Unrolled Iterative Reconstruction Networks for Implicit Motion Compensation in Dynamic Nano Computed Tomography

Dissertation

zur Erlangung des Grades der Doktorin
der Ingenieurwissenschaften (Dr.-Ing.) der Fakultät für
Mathematik und Informatik der Universität des Saarlandes

von
Alice Isabel Oberacker

Saarbrücken, 2025

UNIVERSITÄT
DES
SAARLANDES

Tag des Kolloquiums: 12.02.2026

Dekan:	Prof. Dr. Roland Speicher
Vorsitzender:	Prof. Dr. Michael Bildhauer
Berichterstatter:	Prof. Dr. Thomas Schuster
	Prof. Dr. Tatiana Bubba
Protokollführer:	Dr. Christian Steinhart

Acknowledgement

The authors gratefully acknowledge the computing time granted by the Resource Allocation Board and provided on the supercomputer Emmy/Grete at NHR-Nord@Göttingen as part of the NHR infrastructure. The calculations for this research were conducted with computing resources under the project 'Reducing Motion Artifacts in Nano-CT Imaging with a Learned RESESOP Method'.
This work also used the Scientific Compute Cluster at GWDG, the joint data center of Max Planck Society for the Advancement of Science (MPG) and University of Göttingen. In part funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 405797229.

Furthermore, I would like to sincerely thank Professor Anne Wald and my supervisor, Professor Thomas Schuster, for giving me the opportunity to explore the field of inverse problems and computed tomography. Their guidance has been instrumental in shaping my understanding of these complex topics, and I am grateful for the chance to learn and grow from this experience.

I would like to thank my colleagues, Dr. Lukas Vierus, Dean Zenner, Dr. Clemens Meiser, and Dr. Dimitri Rothermel. I greatly appreciate their generous help in understanding many of the difficult topics that I encountered and their encouragement throughout.

I am grateful to Sam Williams, Dean Zenner, Dr. Philip Oberacker, Dr. Clemens Meiser and Dr. Lukas Vierus for proofreading this thesis.

I would like to thank my parents, Ursula and Jürgen, for continually supporting me throughout my education and my brother Philip and his wife Anne for all their encouragement and support.

Last but not least, I want to thank my partner, Sam Williams, for his support, trust, and the many compromises he made so that I could focus on completing this thesis. His patience and encouragement carried me through the most challenging moments, and I am deeply grateful for the strength and stability he brought to this journey.

Declaration of original authorship

I hereby declare that this dissertation is my own original work except where otherwise indicated. All data or concepts drawn directly or indirectly from other sources have been correctly acknowledged. This dissertation has not been submitted in its present or similar form to any other academic institution either in Germany or abroad for the award of any other degree.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, November 2025

signed / gez.

Alice Isabel Oberacker

Abstract

Computed tomography (CT) has revolutionized imaging, but faces a significant challenge in dynamic scenarios where objects move during scanning. This motion, common in nano-CT due to thermal drift or in medical imaging due to patient breathing, introduces severe reconstruction artifacts. Traditional methods, such as the computationally intensive RESESOP-Kaczmarz algorithm, require precise motion knowledge and can take days to reconstruct a single image. This dissertation addresses this problem by developing novel deep learning architectures that implicitly learn to compensate for motion, building on the intersection of classical iterative reconstruction and modern machine learning. We introduce two distinct architectures: a fully learned operator network and a Radon network, both of which unroll the SESOP algorithm into a neural network structure. Evaluated on simulated dynamic nano-CT datasets, these models demonstrate superior reconstruction quality compared to state-of-the-art iterative methods, while dramatically reducing the computation time from days to seconds. This work's implications suggest a promising pathway toward real-time, high-quality CT imaging in challenging real-world applications where motion is unavoidable. By integrating the mathematical strengths of iterative regularization techniques with the adaptive capabilities of deep learning, this research makes a significant contribution to the advancement of CT imaging, making it more adaptable to a wider range of critical applications.

Zusammenfassung

Die Computertomographie (CT) hat die Bildgebung revolutioniert, steht jedoch in dynamischen Szenarien, in denen sich Objekte während des Scannens bewegen, vor einer großen Herausforderung. Diese Bewegung, die aufgrund von thermischer Drift in der Nano-CT oder aufgrund der Atmung des Patienten in der medizinischen Bildgebung häufig auftritt, führt zu starken Rekonstruktionsartefakten. Herkömmliche Methoden, wie der rechenintensive RESESOP-Kaczmarz Algorithmus, erfordern präzise Bewegungskenntnisse und können Tage dauern, um ein einzelnes Bild zu rekonstruieren. Diese Dissertation befasst sich mit diesem Problem, indem sie neuartige Deep Learning Architekturen entwickelt, die implizit lernen, Bewegungen zu kompensieren, und dabei auf der Schnittmenge zwischen klassischer iterativer Rekonstruktion und modernem maschinellem Lernen aufbauen. Wir stellen zwei unterschiedliche Architekturen vor: ein vollständig gelerntes Operatornetzwerk und ein Radon Netzwerk, die beide den SESOP Algorithmus in eine neuronale Netzwerkstruktur umsetzen. Diese Modelle wurden anhand simulierter dynamischer Nano-CT Datensätze evaluiert und weisen im Vergleich zu den derzeit modernsten iterativen Methoden eine überlegene Rekonstruktionsqualität auf, während sie gleichzeitig die Rechenzeit von Tagen auf Sekunden drastisch reduzieren. Die Ergebnisse dieser Arbeit lassen einen vielversprechenden Weg hin zu einer hochwertigen Echtzeit-CT-Bildgebung in anspruchsvollen realen Anwendungen erkennen, in denen Bewegungen unvermeidbar sind. Durch die Integration der mathematischen Stärken iterativer Regularisierungstechniken mit den adaptiven Fähigkeiten des Deep Learning leistet diese Forschung einen bedeutenden Beitrag zur Weiterentwicklung der CT-Bildgebung und macht sie anpassungsfähiger für ein breiteres Spektrum kritischer Anwendungen.

