp. 11, 60).

[16] C. Belthangady and L. A. Royer. “Applications, Promises, and Pitfalls of Deep Learning for Fluorescence Image Reconstruction”. *Nature Methods* 16.12 (2019), pp. 1215–1225 (cit. on pp. 45, 46, 74, 76).

[17] T. Bendory, A. Bartesaghi, and A. Singer. “Single-Particle Cryo-Electron Microscopy: Mathematical Theory, Computational Challenges, and Opportunities”. *IEEE Signal Processing Magazine* 37.2 (2020), pp. 58–76 (cit. on pp. 2, 102).

[18] T. Bepler, K. Kelley, A. J. Noble, and B. Berger. “Topaz-Denoise: General Deep Denoising Models for Cryo-EM and Cryo-ET”. *Nature Communications* 11.1 (Oct. 2020) (cit. on p. 119).

[19] J. Bian, J. H. Siewerdsen, X. Han, E. Y. Sidky, J. L. Prince, C. A. Pelizzari, and X. Pan. “Evaluation of Sparse-View Reconstruction From Flat-Panel-Detector Cone-Beam CT”. *Physics in Medicine and Biology* 55.22 (2010), pp. 6575–6599 (cit. on p. 23).

[20] A. Biguri, R. Lindroos, R. Bryll, H. Towsyfyan, H. Deyhle, I. E. k. Harrane, R. Boardman, M. Mavrogordato, M. Dosanjh, S. Hancock, and et al. “Arbitrarily Large Tomography With Iterative Algorithms on Multiple GPUs Using the TIGRE Toolbox”. *Journal of Parallel and Distributed Computing* 146 (2020), pp. 52–63 (cit. on p. 92).

[21] F. Bleichrodt, T. van Leeuwen, W. J. Palenstijn, W. van Aarle, J. Sijbers, and K. J. Batenburg. “Easy Implementation of Advanced Tomography Algorithms Using the ASTRA Toolbox With Spot Operators”. *Numerical Algorithms* 71.3 (2015), pp. 673–697 (cit. on pp. 93, 95).

[22] P. Bleuet, E. Welcomme, E. Dooryhée, J. Susini, J.-L. Hodeau, and P. Walter. “Probing the Structure of Heterogeneous Diluted Materials By Diffraction Tomography”. *Nature Materials* 7 (2008), pp. 468–472 (cit. on pp. 74, 81).

[23] R. Boistel, N. Pollet, J.-Y. Tinevez, P. Cloetens, and M. Schlenker. "Irradiation Damage To Frog Inner Ear During Synchrotron Radiation Tomographic Investigation". *Journal of Electron Spectroscopy and Related Phenomena* 170.1-3 (2009), pp. 37–41 (cit. on p. 74).

[24] F. G. Bossema, M. Domínguez-Delmás, W. J. Palenstijn, A. Kostenko, J. Dorscheid, S. B. Coban, E. Hermens, and K. J. Batenburg. "A Novel Method for Dendrochronology of Large Historical Wooden Objects Using Line Trajectory X-Ray Tomography". *Scientific Reports* 11.1 (2021) (cit. on p. 2).

[25] D. J. Brenner and E. J. Hall. "Computed Tomography - an Increasing Source of Radiation Exposure". *New England Journal of Medicine* 357.22 (2007), pp. 2277–2284 (cit. on p. 6).

[26] J. Brokish and Y. Bresler. "Sampling Requirements for Circular Cone Beam Tomography". *2006 IEEE Nuclear Science Symposium Conference Record* (2006) (cit. on p. 25).

[27] T.-O. Buchholz, M. Jordan, G. Pigino, and F. Jug. "Cryo-Care: Content-Aware Image Restoration for Cryo-Transmission Electron Microscopy Data". *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)* (2019) (cit. on pp. 45, 46, 71, 74).

[28] M. Bührer, M. Stampanoni, X. Rochet, F. Büchi, J. Eller, and F. Marone. "High-Numerical-Aperture Macroscope Optics for Time-Resolved Experiments". *Journal of Synchrotron Radiation* 26.4 (2019), pp. 1161–1172 (cit. on pp. 74, 109, 152).

[29] M. Bührer, H. Xu, J. Eller, J. Sijbers, M. Stampanoni, and F. Marone. "Unveiling Water Dynamics in Fuel Cells From Time-Resolved Tomographic Microscopy Data". *Scientific Reports* 10.1 (2020) (cit. on pp. 19, 74, 79, 83).

[30] M. Bührer, H. Xu, A. A. Hendriksen, F. N. Büchi, J. Eller, M. Stampanoni, and F. Marone. "Deep learning based classification of dynamic processes in time-resolved X-ray tomographic microscopy". *Scientific Reports* 11.1 (Dec. 2021) (cit. on p. 140).

[31] T. M. Buzug. *Computed Tomography : from Photon Statistics to Modern Cone-Beam CT*. Springer, 2008 (cit. on pp. 5–9, 22, 26, 45, 51, 57, 58, 77, 78, 91, 92, 153).

[32] E. Cha, J. Jang, J. Lee, E. Lee, and J. C. Ye. "Boosting CNN Beyond Label in Inverse Problems". *CoRR* (2019). arXiv: `1906.07330 [cs.CV]` (cit. on p. 46).

[33] A. Chambolle and T. Pock. "A First-Order Primal-Dual Algorithm for Convex Problems With Applications To Imaging". *Journal of Mathematical Imaging and Vision* 40.1 (2010), pp. 120–145 (cit. on p. 11).

[34] W. Chang, J. M. Lee, K. Lee, J. H. Yoon, M. H. Yu, J. K. Han, and B. I. Choi. “Assessment of a Model-Based, Iterative Reconstruction Algorithm (MBIR) Regarding Image Quality and Dose Reduction in Liver Computed Tomography”. *Investigative Radiology* 48.8 (2013), pp. 598–606 (cit. on pp. 45, 57).

[35] C. Chen, W. Bai, R. H. Davies, A. N. Bhuva, C. H. Manisty, J. B. Augusto, J. C. Moon, N. Aung, A. M. Lee, M. M. Sanghvi, and et al. “Improving the Generalizability of Convolutional Neural Network-Based Segmentation on CMR Images”. *Frontiers in Cardiovascular Medicine* 7 (2020) (cit. on p. 13).

[36] R. J. Chen, M. Y. Lu, T. Y. Chen, D. F. K. Williamson, and F. Mahmood. “Synthetic Data in Machine Learning for Medicine and Healthcare”. *Nature Biomedical Engineering* 5.6 (June 2021), pp. 493–497 (cit. on p. 14).

[37] Z. Cheng, M. Gadelha, S. Maji, and D. Sheldon. “A Bayesian Perspective on the Deep Image Prior”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2019 (cit. on pp. 46, 62).

[38] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. “3D U-Net: Learning Dense Volumetric Segmentation From Sparse Annotation”. *Lecture Notes in Computer Science* (2016), pp. 424–432 (cit. on pp. 11, 41, 69).

[39] V. Cnudde and M. N. Boone. “High-Resolution X-ray Computed Tomography in Geosciences: a Review of the Current Technology and Applications”. *Earth-Science Reviews* 123 (2013), pp. 1–17 (cit. on pp. 21, 22).

[40] S. B. Coban, A. A. Hendriksen, D. M. Pelt, W. J. Palenstijn, and K. J. Batenburg. *Oatmeal Data: Experimental cone-beam tomographic data for techniques to improve image resolution.* 2018 (cit. on p. 39).

[41] S. B. Coban. “Practical approaches to reconstruction and analysis for 3D and dynamic 3D computed tomography”. PhD thesis. The University of Manchester (United Kingdom), 2016 (cit. on p. 5).

[42] 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”. *Journal of Imaging* 6.4 (2020), p. 18 (cit. on pp. 2, 38, 109).

[43] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. “Image Denoising By Sparse 3-D Transform-Domain Collaborative Filtering”. *IEEE Transactions on Image Processing* 16.8 (2007), pp. 2080–2095 (cit. on pp. 47, 60).

[44] F. De Carlo, D. Gürsoy, D. J. Ching, K. J. Batenburg, W. Ludwig, L. Mancini, F. Marone, R. Mokso, D. M. Pelt, J. Sijbers, and et al. “TomoBank: a Tomographic Data Repository for Computational X-Ray Science”. *Measurement Science and Technology* 29.3 (2018), p. 034004 (cit. on pp. 64, 109, 152).

[45] F. Dennerlein and A. Maier. “Approximate Truncation Robust Computed Tomography-ATRACT”. *Physics in Medicine and Biology* 58.17 (2013), pp. 6133–6148 (cit. on p. 30).

[46] S. Dittmer, T. Kluth, P. Maass, and D. Otero Baguer. “Regularization By Architecture: a Deep Prior Approach for Inverse Problems”. *Journal of Mathematical Imaging and Vision* 62.3 (2019), pp. 456–470 (cit. on p. 46).

[47] M. L. Eaton. “Chapter 2, Random Vectors”. In: *Multivariate Statistics.* Vol. Volume 53. Lecture Notes–Monograph Series. Institute of Mathematical Statistics, 2007, pp. 70–102 (cit. on p. 54).

[48] M. Ebner, F. Geldmacher, F. Marone, M. Stampanoni, and V. Wood. “X-Ray Tomography of Porous, Transition Metal Oxide Based Lithium Ion Battery Electrodes”. *Advanced Energy Materials* 3.7 (2013), pp. 845–850 (cit. on p. 4).

[49] M. van Eijnatten, L. Rundo, K. J. Batenburg, F. Lucka, E. Beddowes, C. Caldas, F. A. Gallagher, E. Sala, C.-B. Schönlieb, and R. Woitek. “3D Deformable Registration of Longitudinal Abdominopelvic CT Images Using Unsupervised Deep Learning”. *Computer Methods and Programs in Biomedicine* 208 (2021), p. 106261 (cit. on p. 13).