Contents

1 Introduction

Computed tomography (CT) has revolutionized medical diagnostics and industrial inspection since its inception, enabling non-invasive visualization of internal structures with unprecedented clarity. As CT technology advances toward higher resolutions, particularly in nano-CT applications where details as small as 100 nanometers can be resolved, new challenges emerge that classical reconstruction algorithms struggle to address. Among these challenges, the dynamic nature of many scanning scenarios presents a particularly difficult problem: objects move during scanning, whether due to patient breathing, thermal drift or mechanical vibrations. This dissertation explores the fundamental question: How can we accurately reconstruct CT images when the scanned object moves during data acquisition? Traditional reconstruction methods assume static objects, leading to severe artifacts when this assumption is violated. Although the iterative RESESOP-Kaczmarz, a combination of the Kaczmarz method with the Sequential Subspace Optimization (SESOP) algorithm, can handle dynamic scenarios, it requires knowledge of the movement patterns and is computationally time-consuming, taking days to reconstruct a single image. The central contribution of this work is the development of novel deep learning architectures that learn to reconstruct dynamic CT data without explicit knowledge of object motion. By unrolling the SESOP algorithm into neural network architectures, we create models that combine the mathematical rigor of iterative reconstruction with the adaptive learning capabilities of deep neural networks. Further investigation of the SESOP method in the context of machine learning was inspired by the promising results discussed in [77], where the RESESOP-Kaczmarz method was combined with post-processing models. Building on previous work in data-driven methods, this approach represents an evolution from classical methods that require precise motion estimates to data-driven approaches that implicitly learn to compensate for dynamic effects.

We present two distinct architectures: a fully learned operator (FLO) network that learns the complete reconstruction process end-to-end and a Radon network that incorporates the Radon transform within its architecture to support the reconstruction. These networks are evaluated on specially created datasets that simulate two types of motion common in nano-CT: vibrations and linear drifts. Our results demonstrate that these learned approaches can achieve an improved reconstruction

quality in comparison to state-of-the-art iterative methods, while reducing the computation time from days to seconds. The implications of this work extend beyond computational efficiency. In nano-CT applications where thermal drift and vibrations are unavoidable, in medical imaging where patient motion is inevitable, with these methods new possibilities for real-time, high-quality reconstruction can be explored. By bridging the gap between classical mathematical methods and modern machine learning, this dissertation contributes to the larger goal of making advanced CT imaging more accessible and practical for challenging real-world applications. The structure of this dissertation reflects the interdisciplinary nature of the work, combining mathematical foundations of inverse problems, principles of computed tomography, and modern deep learning techniques. Part I establishes the theoretical background, while Part II presents our methodology and demonstrates through extensive experiments that learned approaches can successfully address one of CT imaging's most persistent challenges: reconstruction in the presence of motion.

1.1 Structure of this dissertation

The structure of this dissertation is divided into two parts. The first part focuses on establishing fundamental knowledge about computed tomography and inverse problems (Chapter 2), as well as machine learning and neural networks (Chapter 3). Chapter 2 establishes the mathematical and physical foundations of CT imaging. It begins with the Radon transform and then addresses the unique challenges of nano-CT. CT reconstruction is introduced as an ill-posed inverse problem, further complicated by quantum and motion-induced noise. The chapter concludes with an exploration of various reconstruction algorithms. These range from classical methods such as filtered back projection to more specialized dynamic techniques, like RESESOP-Kaczmarz. The chapter systematically progresses from basic principles to the specific challenge of reconstructing moving objects. The Deep Learning chapter (see Chapter 3) provides the theoretical foundation required for neural networks as well as their applications to CT reconstruction. Starting with the probabilistic and statistical learning theory that underlies how networks learn from data, the chapter then introduces neural network architectures from basic fully connected networks to convolutional networks suited for image data. The training process, including optimization algorithms and regularization techniques, is detailed before existing applications of deep learning to CT reconstruction are examined. The chapter concludes in the key concept of algorithm unrolling where iterative reconstruction methods are transformed into learnable neural network architectures.

The progression from general machine learning concepts to specific techniques for inverse problems establishes the theoretical framework for transforming the SESOP algorithm into the novel architectures proposed in this dissertation.

The second part of this dissertation then focuses on the main contributions of this work. In Chapter 4, two novel neural network architectures are introduced, created by unrolling the SESOP algorithm to handle dynamic CT reconstruction. The fully learned operator (FLO) network transforms the SESOP iteration formula into a deep architecture with separate CNN blocks to process search directions and measurement data. Three variants for the FLO model are proposed that differ in how they handle the initial reconstruction step. The second model, called the Radon network, takes a hybrid approach by explicitly incorporating the Radon transform within the network structure, using mathematical knowledge as an inductive bias to reduce parameter requirements while learning to compensate for motion-induced errors.

Chapter 5 details the experimental framework for evaluating the proposed architectures, including the creation of realistic dynamic CT datasets simulating vibration and linear drift movements based on real nano-CT observations. Comprehensive configuration details for neural network training and comparison methods are provided along with computational infrastructure requirements. Different performance measures are discussed that prioritize either perceptual quality or simple pixel-wise errors.

Chapter 6 provides detailed experimental findings that the FLO architecture enhances the quality of reconstruction over current state-of-the-art iterative methods, while also reducing the computation time to mere seconds. Through systematic evaluation of architecture parameters on vibration data, validation on linear drift and low-dose datasets, and detailed comparison with methods such as RESESOP-Kaczmarz and Dremel, the results show that the learned approaches successfully handle dynamic reconstruction without requiring explicit motion models. The FLO network, in particular, shows improvements for data affected by severe motion, ultimately validating that algorithm unrolling provides a practical solution to one of nano-CT's most persistent challenges.

In Chapter 7 the findings are further discussed. The results show that the learned models, particularly the FLO model, outperform other techniques, demonstrating a high capacity for handling significant motion, although some artifacts are present. However, it was found that standard performance metrics such as SSIM can be misleading, underscoring the importance of visual evaluation. A key limitation of the learned models is their dependence on large datasets that can be challenging

to produce. Despite this, their fast inference time post-training makes them a compelling option, especially for applications requiring many reconstructions.

Finally, in Chapter 8 we present the conclusions of this research and outline potential directions for future work.

1.2 Notation

The notation used is presented in Table 1.1.

Symbol	**Meaning**
$\mathcal{A} : X \to Y$	Forward operator
$\mathcal{A}^{-1} : Y \to X$	Inverse operator
$\mathcal{A}^* : Y \to X$	Adjoint operator
$\mathcal{A}^+ : \text{ran}(\mathcal{A}) \oplus \text{ran}(\mathcal{A})^\perp \to X$	Pseudo-inverse operator
$\mathcal{R}$	Radon transform
FOV	Field of view
K	Number of scanning positions
L	Number of detector points
$\mathcal{K} \coloneqq \{0, \dots K-1\}$	The index set of scan angles
$\mathcal{L} \coloneqq \{0, \dots L-1\}$	The index set of detector points
φ	Angle of scanner in parallel beam geometry
s	Distance of X-ray to center point in parallel beam geometry
α	Angle between X-rays in fan beam geometry
β	Angle of scanner in fan beam geometry
δ	noise level of measurement data
η	noise level of inexact operator
$\text{ran}(A)$	Range of A
$\mathcal{L}(X, Y)$	Space of linear, continuous mappings $X \to Y$
$\mathcal{N}(\mathcal{A})$	Null space of $\mathcal{A} : X \to Y : \mathcal{N}(\mathcal{A}) = \{x \in X : \mathcal{A}x = 0\}$
$M^\perp$	Orthogonal complement of M
Θ	Parameter space
$\theta \in \Theta$	Parameterization
$\mathcal{F}_\theta$	Neural Network

Tab. 1.1.: Notation overview

Part I

Background

2 Computerized Tomography

Computerized tomography (CT), a revolutionary medical imaging procedure, has transformed the landscape of diagnostic radiology since its inception. Using variations in X-ray absorption, CT constructs a comprehensive cross-sectional view of the body, unveiling details with remarkable clarity that were once obscured by conventional radiography [25]. This non-intrusive approach yields a three-dimensional representation of the body's structures, facilitating the precise identification and analysis of medical conditions.
Several discoveries and inventions contributed to the process we know today, starting with the discovery of X-radiation by Röntgen in 1895 [99]. This was followed by Radon's formulation of the integral transform in 1917, the now-called *Radon transform* [93], which serves as the mathematical foundation for image reconstruction. Finally, the 1960s marked the development of CT, which is attributed to the innovative work of Sir Godfrey Hounsfield [56] and Dr. Allan Cormack [31]. Their contributions to the field were honored with the Nobel Prize in Physiology or Medicine in 1979 [25]. The pioneering efforts by these scientists have paved the way for the rapid progression of CT technology, which has seen significant advancements in image resolution, speed, and reduction of radiation exposure.

Modern medicine incorporates a variety of imaging methods, such as magnetic resonance imaging (MRI) and positron emission tomography (PET). Due to its longer scan time and the fact that MRI cannot be applied to metallic objects, that is, implants or pacemakers [109], it is not an appropriate solution for emergency medical care or material testing. PET scans help to display the metabolic and biochemical functions of tissues and organs by using a radioactive substance, called a tracer. Especially in oncology, PET and CT technologies are combined, to provide even more detailed imaging of cancerous tissues [108]. Therefore, computed tomography plays a pivotal role in present-day medicine, acting as an essential diagnostic tool in several medical fields, such as oncology, vascular imaging, orthopedics, and emergency care[25]. Its relevance is not limited to the healthcare industry, as it also finds applications in the industrial sector for the non-invasive analysis and inspection of materials. For example, the work referenced in [36] explores the non-destructive evaluation of materials such as steel turbine blades to avoid early failures that could result in increased costs and shutdowns of power plants. Furthermore, in the field