[50] J. Eller, J. Roth, F. Marone, M. Stampanoni, A. Wokaun, and F. N. Büchi. “Implications of Polymer Electrolyte Fuel Cell Exposure To Synchrotron Radiation on Gas Diffusion Layer Water Distribution”. *Journal of Power Sources* 245 (2014), pp. 796–800 (cit. on p. 74).

[51] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. “Dermatologist-Level Classification of Skin Cancer With Deep Neural Networks”. *Nature* 542.7639 (Jan. 2017), pp. 115–118 (cit. on p. 119).

[52] D. Evangelista, M. Terreran, A. Pretto, M. Moro, C. Ferrari, and E. Menegatti. “3D Mapping of X-Ray Images in Inspections of Aerospace Parts”. In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).* Vol. 1. 2020, pp. 1223–1226 (cit. on pp. 2, 91).

[53] L. A. Feldkamp, L. C. Davis, and J. W. Kress. “Practical Cone-Beam Algorithm”. *Journal of the Optical Society of America A* 1.6 (1984), p. 612 (cit. on pp. 9, 25, 109).

[54] J. A. Fessler. “On NUFFT-Based Gridding for Non-Cartesian MRI”. *Journal of Magnetic Resonance* 188.2 (2007), pp. 191–195 (cit. on p. 10).

[55] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian. “Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data”. *IEEE Transactions on Image Processing* 17.10 (2008), pp. 1737–1754 (cit. on p. 9).

[56] F. García-Moreno, P. H. Kamm, T. R. Neu, F. Bülk, R. Mokso, C. M. Schlepütz, M. Stampanoni, and J. Banhart. “Using X-ray Tomoscopy To Explore the Dynamics of Foaming Metal”. *Nature Communications* 10.1 (2019) (cit. on pp. 4, 73, 74, 79).

[57] P. Gilbert. “Iterative Methods for the Three-Dimensional Reconstruction of an Object From Projections”. *Journal of theoretical biology* 36.1 (1972), pp. 105–117 (cit. on p. 10).

[58] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2672–2680 (cit. on p. 118).

[59] R. Gordon, R. Bender, and G. T. Herman. “Algebraic Reconstruction Techniques (ART) for Three-Dimensional Electron Microscopy and X-Ray Photography”. *Journal of Theoretical Biology* 29.3 (1970), pp. 471–481 (cit. on p. 10).

[60] J. Gregor and T. Benson. “Computational Analysis and Improvement of SIRT”. *IEEE Transactions on Medical Imaging* 27.7 (2008), pp. 918–924 (cit. on pp. 10, 60, 78, 92).

[61] H. Guan and R. Gordon. “Computed Tomography Using Algebraic Reconstruction Techniques (ARTs) With Different Projection Access Schemes: a Comparison Study Under Practical Situations”. *Physics in Medicine and Biology* 41.9 (1996), pp. 1727–1743 (cit. on p. 22).

[62] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. “Improved Training of Wasserstein GANs”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett. 2017, pp. 5767–5777 (cit. on p. 14).

[63] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, and et al. “Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs”. *JAMA* 316.22 (Dec. 2016), p. 2402 (cit. on p. 119).

[64] D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen. “Tomopy: a Framework for the Analysis of Synchrotron Tomographic Data”. *Journal of Synchrotron Radiation* 21.5 (2014), pp. 1188–1193 (cit. on pp. 64, 92, 113, 153, 154).

[65] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, and et al. “Array Programming With NumPy”. *Nature* 585.7825 (2020), pp. 357–362 (cit. on pp. 100, 101).

[66] T. Hastie, R. Tibshirani, and J. Friedman. “The Elements of Statistical Learning”. *Springer Series in Statistics* (2009) (cit. on pp. 8, 11, 12, 48, 76).

[67] A. Hendriksen. *ahendriksen/cone_ balls: v0.2.2*. Version v0.2.2. 2019 (cit. on p. 36).

[68] A. Hendriksen. *ahendriksen/on_the_fly: Paper release.* Version paper-release. 2019 (cit. on p. 41).

[69] A. Hendriksen, R. de Heide, and P. Grünwald. “Optional Stopping With Bayes Factors: a Categorization and Extension of Folklore Results, With an Application To Invariant Situations”. *Bayesian Analysis* (2020) (cit. on p. 139).

[70] A. Hendriksen and D. Schut. *ahendriksen/ts_algorithms.* 2021 (cit. on p. 113).

[71] A. A. Hendriksen. *ahendriksenh/msd_pytorch: v0.7.2.* Version v0.7.2. 2019 (cit. on pp. 59, 152).

[72] A. A. Hendriksen, M. Bührer, L. Leone, M. Merlini, N. Vigano, D. M. Pelt, F. Marone, M. di Michiel, and K. J. Batenburg. “Deep Denoising for Multi-Dimensional Synchrotron X-Ray Tomography Without High-Quality Reference Data”. *Scientific Reports* 11.1 (2021) (cit. on pp. 73, 139).

[73] A. A. Hendriksen, D. M. Pelt, W. J. Palenstijn, S. B. Coban, and K. J. Batenburg. “On-The-Fly Machine Learning for Improving Image Resolution in Tomography”. *Applied Sciences* 9.12 (2019) (cit. on pp. 21, 69, 139).

[74] A. A. Hendriksen, D. Schut, W. J. Palenstijn, N. Viganó, D. M. Kim Jisoo Pelt, T. van Leeuwen, and K. J. Batenburg. “Tomosipo: Fast, Flexible, and Convenient 3D Tomography for Complex Scanning Geometries in Python”. *Optics Express* (2021) (cit. on pp. 91, 106, 108, 139).

[75] A. A. Hendriksen, D. M. Pelt, and K. J. Batenburg. “Noise2inverse: Self-Supervised Deep Convolutional Denoising for Tomography”. *IEEE Transactions on Computational Imaging* (2020), pp. 1–1 (cit. on pp. 45, 74, 76–79, 87–89, 139).

[76] M. R. Hestenes and E. Stiefel. “Methods of Conjugate Gradients for Solving Linear Systems”. *Journal of Research of the National Bureau of Standards* 49.6 (1952) (cit. on p. 10).

[77] M. Holler, M. Odstrcil, M. Guizar-Sicairos, M. Lebugle, E. Müller, S. Finizio, G. Tinti, C. David, J. Zusman, W. Unglaub, and et al. “Three-Dimensional Imaging of Integrated Circuits With Macro- To Nanoscale Zoom”. *Nature Electronics* 2.10 (2019), pp. 464–470 (cit. on pp. 73, 74).

[78] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. “Image-to-image translation with conditional adversarial networks”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017 (cit. on p. 44).

[79] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun. “What Is the Best Multi-Stage Architecture for Object Recognition?” *2009 IEEE 12th International Conference on Computer Vision* (2009) (cit. on pp. 11, 75).

[80] K. H. Jin, H. Gupta, J. Yerly, M. Stuber, and M. Unser. “Time-Dependent Deep Image Prior for Dynamic MRI”. *CoRR* (2019). arXiv: `1910.01684 [eess.IV]` (cit. on p. 46).

[81] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. “Deep Convolutional Neural Network for Inverse Problems in Imaging”. *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4509–4522 (cit. on pp. 11, 32).

[82] J. S. Jørgensen, E. Ametova, G. Burca, G. Fardell, E. Papoutsellis, E. Pasca, K. Thielemans, M. Turner, R. Warr, W. R. B. Lionheart, and et al. “Core Imaging Library - Part I: a versatile Python framework for tomographic imaging”. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379.2204 (July 2021), p. 20200192 (cit. on p. 113).

[83] M. Kalke and S. Siltanen. “Sinogram Interpolation Method for Sparse-Angle Tomography”. *Applied Mathematics* 05.03 (2014), pp. 423–441 (cit. on p. 81).

[84] E. Kang, H. J. Koo, D. H. Yang, J. B. Seo, and J. C. Ye. “Cycle-Consistent Adversarial Denoising Network for Multiphase Coronary CT Angiography”. *Medical Physics* 46.2 (2018), pp. 550–562 (cit. on pp. 13, 118).

[85] E. Kang, J. Min, and J. C. Ye. “A Deep Convolutional Neural Network Using Directional Wavelets for Low-Dose X-Ray CT Reconstruction”. *Medical Physics* 44.10 (2017), e360–e375 (cit. on pp. 2, 11, 45, 46, 51, 74).

[86] N. Kardjilov, I. Manke, A. Hilger, M. Strobl, and J. Banhart. “Neutron Imaging in Materials Science”. *Materials Today* 14.6 (2011), pp. 248–256 (cit. on p. 21).

[87] A. G. Kashkooli, S. Farhad, D. U. Lee, K. Feng, S. Litster, S. K. Babu, L. Zhu, and Z. Chen. “Multiscale Modeling of Lithium-Ion Battery Electrodes Based on Nano-Scale X-Ray Computed Tomography”. *Journal of Power Sources* 307 (2016), pp. 496–509 (cit. on pp. 23, 74).

[88] A. Katsevich. “An Improved Exact Filtered Backprojection Algorithm for Spiral Computed Tomography”. *Advances in Applied Mathematics* 32.4 (2004), pp. 681–697 (cit. on p. 103).

[89] A. Katsevich. “Theoretically Exact Filtered Backprojection-Type Inversion Algorithm for Spiral CT”. *SIAM Journal on Applied Mathematics* 62.6 (2002), pp. 2012–2026 (cit. on pp. 9, 92).

[90] B. Kawar, G. Vaksman, and M. Elad. “Snips: Solving Noisy Inverse Problems Stochastically”. *CoRR* (2021). arXiv: `2105.14951 [eess.IV]` (cit. on p. 119).

[91] B. Kawar, G. Vaksman, and M. Elad. “Stochastic Image Denoising By Sampling From the Posterior Distribution”. *CoRR* (2021). arXiv: `2101.09552 [eess.IV]` (cit. on p. 119).

[92] R. A. Ketcham and W. D. Carlson. “Acquisition, Optimization and Interpretation of X-ray Computed Tomographic Imagery: Applications To the Geosciences”. *Computers & Geosciences* 27.4 (2001), pp. 381–400 (cit. on pp. 4, 22, 23).

[93] J. Kim, M. Kagias, F. Marone, and M. Stampanoni. “X-Ray Scattering Tensor Tomography With Circular Gratings”. *Applied Physics Letters* 116.13 (2020), p. 134102 (cit. on pp. 20, 91, 107–109).

[94] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015 (cit. on pp. 34, 60).

[95] A. M. Kingston, G. R. Myers, S. J. Latham, B. Recur, H. Li, and A. P. Sheppard. “Space-Filling X-ray Source Trajectories for Efficient Scanning in Large-Angle Cone-Beam Computed Tomography”. *IEEE Transactions on Computational Imaging* 4.3 (2018), pp. 447–458 (cit. on pp. 5, 25).

[96] F. Knoll, J. Zbontar, A. Sriram, M. J. Muckley, M. Bruno, A. Defazio, M. Parente, K. J. Geras, J. Katsnelson, H. Chandarana, and et al. “fastMRI: a Publicly Available Raw K-Space and DICOM Dataset of Knee Images for Accelerated MR Image Reconstruction Using Machine Learning”. *Radiology: Artificial Intelligence* 2.1 (2020) (cit. on pp. 13, 71).

[97] H. Kobayashi, A. C. Solak, J. Batson, and L. A. Royer. “Image Deconvolution Via Noise-Tolerant Self-Supervised Inversion”. *CoRR* (2020). arXiv: `2006.06156 [cs.CV]` (cit. on p. 74).

[98] F. K. Kopp, R. A. Nasirudin, K. Mei, A. Fehringer, F. Pfeiffer, E. J. Rummeny, and P. B. Noël. “Region of Interest Processing for Iterative Reconstruction in X-ray Computed Tomography”. *Medical Imaging 2015: Physics of Medical Imaging* (2015) (cit. on p. 26).

[99] A. Kostenko, W. Palenstijn, S. Coban, A. Hendriksen, R. van Liere, and K. Batenburg. “Prototyping X-Ray Tomographic Reconstruction Pipelines With Flexbox”. *SoftwareX* 11 (2020), p. 100364 (cit. on p. 139).

[100] A. Krull, T.-O. Buchholz, and F. Jug. “Noise2Void - Learning Denoising From Single Noisy Images”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on pp. 46, 47, 74).

[101] M. Krumm, C. Sauerwein, V. Hämmerle, S. Heile, T. Schön, A. Jung, and M. Sindel. “Rapid robotic X-ray computed tomography of large assemblies in automotive production”. In: *Proceedings of the 8th Conference on Industrial Computed Tomography (iCT 2018), Wels, Austria*. 2018, pp. 6–9 (cit. on p. 91).

[102] P. Kuchment and L. Kunyansky. “Mathematics of Photoacoustic and Thermoacoustic Tomography”. In: *Handbook of Mathematical Methods in Imaging*. Ed. by O. Scherzer. New York, NY: Springer New York, 2011, pp. 817–865 (cit. on p. 45).

[103] M. J. Lagerwerf, A. A. Hendriksen, J.-W. Buurlage, and K. J. Batenburg. “Noise2Filter: Fast, Self-Supervised Learning and Real-Time Reconstruction for 3D Computed Tomography”. *Machine Learning: Science and Technology* 2.1 (2020), p. 015012 (cit. on pp. 119, 139).

[104] S. Laine, T. Karras, J. Lehtinen, and T. Aila. “High-Quality Self-Supervised Deep Image Denoising”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 6970–6980 (cit. on p. 46).

[105] L. Landweber. “An Iteration Formula for Fredholm Integral Equations of the First Kind”. *American Journal of Mathematics* 73.3 (1951), p. 615 (cit. on pp. 10, 95).

[106] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. “Photo-realistic single image super-resolution using a generative adversarial network”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on pp. 23, 26, 44, 118).

[107] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. “Noise2Noise: Learning Image Restoration without Clean Data”. In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 2965–2974 (cit. on pp. 46, 48, 50, 72, 74).

[108] M. Li, H. Yang, and H. Kudo. “An Accurate Iterative Reconstruction Algorithm for Sparse Objects: Application To 3D Blood Vessel Reconstruction From a Limited Number of Projections”. *Physics in Medicine and Biology* 47.15 (2002), pp. 2599–2609 (cit. on p. 23).

[109] M. Liebi, M. Georgiadis, J. Kohlbrecher, M. Holler, J. Raabe, I. Usov, A. Menzel, P. Schneider, O. Bunk, and M. Guizar-Sicairos. “Small-Angle X-Ray Scattering Tensor Tomography: Model of the Three-Dimensional Reciprocal-Space Map, Reconstruction Algorithm and Angular Sampling requirements”. *Acta Crystallographica Section A* 74.1 (2018), pp. 12–24 (cit. on p. 108).

[110] M. Liebi, M. Georgiadis, A. Menzel, P. Schneider, J. Kohlbrecher, O. Bunk, and M. Guizar-Sicairos. “Nanostructure Surveys of Macroscopic Specimens By Small-Angle Scattering Tensor Tomography”. *Nature* 527.7578 (2015), pp. 349–352 (cit. on p. 107).

[111] H. Liu, S. Kazemiabnavi, A. Grenier, G. Vaughan, M. Di Michiel, B. J. Polzin, K. Thornton, K. W. Chapman, and P. J. Chupas. “Quantifying Reaction and Rate Heterogeneity in Battery Electrodes in 3D Through Operando X-Ray Diffraction Computed Tomography”. *ACS Applied Materials & Interfaces* 11.20 (2019), pp. 18386–18394 (cit. on p. 4).

[112] J. Liu, Y. Sun, C. Eldeniz, W. Gan, H. An, and U. S. Kamilov. “RARE: Image Reconstruction Using Deep Priors Learned Without Ground Truth”. *IEEE Journal of Selected Topics in Signal Processing* (2020), pp. 1–1 (cit. on pp. 46, 74, 81).

[113] G. Lovric, S. F. Barré, J. C. Schittny, M. Roth-Kleiner, M. Stampanoni, and R. Mokso. “Dose Optimization Approach To Fast X-Ray Microtomography of the Lung Alveoli”. *Journal of Applied Crystallography* 46.4 (2013), pp. 856–860 (cit. on p. 74).

[114] C. Maaß, M. Knaup, and M. Kachelrieß. “New Approaches To Region of Interest Computed Tomography”. *Medical Physics* 38.6Part1 (2011), pp. 2868–2878 (cit. on p. 30).

[115] E. Maire and P. J. Withers. “Quantitative X-Ray Tomography”. *International Materials Reviews* 59.1 (2013), pp. 1–43 (cit. on pp. 4, 119).

[116] Y. Makinen, L. Azzari, and A. Foi. “Exact Transform-Domain Noise Variance for Collaborative Filtering of Stationary Correlated Noise”. *2019 IEEE International Conference on Image Processing (ICIP)* (2019) (cit. on p. 60).

[117] A. Malecki, G. Potdevin, T. Biernath, E. Eggl, K. Willer, T. Lasser, J. Maisenbacher, J. Gibmeier, A. Wanner, and F. Pfeiffer. “X-Ray Tensor Tomography”. *EPL (Europhysics Letters)* 105.3 (2014), p. 38002 (cit. on pp. 107, 108).

[118] F. Marone and M. Stampanoni. “Regridding Reconstruction Algorithm for Real-Time Tomographic Imaging”. *Journal of Synchrotron Radiation* 19.6 (2012), pp. 1029–1037 (cit. on pp. 10, 19, 45, 78, 83, 153).

[119] D. N. Mastronarde and S. R. Held. “Automated Tilt Series Alignment and Tomographic Reconstruction in IMOD”. *Journal of Structural Biology* 197.2 (2017), pp. 102–113 (cit. on p. 92).

[120] G. Mataev, P. Milanfar, and M. Elad. “DeepRED: Deep Image Prior Powered by RED”. In: *The IEEE International Conference on Computer Vision (ICCV) Workshops.* 2019 (cit. on p. 46).

[121] G. Matrone, A. S. Savoia, G. Caliano, and G. Magenes. “The Delay Multiply and Sum Beamforming Algorithm in Ultrasound B-Mode Medical Imaging”. *IEEE Transactions on Medical Imaging* 34.4 (2015), pp. 940–949 (cit. on pp. 45, 71).

[122] M. T. McCann, K. H. Jin, and M. Unser. “Convolutional Neural Networks for Inverse Problems in Imaging: a Review”. *IEEE Signal Processing Magazine* 34.6 (2017), pp. 85–95 (cit. on pp. 11, 75).

[123] C. H. McCollough, A. C. Bartley, R. E. Carter, B. Chen, T. A. Drees, P. Edwards, D. R. Holmes, A. E. Huang, F. Khan, S. Leng, and et al. “Low-Dose CT for the Detection and Classification of Metastatic Liver Lesions: Results of the 2016 Low Dose CT Grand Challenge”. *Medical Physics* 44.10 (2017), e339–e352 (cit. on pp. 4, 13, 45, 62, 71).

[124] S. A. McDonald, P. Reischig, C. Holzner, E. M. Lauridsen, P. J. Withers, A. P. Merkle, and M. Feser. “Non-Destructive Mapping of Grain Orientations in 3D By Laboratory X-Ray Microscopy”. *Scientific Reports* 5.1 (2015) (cit. on p. 104).

[125] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. “Unrolled Generative Adversarial Networks”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.* 2017 (cit. on p. 14).

[126] C. A. Metzler, A. Mousavi, R. Heckel, and R. G. Baraniuk. "Unsupervised Learning With Stein's Unbiased Risk Estimator". *CoRR* (2018). arXiv: `1805.10531 [stat.ML]` (cit. on p. 46).