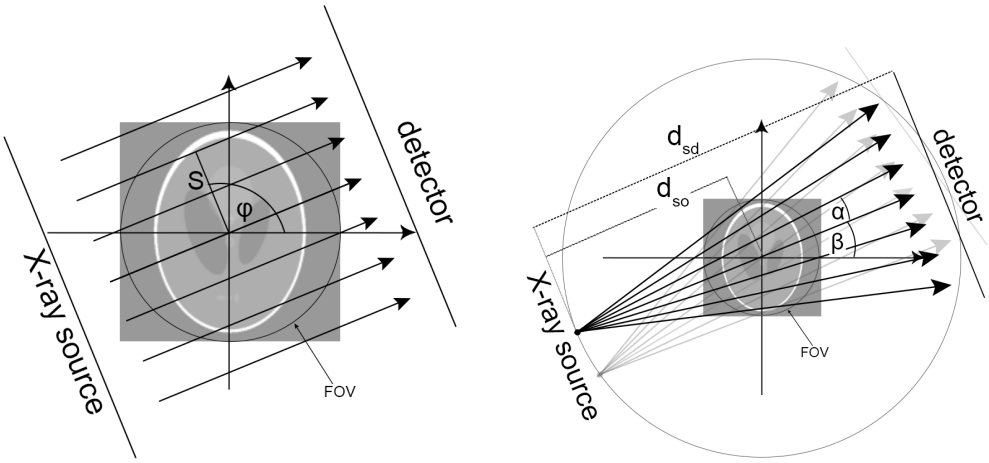
of paleontology, CT is a valuable method for examining ancient mummification processes, as discussed in [55].
This chapter will investigate the essential principles of computed tomography, discuss its wide-ranging applications in the industrial sector, and explore mathematical methods that contribute to its success.

2.1 CT Principles and the Radon Transform

As stated above, computed tomography utilizes radiation. In this section, we present two 2-dimensional scanning methods along with a concise overview of how radiation is converted into measurement data. The following is based on the respective sections in [85] and [96].
When scanning an object, the device emits a series of X-rays from the source of the scanner, shown as arrows in Figure 2.1a. The X-ray source then rotates around the subject, or the subject itself is rotated so that the X-rays pass through the medium Ω at different angles. For each beam, the intensity of the X-ray is measured by the detector. Repeating this process for all beams at a specific scan angle is called a *projection*. Projections encompassing all angles are gathered into what is referred to as a *sinogram*. For two-dimensional imaging, the scanner rotates in a single plane around the object, providing slices of the anatomy. In this case, the sinogram is a 2-d image where each pixel represents the intensity of a single X-ray collected on a detector pixel. In contrast, for three-dimensional images, the scanner rotates along multiple planes, allowing for a comprehensive three-dimensional view [85].

Various geometrical configurations are utilized to arrange multiple X-rays to examine an object. In Figure 2.1 two types of CT geometries for two-dimensional images are visualized: parallel beam and fan beam geometry.

The scanned object lies within the field of view (FOV), which is defined by the unit circle. In *parallel beam geometry* (see Figure 2.1a), the path of each beam is precisely characterized by specifying the angle φ formed with the x-axis and the distance s from the beam to the center. Because each X-ray is generated individually, the X-ray source must be incrementally shifted to construct a comprehensive beam array, enabling a complete examination of the object. Additionally, the X-ray source must rotate around the object. Due to the symmetry of the system, it is appropriate to confine the scan angles to the range of $[0°, 180°]$. In fan beam geometry, an X-ray source produces a set of beams arranged in a fan-like configuration. The angular separation between these beams is defined by the angle α. The orientation of the



(a) Parallel beam **(b)** Fan beam

Fig. 2.1.: Shepp Logan phantom with two geometry types: Figure (a) shows a parallel beam geometry with a parallel array of X-rays. Figure (b) shows a fan beam geometry with a single point that emits a fan of X-rays.

X-ray source's rotation is characterized by the angle β, which is measured from the central beam to the x-axis. The fan beam configuration causes a magnification effect, making the distances from the X-ray source to both the detector and the object crucial, denoted d_{sd} and d_{so}, respectively.

At the source the X-ray has an intensity of I_0, the detectors measure the residual intensity of the rays, I_1. Different parts of the object have different *absorption coefficients* and therefore absorb and reflect the photons of the X-rays to varying degrees, this is called *attenuation*. Dense tissues, such as bone, absorb more X-rays, resulting in fewer photons reaching the detectors. Conversely, less dense tissues, such as muscles or organs, absorb fewer X-rays, allowing more to reach the detectors. The same holds for inanimate objects, where, for example, cracks or pockets of impurities in the material have densities different from those of the surrounding material.
This section demonstrates the concepts of two-dimensional CT geometries and attenuation. Here, we present a mathematical framework that models the forward model of CT scans.

The absorption coefficient can be defined as a function $f : \mathbb{R}^2 \to \mathbb{R}$, where $f(x)$ represents the absorption coefficient at $x \in \Omega \subset \mathbb{R}^2$. We assume that f is square integrable, that is, $f \in L^2(\Omega)$ and sufficiently smooth. For a more detailed discussion, see [84]. Furthermore, it is assumed that $\mathrm{supp} f \subset \Omega$ and therefore $f = 0$ in $\mathbb{R}^2 \backslash \overline{\Omega}$.