[127] P. Midgley and M. Weyland. "3D Electron Microscopy in the Physical Sciences: the Development of Z-Contrast and EFTEM Tomography". *Ultramicroscopy* 96.3-4 (2003), pp. 413–431 (cit. on p. 58).

[128] P. A. Midgley and R. E. Dunin-Borkowski. "Electron Tomography and Holography in Materials Science". *Nature Materials* 8.4 (2009), pp. 271–280 (cit. on pp. 21, 98).

[129] J. Minnema, M. Van Eijnatten, J. Wolff, A. A. Hendriksen, K. J. Batenburg, and T. Forouzanfar. "CT Image Segmentation for Additive Manufactured Skull Implants Using Deep Learning". en. *Transactions on Additive Manufacturing Meets Medicine* (2019), Vol 1 (2019): Trans. AMMM (cit. on p. 139).

[130] A. Mirone, E. Brun, E. Gouillart, P. Tafforeau, and J. Kieffer. "The PyHST2 Hybrid Distributed Code for High Speed Tomographic Reconstruction With Iterative Reconstruction and a Priori Knowledge Capabilities". *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 324 (2014), pp. 41–48 (cit. on p. 92).

[131] R. Mokso, C. M. Schlepütz, G. Theidel, H. Billich, E. Schmid, T. Celcer, G. Mikuljan, L. Sala, F. Marone, N. Schlumpf, and et al. "GigaFRoST: the Gigabit Fast Readout System for Tomography". *Journal of Synchrotron Radiation* 24.6 (2017), pp. 1250–1259 (cit. on pp. 6, 118).

[132] K. Mueller. "Fast and accurate three-dimensional reconstruction from cone-beam projection data using algebraic methods". PhD thesis. The Ohio State University, 1998 (cit. on p. 22).

[133] V. Nair and G. E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning.* PMLR, 2010, pp. 807–814 (cit. on pp. 11, 75).

[134] F. Natterer. *The mathematics of computerized tomography.* SIAM, 2001 (cit. on pp. 9, 10).

[135] K. Niinimäki, S. Siltanen, and V. Kolehmainen. "Bayesian Multiresolution Method for Local Tomography in Dental X-ray Imaging". *Physics in Medicine and Biology* 52.22 (2007), pp. 6663–6678 (cit. on p. 22).

[136] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis. "CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations". In: *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS).* 2017 (cit. on p. 100).

[137] 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”. *Journal of Microscopy* 206.1 (2002), pp. 33–40 (cit. on p. 83).

[138] X. Pan, E. Y. Sidky, and M. Vannier. “Why Do Commercial CT Scanners Still Employ Traditional, Filtered Back-Projection for Image Reconstruction?” *Inverse Problems* 25.12 (2009), p. 123009 (cit. on p. 92).

[139] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. “Automatic differentiation in PyTorch”. In: *NIPS-W*. 2017 (cit. on pp. 34, 68, 93, 100).

[140] D. M. Pelt, K. J. Batenburg, and J. A. Sethian. “Improving Tomographic Reconstruction From Limited Data Using Mixed-Scale Dense Convolutional Neural Networks”. *Journal of Imaging* 4.11 (2018), p. 128 (cit. on pp. 9, 11, 13, 32, 45, 59, 69, 74).

[141] D. M. Pelt, D. Gürsoy, W. J. Palenstijn, J. Sijbers, F. De Carlo, and K. J. Batenburg. “Integration of TomoPy and the ASTRA Toolbox for Advanced Processing and Reconstruction of Tomographic Synchrotron Data”. *Journal of Synchrotron Radiation* 23.3 (2016), pp. 842–849 (cit. on p. 10).

[142] D. M. Pelt, A. A. Hendriksen, and K. J. Batenburg. “Foam-Like Phantoms for Comparing Tomography Algorithms”. *Journal of Synchrotron Radiation* 29.1 (Jan. 2022) (cit. on p. 140).

[143] D. M. Pelt and J. A. Sethian. “A Mixed-Scale Dense Convolutional Neural Network for Image Analysis”. *Proceedings of the National Academy of Sciences* 115.2 (2017), pp. 254–259 (cit. on pp. 11, 32, 34, 59, 68, 69, 82, 152).

[144] D. M. Pelt and K. J. Batenburg. “Fast Tomographic Reconstruction From Limited Data Using Artificial Neural Networks”. *IEEE Transactions on Image Processing* 22.12 (2013), pp. 5238–5251 (cit. on p. 119).

[145] J. M. Perkel. “Why Jupyter Is Data Scientists’ Computational Notebook of Choice”. *Nature* 563.7732 (2018), pp. 145–147 (cit. on p. 93).

[146] P. Pietsch and V. Wood. “X-Ray Tomography for Lithium Ion Battery Research: a Practical Guide”. *Annual Review of Materials Research* 47.1 (2017), pp. 451–479 (cit. on p. 73).

[147] G. E. Possin and C.-Y. Wei. *CT array with improved photosensor linearity and reduced crosstalk*. US Patent 5,430,298. 1995 (cit. on p. 25).

[148] J. Poudel, Y. Lou, and M. A. Anastasio. “A Survey of Computational Frameworks for Solving the Acoustic Inverse Problem in Three-Dimensional Photoacoustic Computed Tomography”. *Physics in Medicine & Biology* 64.14 (2019), 14TR01 (cit. on p. 45).

[149] Y. Quan, M. Chen, T. Pang, and H. Ji. “Self2self With Dropout: Learning Self-Supervised Denoising From Single Image”. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020) (cit. on p. 74).

[150] M. Ravasi and I. Vasconcelos. “Pylops — a Linear-Operator Python Library for Scalable Algebra and Optimization”. *SoftwareX* 11 (2020), p. 100361 (cit. on pp. 92, 113).

[151] S. V. Ravuri and O. Vinyals. “Classification Accuracy Score for Conditional Generative Models”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada.* Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett. 2019, pp. 12247–12258 (cit. on p. 14).

[152] S. D. Rawson, J. Maksimcuka, P. J. Withers, and S. H. Cartmell. “X-ray Computed Tomography in Life Sciences”. *BMC Biology* 18.21 (2020) (cit. on p. 73).

[153] S. J. Reddi, S. Kale, and S. Kumar. “On the Convergence of ADAM and Beyond”. In: *International Conference on Learning Representations.* 2018 (cit. on pp. 11, 76, 152).

[154] D. Rogers, D. Rogers, J. Adams, and M.-H. (1968-1995). *Mathematical Elements for Computer Graphics.* McGraw-Hill, 1990 (cit. on p. 98).

[155] Y. Romano, J. Isidoro, and P. Milanfar. “RAISR: Rapid and Accurate Image Super Resolution”. *IEEE Transactions on Computational Imaging* 3.1 (2017), pp. 110–125 (cit. on p. 23).

[156] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015* (2015), pp. 234–241 (cit. on pp. 11, 32, 41, 68, 69).

[157] J. Roth, J. Eller, and F. N. Büchi. “Effects of Synchrotron Radiation on Fuel Cell Materials”. *Journal of The Electrochemical Society* 159.8 (2012), F449–F455 (cit. on p. 74).

[158] B. P. Rynne and M. A. Youngson. “Linear Functional Analysis”. *Springer Undergraduate Mathematics Series* (2008) (cit. on p. 53).

[159] M. Saadatfar, F. Garcia-Moreno, S. Hutzler, A. P. Sheppard, M. A. Knackstedt, J. Banhart, and D. Weaire. “Imaging of Metallic Foams Using X-ray micro-CT”. *Colloids and Surfaces A: Physicochemical and Engineering Aspects* 344.1-3 (2009), pp. 107–112 (cit. on pp. 21, 23).

[160] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. “Image Super-Resolution Via Iterative Refinement”. *CoRR* (2021). arXiv: `2104.07636 [eess.IV]` (cit. on p. 119).

[161] M. Salamon, R. Hanke, P. Krüger, F. Sukowski, N. Uhlmann, and V. Voland. “Comparison of Different Methods for Determining the Size of a Focal Spot of Microfocus X-Ray Tubes”. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 591.1 (2008), pp. 54–58 (cit. on p. 5).

[162] R. Schoonhoven, A. A. Hendriksen, D. M. Pelt, and K. J. Batenburg. "Lean: Graph-Based Pruning for Convolutional Neural Networks By Extracting Longest Chains". *CoRR* (2020). arXiv: `2011.06923 [cs.LG]` (cit. on p. 140).

[163] A.-K. Seghouane and Y. Saad. "Prewhitening High-Dimensional fMRI Data Sets Without Eigendecomposition". *Neural Computation* 26.5 (2014), pp. 907–919 (cit. on p. 60).

[164] P. Shearing, L. Howard, P. Jørgensen, N. Brandon, and S. Harris. "Characterization of the 3-dimensional Microstructure of a Graphite Negative Electrode From a Li-Ion Battery". *Electrochemistry Communications* 12.3 (2010), pp. 374–377 (cit. on p. 4).

[165] E. Shelhamer, J. Long, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (2017), pp. 640–651 (cit. on p. 32).

[166] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) (cit. on pp. 13, 74).

[167] J.-B. Sibarita. "Deconvolution Microscopy". *Advances in Biochemical Engineering/Biotechnology* (2005), pp. 201–243 (cit. on pp. 45, 51, 71).

[168] E. Y. Sidky, J. H. Jørgensen, and X. Pan. "Convex Optimization Problem Prototyping for Image Reconstruction in Computed Tomography With the Chambolle-Pock Algorithm". *Physics in Medicine and Biology* 57.10 (2012), pp. 3065–3091 (cit. on pp. 10, 87, 92, 154).

[169] E. Y. Sidky and X. Pan. "Image Reconstruction in Circular Cone-Beam Computed Tomography By Constrained, Total-Variation Minimization". *Physics in Medicine and Biology* 53.17 (2008), pp. 4777–4807 (cit. on p. 23).

[170] S. Soltanayev and S. Y. Chun. "Training deep learning based denoisers without ground truth data". In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 3261–3271 (cit. on p. 46).