Let $L \subset \mathbb{R}^2$ be the line representing an X-ray and $I_L(x)$ be the intensity in $x \in L$.

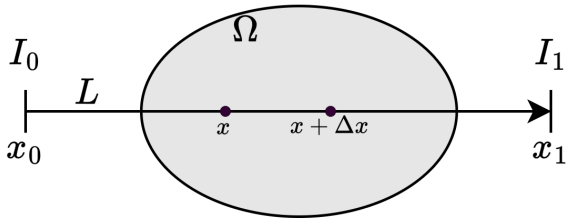


Fig. 2.2.: Attenuation on single X-ray.

The lines $L_{\omega,s}$ can be parameterized with angle $\varphi \in [0, 2\pi]$ and detector position $s \in [-1, 1]$ that signify the signed distance of L from the origin. The parameterization reads

$$L_{\omega,s} := \{x \in \mathbb{R}^2 : \langle x, \omega \rangle_2 = s\}, \qquad L_{\omega,s}(t) := s\,\omega(\varphi) + t\,\omega(\varphi)^{\perp} \tag{2.1}$$

for unit vectors

$$\omega := \omega(\varphi) := \begin{pmatrix} \cos(\varphi) \\ \sin(\varphi) \end{pmatrix} \in S^1 \qquad \omega^{\perp} := \omega(\varphi)^{\perp} := \begin{pmatrix} -\sin(\varphi) \\ \cos(\varphi) \end{pmatrix} \in S^1,$$

where $S^1 := \{x \in \mathbb{R}^2 : ||x||_2 = 1\}$ be the unit circle in $\mathbb{R}^2$ [96].

Figure 2.3 shows the parameterization of $L_{\omega,s}$.

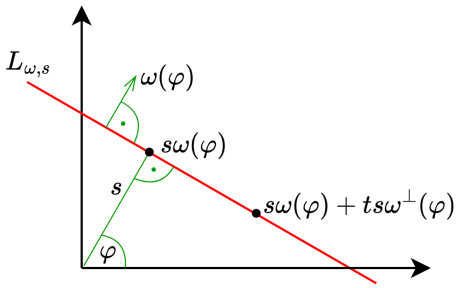


Fig. 2.3.: Line $L_{\omega,s}$ with parameters $\omega(\varphi)$ as the normal vector with line angle φ and s as the signed distance to the origin.

The decrease in X-ray intensity is directly related to both the traveled distance $||\Delta x||$ and the function $I_L(x)$:

$$I_L(x + \Delta x) = I_L(x) - f(x)||\Delta x||I_L(x). \tag{2.2}$$

Let $\gamma : \mathbb{R} \to \mathbb{R}^2$ be a parameterization of L with $\gamma(t) = s\,\omega + t\,\omega^{\perp}$. Then for parameter values τ and $\tau + \Delta\tau$, with $0 < \tau \in \mathbb{R}$ it holds that

$$x = \gamma(\tau), \quad x + \Delta x = \gamma(\tau + \Delta\tau) \tag{2.3}$$

and

$$\begin{aligned}\|\Delta x\| &= \|\gamma(\tau+\Delta\tau)-x\| \\ &= \|s\omega+(\tau+\Delta\tau)\omega^{\perp}-(s\omega+\tau\omega^{\perp})\| \\ &= \|\Delta\tau\omega^{\perp}\| = \Delta\tau.\end{aligned}$$

Expressing Equation 2.2 using $\gamma(x+\Delta x)$ we arrive at

$$\begin{aligned} & I_L(\gamma(\tau+\Delta\tau)) && = I_L(\gamma(\tau)) - f(\gamma(\tau))\ \Delta\tau\ I_L(\gamma(\tau)) \\ \Leftrightarrow\ & \frac{I_L(\gamma(\tau+\Delta\tau)) - I_L(\gamma(\tau))}{\Delta\tau} && = -\ f(\gamma(\tau))\ I_L(\gamma(\tau)).\end{aligned}$$

Letting $\Delta\tau \to 0$ yields the derivative of the intensity

$$I'_L(\gamma(\tau)) = -I_L(\gamma(\tau))f(\gamma(\tau))$$

and

$$-\frac{I'_L(\gamma(\tau))}{I_L(\gamma(\tau))} = f(\gamma(\tau)).$$

Integrating the traveling path of the beam L with the starting points $x_0 = \gamma(\tau_0)$ and the end point $x_1 = \gamma(\tau_1)$ we get the following:

$$\begin{aligned}\int_{\tau_0}^{\tau_1} f(\gamma(\tau))d\tau &= -\int_{\tau_0}^{\tau_1} \frac{I'_L(\gamma(\tau))}{I_L(\gamma(\tau))}d\tau \\ &= -\ln(I_L(\gamma(\tau))\Big|_{\tau_0}^{\tau_1} \\ &= -\ln(I_L(\tau_1)) + \ln(I_L(\tau_0)) \\ &= \ln\left(\frac{I_0}{I_1}\right),\end{aligned}$$

which can be calculated from the measured intensities at the detector [96].

Therefore, with measurements I_0 and I_1 we can calculate

$$\int_{L_{\omega,s}} f(x)dx = \int_{\mathbb{R}} f(s\omega+t\omega^{\perp})dt. \tag{2.4}$$

Definition 2.1 (2-d Radon transform). *For $f : \mathbb{R}^2 \to \mathbb{R}$ the two-dimensional* Radon transform *is given by*

$$(\mathcal{R}f)(\omega, s) := \int_{L_{\omega,s}} f(x)dx, \tag{2.5}$$

where L is a line parameterized by the normal vector $\omega(\varphi) := (\cos\varphi, \sin\varphi)^T \in S^1$, for all $\varphi \in [0, 2\pi]$ and $s \in \mathbb{R}$.

Given I_0 and I_1, we can then use Definition 2.1 to recover $f(x)$ for $x \in \Omega$, assuming that $||\omega^\perp|| = 1$ and $f = 0$ in $\mathbb{R}^2 \backslash \overline{\Omega}$.

The Radon transform can be defined accordingly for the geometry of the fan beam by adjusting the parameterization of L.

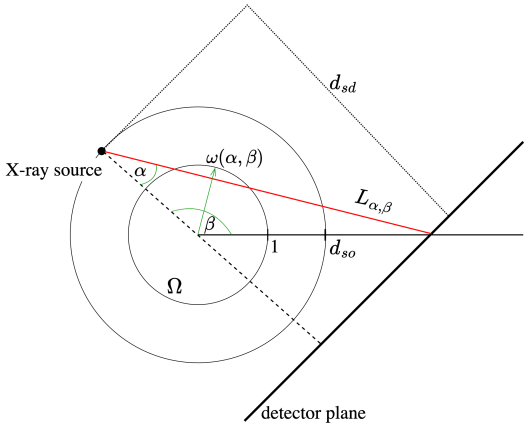


Fig. 2.4.: Parametrization of line L in fan beam geometry. β is the scanner angle, shown with a dotted line going through the origin, α is the angle between the X-ray and the origin line of the source. ω is the normal vector of L, d_{sd} is the distance between source and detector and d_{so} is the distance between source and origin.

In fan beam geometry, the X-ray source rotates around the object with angle $\beta \in [0, 2\pi]$ measured from the x-axis to a line perpendicular to the detector plane. The array of X-rays is determined by the angle $\alpha \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ of this line. Therefore, the line L, which describes the beam, can be parameterized by α and β.

The following equations show how to translate the parameterization of L from fan to parallel beam:

$$\begin{aligned} \sin(\alpha) = \frac{s}{d_{so}} \quad &\Leftrightarrow \quad s = d_{so} \cdot \sin(\alpha), \\ \alpha + \beta = \varphi + \frac{\pi}{2} \quad &\Leftrightarrow \quad \varphi = \alpha + \beta - \frac{\pi}{2}. \end{aligned}$$

The fan beam Radon transform then reads

$$(\mathcal{R}_{FB}f)(\alpha,\beta) := (\mathcal{R}f)\left(\omega(\alpha+\beta-\frac{\pi}{2}), d_{so}\cdot\sin(\alpha)\right), \tag{2.6}$$

for $d_{so} > 1$, $\alpha \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $\beta \in [0, 2\pi]$.

2.1.1 Semi-Discrete and Discrete CT Model

For the numerical representation of a CT scan, it is necessary to convert the continuous Radon transform into its discrete version.

As presented in the pioneering work of Frank Natterer [84], we continue to discuss the two-dimensional case here. To transform Radon's integral equation (see (2.5)) into a set of linear equations, discretization of the scanner area is required. For the 2-dimensional space, this is defined as a bounded unit circle

$$\mathcal{B}^2 := \overline{\mathcal{B}_1(0)} \subset \mathbb{R}^2. \tag{2.7}$$

As a first step, we discretize the lines $L_{\omega,s}$. For ω and the corresponding scan angles, we define a set Φ of angles in degrees or radians, that is, $\Phi := \{\varphi_0, \dots \varphi_{K-1}\} \subset [0, \pi]$. $\mathcal{K} := \{0, \dots K-1\}$ defines the index set of these scan angles where K is the number of angles. Here, we use equidistant angles

$$\varphi_k = \frac{180^\circ}{K}\cdot k, \text{ for } k \in \mathcal{K} \tag{2.8}$$

and the corresponding unit vector as $\omega_k := \omega_k(\varphi_k)$.

Likewise, the detector points are also discretized and defined as $\mathcal{Q} := \{s_0, \dots s_{L-1}\} \subset [-1, \dots, 1]$, where $\mathcal{L} := \{0, \dots L-1\}$ is the corresponding index set with a total of L detector points. For $s_l \in \mathcal{Q}$, we write

$$s_l = \frac{2}{L}\cdot l - 1. \tag{2.9}$$

We define a new index, which iterates through Φ and $\mathcal{Q}$:

$$j(k,l) := k\cdot L + l, \text{ for } k \in \mathcal{K}, l \in \mathcal{L}. \tag{2.10}$$

Thus, with $J \coloneqq K \cdot L$ this index function includes all possible combinations of $k \in \mathcal{K}, l \in \mathcal{L}$ and $0 \leqslant j \coloneqq j(k,l) \leqslant J-1 = K \cdot L - 1$.
This allows us to write the Radon transform from (2.5) as a semi-discrete equation

$$\mathcal{R}_j f = g_j,$$

with

$$\mathcal{R}_j f = \mathcal{R}_{j(k,l)} f \coloneqq (\mathcal{R}f)(\omega(\varphi_k), s_l)$$

as in Definition 2.1.