[171] Y. Song and S. Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett. 2019, pp. 11895–11907 (cit. on p. 119).

[172] M. Stampanoni, G. Borchert, P. Wyss, R. Abela, B. Patterson, S. Hunt, D. Vermeulen, and P. Rüegsegger. "High Resolution X-ray Detector for Synchrotron-Based Microtomography". *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 491.1 (2002), pp. 291–301 (cit. on pp. 21, 22).

[173] S. Stock. *MicroComputed tomography : methodology and applications.* CRC Press, Taylor and Francis Group, 2020 (cit. on pp. 9, 73).

[174] Y. Sun, Z. Xia, and U. S. Kamilov. "Efficient and Accurate Inversion of Multiple Scattering With Deep Learning". *Optics Express* 26.11 (2018), p. 14678 (cit. on p. 45).

[175] C. Syben, M. Michen, B. Stimpel, S. Seitz, S. Ploner, and A. K. Maier. "Technical Note: Pyro-NN: Python Reconstruction Operators in Neural Networks". *Medical Physics* (2019) (cit. on p. 92).

[176] G. Tisson, P. Scheunders, and D. Van Dyck. "3D Region of Interest X-ray CT for Geometric Magnification From Multiresolution Acquisitions". *2nd IEEE International Symposium on Biomedical Imaging: Macro to Nano* (2004) (cit. on pp. 26, 30).

[177] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. "Deep Image Prior". *Int. J. Comput. Vis.* 128.7 (2020), pp. 1867–1888 (cit. on pp. 46, 60).

[178] A. van der Vaart. *Mathematische Statistiek.* 1995 (cit. on p. 7).

[179] G. Van Rossum and F. L. Drake Jr. *Python tutorial.* Centrum voor Wiskunde en Informatica, 1995 (cit. on p. 96).

[180] L. Varga and R. Mokso. "Iterative High Resolution Tomography From Combined High-Low Resolution Sinogram Pairs". *Combinatorial Image Analysis* (2018), pp. 150–163 (cit. on pp. 26, 30).

[181] S. Venkatakrishnan, A. Ziabari, J. Hinkle, A. W. Needham, J. M. Warren, and H. Z. Bilheux. "Convolutional Neural Network Based Non-Iterative Reconstruction for Accelerating Neutron Tomography". *Machine Learning: Science and Technology* 2.2 (2021), p. 025031 (cit. on p. 13).

[182] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. "Plug-And-Play Priors for Model Based Reconstruction". *2013 IEEE Global Conference on Signal and Information Processing* (2013) (cit. on p. 48).

[183] R. Vescovi, M. Du, V. de Andrade, W. Scullin, D. Gürsoy, and C. Jacobsen. "Tomosaic: Efficient Acquisition and Reconstruction of Teravoxel Tomography Data Using Limited-Size Synchrotron X-ray Beams". *Journal of Synchrotron Radiation* 25.5 (2018), pp. 1478–1489 (cit. on p. 22).

[184] N. Viganò and W. Ludwig. "X-Ray Orientation Microscopy Using Topo-Tomography and Multi-Mode Diffraction Contrast Tomography". *Current Opinion in Solid State and Materials Science* 24.4 (2020), p. 100832 (cit. on pp. 20, 91, 104, 106).

[185] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. 1. 0. Contributors. "SciPy 1.0–fundamental Algorithms for Scientific Computing in Python". *arXiv e-prints*, arXiv:1907.10121 (2019), arXiv:1907.10121. arXiv: `1907.10121 [cs.MS]` (cit. on p. 60).

[186] N. T. Vo, R. C. Atwood, M. Drakopoulos, and T. Connolley. "Data Processing Methods and Data Acquisition for Samples Larger Than the Field of View in Parallel-Beam Tomography". *Optics Express* 29.12 (2021), p. 17849 (cit. on p. 92).

[187] J. Vogel, F. Schaff, A. Fehringer, C. Jud, M. Wieczorek, F. Pfeiffer, and T. Lasser. "Constrained X-Ray Tensor Tomography Reconstruction". *Opt. Express* 23.12 (2015), pp. 15134–15151 (cit. on p. 107).

[188] N. Wadeson and M. Basham. "Savu: a Python-Based, MPI Framework for Simultaneous Processing of Multiple, N-Dimensional, Large Tomography Datasets". *CoRR* (2016). arXiv: `1610.08015 [cs.DC]` (cit. on pp. 92, 113).

[189] S. M. Walker, D. A. Schwyn, R. Mokso, M. Wicklein, T. Müller, M. Doube, M. Stampanoni, H. G. Krapp, and G. K. Taylor. "In Vivo Time-Resolved Microtomography Reveals the Mechanics of the Blowfly Flight Motor". *PLoS Biology* 12.3 (2014). Ed. by A. Hedenström, e1001823 (cit. on p. 74).

[190] C. Wang, U. Steiner, and A. Sepe. "Synchrotron Big Data Science". *Small* 14.46 (2018), p. 1802291 (cit. on pp. 4, 74).

[191] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. "Image Quality Assessment: From Error Visibility To Structural Similarity". *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612 (cit. on pp. 34, 62).

[192] B. R. Whiting, P. Massoumzadeh, O. A. Earl, J. A. O'Sullivan, D. L. Snyder, and J. F. Williamson. "Properties of Preprocessed Sinogram Data in X-Ray Computed Tomography". *Medical Physics* 33.9 (2006), pp. 3290–3303 (cit. on pp. 8, 9).

[193] P. A. Witte, M. Louboutin, N. Kukreja, F. Luporini, M. Lange, G. J. Gorman, and F. J. Herrmann. "A Large-Scale Framework for Symbolic Implementations of Seismic Inversion Algorithms in Julia". *Geophysics* 84.3 (2019), F57–F71 (cit. on pp. 92, 113).

[194] Y. D. Witte, J. Vlassenbroeck, and L. van Hoorebeke. "A Multiresolution Approach To Iterative Reconstruction Algorithms in X-ray Computed Tomography". *IEEE Transactions on Image Processing* 19.9 (2010), pp. 2419–2427 (cit. on p. 26).

[195] H. Xu, S. Nagashima, H. P. Nguyen, K. Kishita, F. Marone, F. N. Büchi, and J. Eller. "Temperature Dependent Water Transport Mechanism in Gas Diffusion Layers Revealed By Subsecond Operando X-Ray Tomographic Microscopy". *Journal of Power Sources* 490 (2021), p. 229492 (cit. on p. 4).

[196] B. Yaman, S. A. H. Hosseini, S. Moeller, J. Ellermann, K. Uğurbil, and M. Akçakaya. "Self-supervised Learning of Physics-guided Reconstruction Neural Networks Without Fully Sampled Reference Data". *Magnetic Resonance in Medicine* (2020) (cit. on pp. 46, 74).

[197] S. Zabler, P. Cloetens, J.-P. Guigay, J. Baruchel, and M. Schlenker. "Optimization of Phase Contrast Imaging Using Hard X Rays". *Review of Scientific Instruments* 76.7 (2005), p. 073705 (cit. on pp. 45, 71).

[198] G. L. Zeng, Y. Li, and Q. Huang. "Analytic Time-Of-Flight Positron Emission Tomography Reconstruction: Two-Dimensional Case". *Visual Computing for Industry, Biomedicine, and Art* 2.1 (2019) (cit. on p. 45).

[199] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising". *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155 (cit. on pp. 11, 47, 48, 50, 68, 69, 75).

[200] M. Zhussip, S. Soltanayev, and S. Y. Chun. "Extending Stein's unbiased risk estimator to train deep denoisers with correlated pairs of noisy images". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 1465–1475 (cit. on p. 46).

[201] A. Ziabari, D. H. Ye, S. Srivastava, K. D. Sauer, J.-B. Thibault, and C. A. Bouman. "2.5D Deep Learning for CT Image Reconstruction Using a Multi-GPU Implementation". *2018 52nd Asilomar Conference on Signals, Systems, and Computers* (2018) (cit. on pp. 76, 79).

List of publications

"Talking. . . , if I were as good at everything as at talking. . . "

"Praten. . . , als ik alles zo zou kunnen als praten. . . "

Johan Cruijff,
Tiq, 1 March 1967

Publications that are part of this thesis:

1. A. A. Hendriksen, D. M. Pelt, W. J. Palenstijn, S. B. Coban, and K. J. Batenburg. "On-The-Fly Machine Learning for Improving Image Resolution in Tomography". *Applied Sciences* 9.12 (2019).

2. A. A. Hendriksen, D. M. Pelt, and K. J. Batenburg. "Noise2inverse: Self-Supervised Deep Convolutional Denoising for Tomography". *IEEE Transactions on Computational Imaging* (2020), pp. 1–1.

3. A. A. Hendriksen, M. Bührer, L. Leone, M. Merlini, N. Vigano, D. M. Pelt, F. Marone, M. di Michiel, and K. J. Batenburg. "Deep Denoising for Multi-Dimensional Synchrotron X-Ray Tomography Without High-Quality Reference Data". *Scientific Reports* 11.1 (2021).

4. A. A. Hendriksen, D. Schut, W. J. Palenstijn, N. Viganó, D. M. Kim Jisoo Pelt, T. van Leeuwen, and K. J. Batenburg. "Tomosipo: Fast, Flexible, and Convenient 3D Tomography for Complex Scanning Geometries in Python". *Optics Express* (2021).

Publications that are not part of this thesis:

1. A. Hendriksen, R. de Heide, and P. Grünwald. "Optional Stopping With Bayes Factors: a Categorization and Extension of Folklore Results, With an Application To Invariant Situations". *Bayesian Analysis* (2020).

2. A. Kostenko, W. Palenstijn, S. Coban, A. Hendriksen, R. van Liere, and K. Batenburg. "Prototyping X-Ray Tomographic Reconstruction Pipelines With Flexbox". *SoftwareX* 11 (2020), p. 100364.