Now, we also want to discretize f to formalize the fully discrete CT model, which will be of importance for the implementation, specifically for machine learning methods.

Next, the area in $\mathcal{B}^2$ is decomposed into pixels $S_m, m = 0, \dots M-1$, such that

$$\mathcal{B}^2 \subset \bigcup_{m=0}^{M-1} S_m.$$

Figure 2.5 visualizes this discretization with $M = 25$. Let

$$f^{[M]}(x) \coloneqq \sum_{m=0}^{M-1} F_m \chi_{S_m}(x), \tag{2.11}$$

where $x \in \mathcal{B}^2$ and $F_m \in \mathbb{R}$ for all $m = 0, \dots, M-1$ and

$$\chi_{S_m}(x) = \begin{cases} 1 & \text{for } x \in S_m \\ 0 & \text{for } x \notin S_m. \end{cases}$$

For each line L_j and pixel S_m we can compute $a_{j,m} =$ Length of $(L_j \cap S_m)$ and

$$\int_{L_j} f^{[M]}(x)dx = \sum_{m=0}^{M-1} F_m a_{j,m}.$$

The matrix $F = (F_0, \dots F_{M-1}) \in \mathbb{R}^M$ must be calculated so that $f^{[M]}$ is a good approximation of f. For a good resolution M must be chosen large enough. For $a_j \coloneqq (a_{j,0}, \dots, a_{j,M-1})^T \in \mathbb{R}^M$ and F, it follows that

$$\langle a_j, F \rangle = g_j, \quad j = 0, \dots, J-1. \tag{2.12}$$

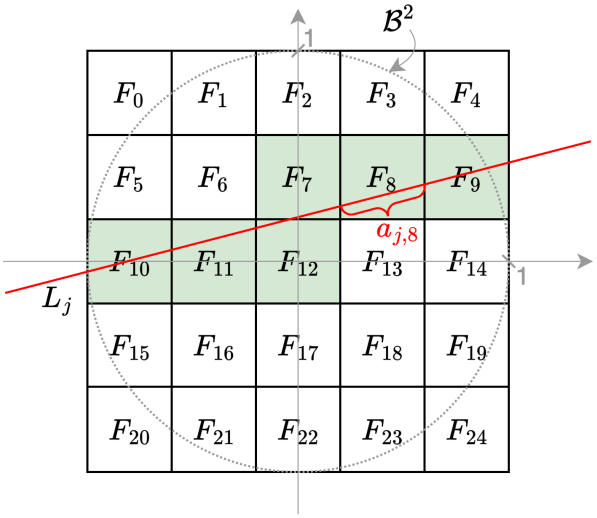


Fig. 2.5.: Discretizations of f and $L_{\omega,s}$. $\mathcal{B}^2$ is decomposed into $M = 25$ pixels. Line L_j cuts through pixels S_m with $m = \{7, 8, 9, 10, 11, 12\}$. For each of those pixels holds $a_{j,m} > 0$.

Each line L_j intersects only a fraction of pixels S_m, thus a_j is sparse for all $j = 0, \ldots, J-1$ (see Figure 2.5). With $A := (a_{j,m})_{j=0,\ldots J-1, m=0,\ldots M-1}$ and $g = (g_0, \ldots, g_{J-1})$, the system of equations can be written as $AF = g$.

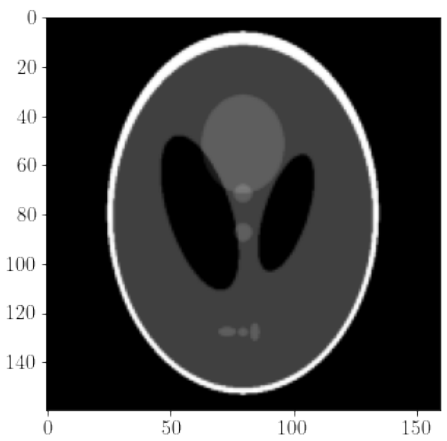
Definition 2.2 (2-d discrete Radon transform)**.** *The two-dimensional* discrete Radon transform *is given by*

$$(\mathcal{R}_d f)(\omega_k, s_l) := \sum_{m=0}^{M-1} F_m a_{j,m}, \tag{2.13}$$

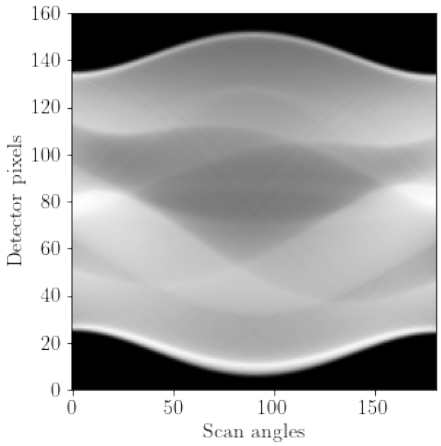
with $j := j(k, l)$.

Applying $\mathcal{R}_d$ to an object f for all $j \in \{0, \ldots, J-1\}$ is equivalent to applying the matrix A to F: $\mathcal{R}_d f \equiv AF$.

Figure 2.6 shows the Radon transform applied to the standard Shepp-Logan Phantom. Each column of the sinogram contains the attenuation values for all detector points.