3. M. J. Lagerwerf, A. A. Hendriksen, J.-W. Buurlage, and K. J. Batenburg. "Noise2Filter: Fast, Self-Supervised Learning and Real-Time Reconstruction for 3D Computed Tomography". *Machine Learning: Science and Technology* 2.1 (2020), p. 015012.

4. J. Minnema, M. Van Eijnatten, J. Wolff, A. A. Hendriksen, K. J. Batenburg, and T. Forouzanfar. "CT Image Segmentation for Additive Manufactured Skull Implants Using Deep Learning". en. *Transactions on Additive Manufacturing Meets Medicine* (2019), Vol 1 (2019): Trans. AMMM.

5. R. Schoonhoven, A. A. Hendriksen, D. M. Pelt, and K. J. Batenburg. “Lean: Graph-Based Pruning for Convolutional Neural Networks By Extracting Longest Chains”. *CoRR* (2020). arXiv: `2011.06923 [cs.LG]`.

6. D. M. Pelt, A. A. Hendriksen, and K. J. Batenburg. “Foam-Like Phantoms for Comparing Tomography Algorithms”. *Journal of Synchrotron Radiation* 29.1 (Jan. 2022).

7. M. Bührer, H. Xu, A. A. Hendriksen, F. N. Büchi, J. Eller, M. Stampanoni, and F. Marone. “Deep learning based classification of dynamic processes in time-resolved X-ray tomographic microscopy”. *Scientific Reports* 11.1 (Dec. 2021).

Samenvatting

"And if I had wanted you to understand, then I would have explained it better."

"En als ik zou willen dat je het begreep, legde ik het wel beter uit. "

Johan Cruijff,
Voetbal Magazine, June 1997

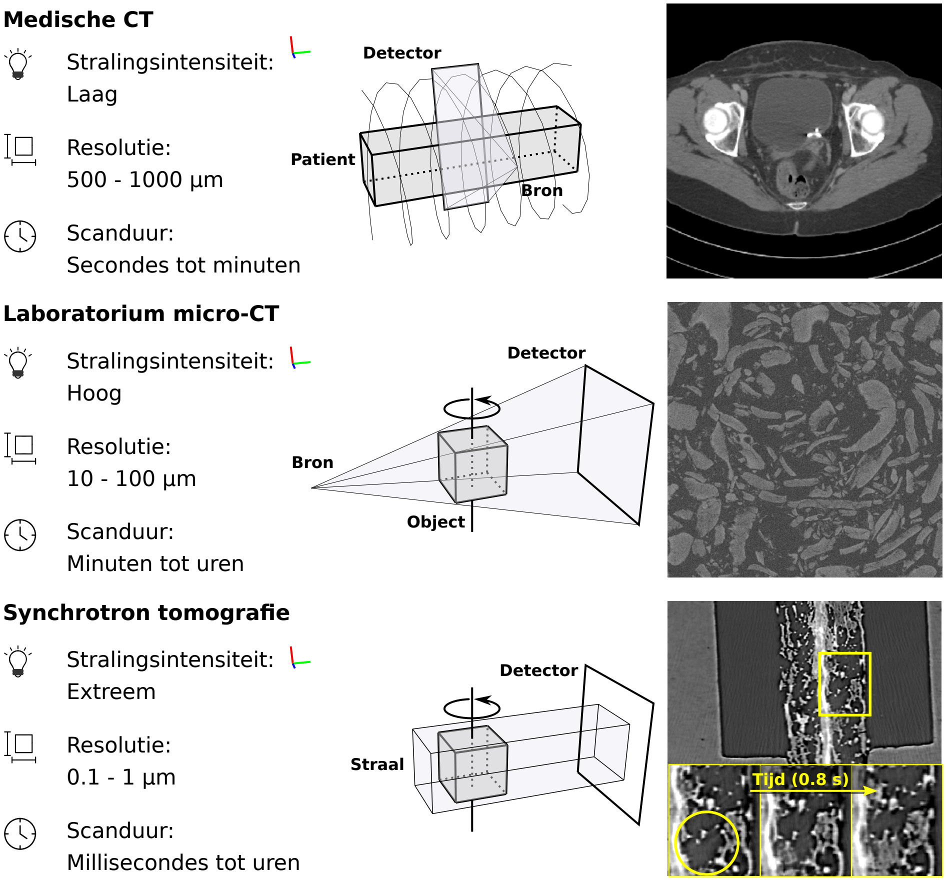
Introductie en toepassingen van computertomografie

In veel gevallen is het nuttig om een object van binnen te kunnen bekijken zonder het te hoeven openen of beschadigen. Een veel gebruikte manier om dit te bewerkstelligen is doormiddel van CT, oftewel computertomografie. CT wordt ondermeer gebruikt in ziekenhuizen om patienten te scannen en diagnoses te stellen van ziekten die van buiten niet te zien zijn. Actueel is het scannen van de longen, waarmee schade van Covid-19 aan de longblaasjes kan worden geidentificeerd.

CT wordt veel gebruikt in de industrie en de wetenschap. Deze toepassingen maken zowel gebruik van zogenaamde "micro-CT", wat meestal wordt uitgevoerd met een apparaat ter grootte van een flinke kledingkast, en van "synchrotron-tomografie", waar een deeltjesversneller met een lengte van vele honderden meters aan te pas moet komen. Met behulp van micro-CT kunnen bijvoorbeeld gesteenten gescand worden om de incidentie van bepaalde mineralen te quantificeren, of kunst-historische objecten om vragen over hun binnenkant te kunnen beantwoorden zonder het object te beschadigen. Synchrotron-tomografie is vele malen krachtiger dan micro-CT. Daardoor duurt een scan in het synchrotron korter en kunnen meer details worden ontwaard. Dit wordt bijvoorbeeld gebruikt bij het scannen van werkende brandstofcellen (zoals in Hoofdstuk 4), waar het van belang is om waterbubbels te identificeren die onststaan bij het omzetten van waterstof in water.

Computertomografie maakt gebruik van Röntgenstraling. In tegenstelling tot zichtbaar licht, heeft Röntgenstraling de neiging om zich dwars door weefsel heen te bewegen en wordt het slechts ten dele weerkaatst of opgenomen door de atomen waarlangs het beweegt. Een zogenaamde "Röntgenfoto" wordt genomen door een object tussen de stralingsbron en de detector in te plaatsen. De stralen bewegen door het object en worden opgevangen door de detector. Het contrast, en daarmee de zichtbaarheid, van de foto wordt bepaald door de mate waarin het object de Röntgenstraling opneemt. Een bot heeft bijvoorbeeld een hoge dichtheid en neemt veel straling op; lucht, daarentegen, laat bijna alle straling door.

Met één enkele Röntgenfoto kan men al veel over het binnenste van een object te weten komen. Botbreuken worden bijvoorbeeld vaak gediagnosticeerd met één foto. Echter, voor een gedetailleerd drie-dimensionaal beeld zijn meerdere foto's nodig. Hierbij wordt het object meestal op een draaitafel geplaatst en worden er meerdere (soms zelfs duizenden) foto's gemaakt. Om vervolgens van al deze foto's een accurate drie-dimensionale reconstructie van het binnenste van een object te kunnen maken, zijn algoritmes uit de computertomografie onmisbaar.



Figuur S1: Drie toepassingsgebieden van computertomografie. Medische CT wordt gebruikt om met zo min mogelijk straling een diagnose te stellen, resulterend in vrij grove beelden. Hier draaien de bron en detector in een helix om de liggende patient heen. Een doorsnede van de heup is te zien op de reconstructie rechts. Micro-CT wordt in de industrie en wetenschap gebruikt om kleinere objecten op hogere resolutie te scannen dan in medische CT. Het object ligt vaak op een draaitafel tussen de stralingsbron en detector in. Rechts is een dwardsdoorsnede van havermout te zien met een pixelresolutie van 17 µm. Synchrotron-tomografie maakt scans mogelijk met extreem veel straling en nog hogere resolutie. De stralingsbron staat zo ver weg dat de straling praktisch parallel door het object heen gaat. Rechts zijn beelden uit een “reconstructiefilmpje” van een brandstofcel te zien. De regio in het gele vierkant wordt op drie tijdstippen getoond, waardoor de vorming van een waterbubbel te zien is (aangegeven door de gele cirkel).

CT-reconstructie in gebrekkige omstandigheden en deep learning

De ontwikkeling van computertomografie-algoritmes is ver gevorderd. In ziekenhuizen wordt CT bijvoorbeeld al jaren gebruikt als diagnostisch hulpmiddel. De crux van de huidige ontwikkelingen zit hem in het ontwerpen van algoritmes die met geringe beeldinformatie complete reconstructies kunnen produceren. Een verzameling algoritmes die in recente jaren voor spectaculaire ontwikkelingen hebben gezorgd zijn zogenaamde “deep learning” algoritmes, in de volksmond ook wel “zelf-lerende” algoritmes genoemd. Zij blijken ook erg goed te werken om onvolkomenheden “weg te poetsen” uit tomografische reconstructies. Hieronder wordt beschreven welke gebreken kunnen voorkomen in Röntgenfoto’s en hoe deze onvolkomenheden met behulp van deep learning kunnen worden weggepoetst.

Binnen de computertomografie heeft een veelheid aan factoren invloed op de beeldkwaliteit. De belangrijkste factoren voor dit proefschrift worden hier besproken. In Sectie1.1.2 wordt dit onderwerp uitgebreider behandeld.

De eerste factor is ruis. Bij iedere CT-scan staat de afweging tussen het minimaliseren van de schade aan het object en het maximaliseren van de beeldkwaliteit centraal. Te veel Röntgenstraling kan een object beschadigen, maar een tekort aan straling zorgt er ook voor dat de beeldkwaliteit slechter wordt. Dit laatste heeft de lezer wellicht zelf ook opgemerkt bij het maken van een nachtelijke foto: er ontstaat ruis in het beeld omdat er te weinig licht in de sensor komt. Ruis speelt precies dezelfde rol bij het maken van een Röntgenfoto. Daarom is het ontwikkelen van reconstructie-algoritmes die de ruis kunnen onderdrukken een belangrijk onderzoeksgebied.

Een tweede factor die aan bod komt is de resolutie van de detector (de dichtheid van de pixels die het licht ontvangen). Hoe dichter deze pixels op elkaar staan, hoe kleiner de details die kunnen worden onderscheiden op de Röntgenfoto en daarmee ook de details die kunnen worden onderscheiden in de 3D reconstructie. Een uitdagend probleem is om een fijnmazigere reconstructie te maken dan de detector in eerste instantie toestaat (zogenaamde super-resolutie).

Op het gebied van zowel ruisonderdrukking als super-resolutie hebben deep learning algoritmes de laatste jaren spectaculaire ontwikkelingen laten zien. Voor beeldbewerking zijn specifiek de convolutionele neurale netwerken populair. Deze netwerken kunnen gezien worden als lege hulzen met duizenden tot tientallen miljoenen vrije parameters, die de werking van het netwerk bepalen. Deze netwerken kunnen een verscheidenheid aan vervormingen uitvoeren, waaronder ruisonderdrukking en super-resolutie.

Om de juiste parameters van een convolutioneel neuraal netwerk te bepalen, moet het eerst “getraind” worden. Dit gebeurt met het verzamelen van een stapel invoerbeelden en gewenste uitkomstbeelden. Afhankelijk van de toepassing kan het benodigde aantal voorbeelden uiteenlopen van enkele honderden tot vele miljoenen. Het netwerk wordt vervolgens toegepast op deze stapel plaatjes en de parameters worden net zo lang aangepast totdat de uitvoer van het netwerk overeenkomt met de gewenste uitkomstbeelden. Hierna kan het netwerk op nieuwe beelden worden toegepast om bijvoorbeeld ruis te onderdrukken.

Probleemstelling en samenvatting van de hoofdstukken

Een groot struikelblok bij het toepassen van deep learning algoritmes in de tomografie is het verzamelen van de vereiste voorbeeldplaatjes. Vaak is het gewenste uitkomstbeeld niet beschikbaar — als het immers wel beschikbaar was, zou er überhaupt geen reden zijn om een deep learning algoritme te gebruiken. In dit proefschrift worden technieken beschreven om het verzamelen van de vereiste voorbeeldplaatjes mogelijk te maken (in het geval van super-resolutie) en om het verzamelen van de gewenste uitkomstbeelden helemaal te omzeilen (in het geval van ruisonderdrukking). Hierdoor kunnen deep learning algoritmes getrained en toegepast worden op unieke objecten zonder eerst een bibliotheek van vergelijkbare objecten te hoeven scannen.

In **Hoofdstuk 2** wordt een techniek geintroduceerd om super-resolutie toe te passen op een enkel object met behulp van deep learning. Hierbij wordt een object op twee verschillende afstanden van een puntbron gescand. Bij de tweede afstand wordt het beeld op de detector uitvergroot. Door de foto's van de twee scans te combineren kan een reconstructie van een deelgebied van het object worden gemaakt op zowel de "normale" resolutie alsook op een hogere resolutie. Met behulp van de beelden van dit deelgebied wordt een convolutioneel neuraal netwerk getraind om de resolutie van een invoerbeeld te verhogen en zo meer details zichtbaar te maken. Vervolgens kan dit netwerk op de rest van het object worden toegepast om een reconstructie op hoge resolutie van het gehele object te verkrijgen.

We demonstreren de effectiviteit van de methode op zowel gesimuleerde als op experimentele data. De resultaten tonen aan dat de methode de resolutie van de reconstructies significant kan verbeteren.

In **Hoofdstuk 3** wordt een techniek geintroduceerd om een convolutioneel neuraal netwerk te leren om ruis te onderdrukken terwijl tijdens het trainen alleen beelden *met* ruis worden gebruikt. De wetenschappelijke bijdrage van dit hoofdstuk is tweeledig. Ten eerste wordt gedemonstreerd dat bestaande ruisonderdrukkingstechnieken niet werken als ze worden toegepast op tomografische reconstructies. We tonen aan dat dit veroorzaakt wordt door de specifieke eigenschappen van ruis in tomografische reconstructiebeelden. Ten tweede ontwikkelen we een manier om reconstructies te splitsen waardoor er invoerbeelden en gewenste uitvoerbeelden ontstaan (beide met ruis) die kunnen worden gebruikt om een neuraal netwerk te trainen. We bewijzen dat deze manier van splitsen het netwerk leert om de ruis te onderdrukken. De techniek kan worden toegepast op een scan van een enkel object: eerst wordt het netwerk op de gesplitste ruizige reconstructies getraind en vervolgens wordt het getrainde netwerk toegepast om een reconstructie zonder ruis te verkrijgen. Daarnaast kan een getraind netwerk ook op nieuwe ongeziene objecten worden toegepast.

In **Hoofdstuk 4** passen we de ruisonderdrukkingstechniek toe op scans die gemaakt zijn in verschillende synchrotrons. We beschrijven hoe de techniek kan worden aangepast om optimaal gebruik te maken van extra dimensies in de ruimte, tijd en spectrum (verstrooiing). Hierdoor wordt het mogelijk om de techniek toe te passen op tomografische "films", waarin het object een verandering ondergaat terwijl het gescand wordt. Ook demonstreren we de techniek op XRD-CT scans, waar niet de absorptie maar juist de verstrooiing van Röntgenstraling een rol speelt. Hiermee kan de mate van verstrooiing (hierboven aangeduid met spectrum) op elke locatie van het object worden bepaald. Daardoor is het mogelijk om materialen preciezer te onderscheiden.

In **Hoofdstuk 5** wordt ten slotte een softwarepakket geintroduceerd dat het maken van ingewikkelde reconstructies vergemakkelijkt. Het pakket bevat abstracties die de dynamische locatie en orientatie van de bron, detector en object beschrijven. Voorbeelden zijn de verplaatsing van het object (Hoofdstuk 2) tijdens de scan of de verstrooiing van het licht (Hoofdstuk 4). Een nuttige toevoeging is de mogelijkheid om de beschreven geometrie (locaties en orientaties) ook te visualiseren, zodat de geometrie kan worden gecontroleerd en ook kan worden getoond aan anderen (zie bijvoorbeeld Figuur S1). Ten slotte maakt het softwarepakket een nauwe integratie mogelijk tussen een bestaand tomografiesoftwarepakket en vele dataverwerkingspakketten. Hierdoor kan een groot aantal wiskundige operaties gecombineerd worden met tomografische operaties zonder dat er snelheidsverlies optreedt. We demonstreren dit op een recente techniek die gebruik maakt van verstrooiing van het licht, waarin de rekensnelheid met een factor 9 toeneemt.

Curriculum Vitae

Allard A. Hendriksen was born in Muscat, Oman. He completed his secondary education in 2009 at the Christelijk Gymnasium Sorghvliet, The Hague, The Netherlands. After his graduation, he passed the Freshman year at the California Institute of Technology (Caltech) in Los Angeles, CA, USA. He obtained his bachelor's degree in mathematics in 2015 and his master's degree (cum laude) in mathematics in 2017 from Leiden University, The Netherlands. The master's thesis with the title "Betting as an alternative to p-values" was supervised by prof. dr. P. D. Grünwald. In 2017, he started pursuing a Ph.D. degree under supervision of prof. dr. K. J. Batenburg at Centrum Wiskunde & Informatica, the national research institute for mathematics and computer science in Amsterdam, The Netherlands. He is currently employed by Lidl Nederland GmbH.

Acknowledgments

First, I would like to thank Prof. Eric Eliel for providing both firm guidance on the contents of the acknowledgements contained in this thesis, and for defining the leeway for acknowledgments that is available within the confines of what is considered appropriate in a scientific publication.

Foremost, I thank my advisors, Prof. Joost Batenburg and Dr. Daniël Pelt for the time and energy they have put in our research projects and for trusting me to pursue several projects with substantial freedom and independence. In addition, I must thank Prof. Peter Grünwald, for without his introduction to Prof. Batenburg, this thesis would never have materialized.

A special thanks to my co-authors that made this research possible at the ESRF and PSI, among which I have to mention Nicola Viganò, who tirelessly promoted my projects within the ESRF.

I thank the colleagues who I shared an office with, Zhichao Zhang, Francien Bossema, Rien Lagerwerf, and Jan-Willem Buurlage, for stimulating conversations. Close collaborations with Richard Schoonhoven and Dirk Schut made the Covid lockdowns more bearable. Many others contributed to a great research environment, among which Holger, Alex, Jordi, Maureen, Dzemila, Georgios, Henri, Sophia, Adriaan, Poulami, Vlad, Mathé, Adriaan, Max, Ben, Felix, Rob, Robert, and Tristan. I thank Willem Jan Palenstijn for answering countless questions.

I would like to thank Remco Westra, Bikkie Aldeias, Vera Sarkol, Nada Mitrovic, Duda Tepšić, Minnie Middelberg, and Adam van Arkel for providing logistical and IT support at critical times.

I would like to thank my family for their support and finally Loes for her unrelenting support and infinite patience.

A

APPENDICES

A.1 Data acquisition

X-ray tomography datasets were acquired at the TOMCAT beamline at the Swiss Light Source (SLS) at the Paul Scherrer Institut (PSI), Villigen, Switzerland. In addition, an X-ray diffraction tomography (XRD-CT) dataset was acquired at the ID15A beamline at the European Synchrotron (ESRF), Grenoble, France.

Static X-ray 3D micro-tomography The data was acquired at the TOMCAT beamline [28], and is publicly available [44]. The dataset contained 1001 projection images across an angular range of 180° measuring 1100 × 1440 square pixels of size 2.75 ¯m. The acquisition took 1 s, the exposure time was 1 ms, and the mean energy of the polychromatic beam was 30 keV.

Dynamic X-ray micro-tomography This dataset was acquired at the TOMCAT beamline [28], and is publicly available [44]. A full scan was performed every 0.1 s, resulting in a dataset of 180 time steps spanning approximately 32 seconds, divided into three 6 s chunks with 7 s pauses in between. Each time step contained 300 projection images across an angular range of 180° measuring 1100 × 1440 square pixels of size 2.75 ¯m. The mean energy of the polychromatic beam was 30 keV.

The documented angular increment was $^{\pi}/_{300}$ radians, consistent with the acquisition of 300 projection images per half rotation [44]. We observed a small deviation in this case. The angular increment was $^{\pi}/_{299.924}$, which could be determined up to five significant digits by visual inspection of the combined GridRec reconstruction of several time steps.

XRD-CT High-energy X-ray diffraction measurements were taken at the ID15A beamline using a monochromatic pencil beam (90 keV energy). Data was collected of 3 horizontal slices, spaced 7 mm apart, and acquisition of each slice took 20 minutes. Acquisition was performed in 273 translation steps over a scan range of 12 mm and in 225 rotational steps over an angular range of 180°. Sinograms were computed from the acquired images using the pyFAI library [11]. A subset of the sinograms was selected, containing 3 horizontal slices with 11 channels each. The displayed FBP reconstructions were computed with the Shepp-Logan filter using the ASTRA-toolbox [1].

A.2 Noise2Inverse training

Network and hyper-parameters On each dataset, CNN training was performed using the same network architecture and the same hyper-parameters. An open source implementation of the MS-D network was used [71, 143]. The networks were trained using the ADAM algorithm [153] with a mini-batch size of 24 and a learning rate of 10^{-3}. The networks had 100 single-channel intermediate layers, and the convolution in layer i was dilated by $d_i = 1 + (i \mod 10)$, as described in [143].

Static X-ray micro-tomography Noise2Inverse training was performed using the MS-D network architecture [143] with $54,796$ parameters. 2.5D-CNN training was performed using 10 context slices, of which 5 were below and 5 above the target slice. Reconstructions were computed using the GridRec algorithm [64, 118], resulting in a rectangular reconstruction volume of 1100 × 1440 × 433 voxels, containing the region of interest in which the fuel cell was located.

Dynamic X-ray micro-tomography Dynamic Noise2Inverse training was performed on the first 36 time steps of the experiments (3.6 seconds), in which the sample was stationary. The time steps were divided into 26 batches, each containing 6 time steps spaced one time step apart. The first batch contained time steps $1, 3, 5, 7, 9, 11$, the second batch contained time steps $2, 4, 6, 8, 10, 12$, et cetera. In this arrangement, the angles of the projection images were evenly divided over a 360° arc. Within each batch, the individual time steps served alternately as input, and the reconstruction of the combined remaining time steps served as the target. As in the static case, 2.5D-CNN training was used to supply an MS-D network with 10 context slices, and reconstruction was performed using the GridRec reconstruction algorithm.

XRD-CT Noise2Inverse training was performed using the MS-D network with 11 input channels and 11 output channels, resulting in $56,046$ parameters. Because the number of projection angles (225) was divisible by three, the sinogram was split in three parts in the angular domain. In each training iteration, two parts were used in the input reconstruction, and one part was used in the target reconstruction. We observed better performance using this scheme than by splitting the sinogram in two. During training, reconstructions were computed using FBP with the Ram-Lak filter [31] (instead of the Shepp-Logan filter) using the ASTRA-toolbox.

A.3 Synthetic noise procedure for XRD-CT

We describe the procedure for applying synthetic noise to the original XRD-CT acquisition. In addition, we describe how we estimated the virtual acquisition time using noise estimates from the reconstructed images.

Synthetic noise was applied to the acquired sinograms. The noisy value of a pixel with intensity p at scattering angle θ was determined by adding Gaussian noise

$$p_{\text{noisy}} = p + \sqrt{\frac{|p|}{\theta \cdot T}} \mathcal{N}(0, 1),$$

where $\mathcal{N}(0, 1)$ is a unit-normal Gaussian random variable, and T is a fixed constant that determines the intensity of the noise. We used values $T = 1.0, 0.1$, corresponding to moderate and high noise, respectively. The Gaussian distribution is a good approximation of the noise in practice, due to the azimuthal integration, which averages multiple Poisson-distributed pixels.

We estimated the virtual acquisition time that corresponded to the applied noise. The variance of the noise follows an inverse linear relation with respect to the acquisition time: higher noise is associated with shorter acquisition times. Therefore, we picked three uniformly constant regions of interest (ROIs) in the three reconstructed slices for which we estimated the variance of the noise. This estimation was performed on the original reconstructions and the reconstructions

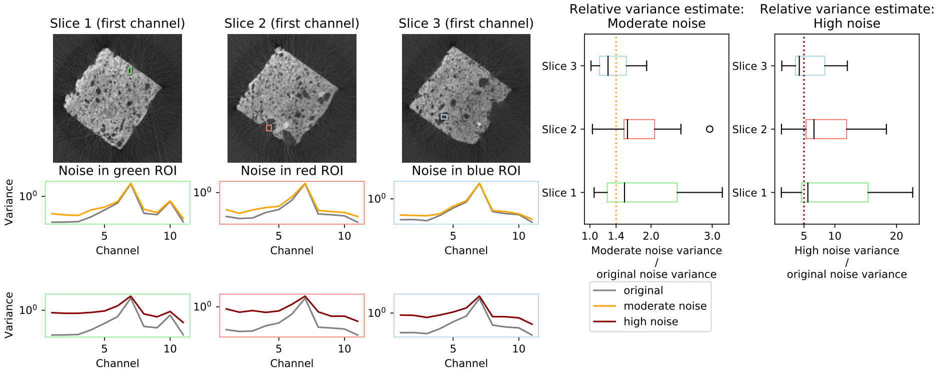


Figure A1: Estimating the noise level from reconstructed images (original, moderate synthetic noise, and high synthetic noise). In each of the three reconstructed slices, a region of interest (ROI) was chosen — indicated in green, red, and blue — that was uniformly dark in each of the 11 channels. In the left bottom panels, for each of these ROIs, the variance of the noise is plotted for the original, moderate noise, and high noise acquisition. In the right two panels, a box plot displays the uncertainty in the relative variance of the synthetic noise as it is divided by the variance in the original reconstruction. In the box plot, the vertical dotted lines show the conservative estimate of the relative variance, which was used to calculate the virtual acquisition time.

with synthetic noise. The relative variance was estimated by dividing the variances in the ROIs with synthetic noise by the variance of the ROI in the original reconstruction. A box plot of the resulting estimates is shown in Figure A1. A conservative estimate yields that the variance of the moderate noise was 1.4 times higher than the original noise, and the variance of the high noise was 5 times higher than the original noise. This corresponds to an estimated virtual acquisition time of 70% and 20% of the original acquisition time, respectively.

A.4 Total-variation minimization

The Total-Variation Minimization (TV-MIN) reconstructions were computed using the Chambolle-Pock algorithm [168], of which we used an open source implementation[1]. Because this implementation computes the full algorithm on the GPU, it is faster than comparable implementations.

On a single slice of the XRD-CT dataset (225 angles, 273 detector pixels), a reconstruction with 500 iterations takes roughly 1.7 seconds on a single GPU. The same reconstruction takes 184 seconds when computed on a single thread on the CPU with the widely used Tomopy package [64]. Assuming that work can be distributed over 32 threads, the CPU implementation would take 5.8 seconds per slice. This comparison was performed on a dual-socket system with Intel Xeon

[1] https://github.com/ahendriksen/ts_algorithms

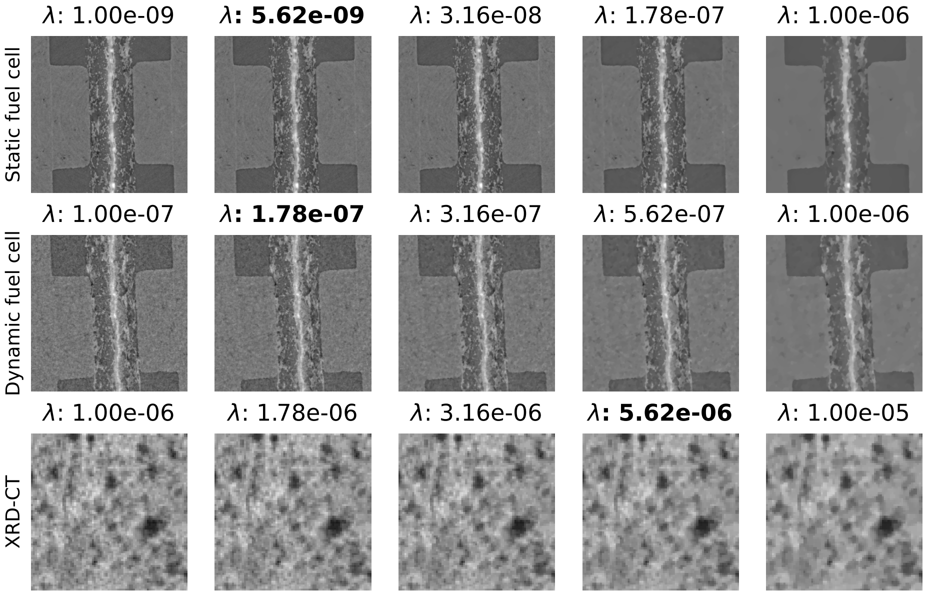


Figure A2: Total-variation minimization reconstructions with various values of the λ regularization parameter. Bold font indicates the chosen reconstruction that is used in Figure 4.7.

Silver 4110 CPUs clocked at 2.10GHz and four Nvidia GeForce GTX 1080 Ti GPUs.

Reconstructions were computed using various values of the regularization parameter λ on a regular exponential grid. These reconstructions are displayed in Figure A2. The chosen reconstruction is indicated in bold font.