(a) Shepp-Logan Phantom



(b) Sinogram of Shepp Logan phantom with 180 scan angles and 160 detector pixels. Generated using Python's skimage package [120].

Fig. 2.6.: Example of Radon transform generating the forward projection of Shepp-Logan phantom.

2.2 Nano-CT

Medical computerized tomography is a well-established imaging technique optimized for human anatomy. There are several differences from industrial CT.
In medical contexts, minimizing radiation exposure is critical. However, in industrial applications, this is less of a concern. Here, denser materials can be scanned with increased X-ray doses to create more detailed images. Unlike medical CT, where the X-ray source and detector move around the stationary patient, industrial CT maintains fixed positions for both components, with the object rotating to achieve scan angles. This fixed setup allows flexible adjustment of the image resolution for objects of different sizes [89, 118]. Medical CT typically achieves resolutions ranging from 70 to 1000 micrometers [72, 89, 118], while industrial CT operates within 5 to 150 micrometers. Recent advances in nano-CT have pushed detail resolution to an impressive 100 nanometers [78].

Nano-CT is an interesting field with many applications in material testing. For example, the development and testing of microchips [81, 7]; the detection and analysis of counterfeit semiconductor structures [10]; and electrode degradation studies in batteries [81, 124, 127]. Another field that benefits from micro- and nano-CT is life sciences, where, for example, cell growth is examined on polymer scaffolds, making it possible to study the influence of the scaffold architecture on cell location and morphology [17].

Three main approaches for reaching resolutions on the nanoscale with X-ray tomography are:

- Optics-based X-ray microscope, which can achieve a resolution of 35-100nm [111], but is limited to low energies, narrowing their application in material science [81].
- X-ray microscopy based on scanning electron microscopy (SEM). Through SEM it is possible to generate an X-ray beam fine enough to reach a resolution of 100 nm in 3D imaging [40]. When using SEM, the object must be conductive and because it is placed in a vacuum chamber it must also be resistant to vacuum [81].
- This is not necessary when using microscopy with a nanofocus X-ray source. Through the development of a specialized nanofocus X-ray source, consistent high-resolution imaging is possible at the submicrometer level [44]. This has been integrated in the ntCT nanotomography system and can reach a resolution of 150 nm [44, 10].

Up to this point, we have discussed the creation of sinograms with CT scanners. The essential task that follows is to convert these sinograms into visual representations of the object being scanned, called *reconstruction*. Computerized tomography exemplifies a class of mathematical challenges known as *inverse problems*. The subsequent sections provide an explanation of *inverse problems* and present methodologies to address the challenges of reconstruction.

2.3 Inverse Problems

For hundreds of years, scientists have striven to create representations of the natural world. Specifically, when trying to understand the behavior of physical systems, researchers formulate mathematical models that aim to accurately describe these systems.
Such models take input parameters and compute an output, i.e. the effect of the given cause. This is called a *forward problem*. A relevant example is the change in the ice volume in Greenland and its influence on the rise of sea level. Climate scientists use measurable factors such as temperature in the ice and the rate of change in surface elevation to predict the rise of sea level [86].
The reverse direction is then called an *inverse problem*: scientists collect data on the rise in sea level (the effect) and attempt to conclude which factors (the cause) lead

to the observed changes.

To describe inverse problems in simpler terms, we are going to discuss an everyday scenario.
Imagine Emily's grandmother baking an incredible chocolate cake for family occasions, and it is her delight to keep the recipe secret. Emily has always had an interest in baking and has been keeping notes on Grandma's cakes for years. Through trial and error, she has been developing her version of Grandma's recipe.
To phrase this in terms of an inverse problem: the ingredients are the *cause*, the recipe represents the *model* and the cake gives us the *effect*.

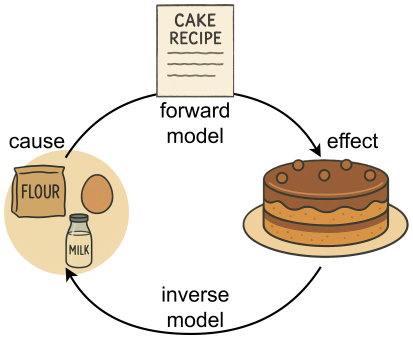


Fig. 2.7.: Image parts created using Microsoft Copilot.

Grandma simply solves the *forward problem* because she can buy the ingredients and has the recipe. Emily, however, has to solve the *inverse problem*, by analyzing the cakes and, through rigorous testing, comes up with her own recipe and determines the correct ingredients.

Returning to the application of CT, which is a linear inverse problem: Obtaining measurement data, i.e. the sinogram, through scanning an object represents the forward problem; conversely, reconstructing the object's cross-sectional view, i.e. the phantom, is termed the inverse problem. The subsequent sections introduce mathematical concepts for illustrating inverse problems, discussing their inherent challenges and possible resolution methods.

Let $(X, ||\cdot||_X)$ and $(Y, ||\cdot||_Y)$ be Hilbert spaces with their corresponding norms that serve as parameter and data spaces, respectively. Then the *forward model* is defined as

$$\mathcal{A} : X \to Y,$$

mapping the relationship between cause, X and effect, Y.

2.3.1 Linear Inverse Problems and their Problems

As we have seen, a forward model consists of three parts: the effect or observed data $g \in Y$, the cause or requested data $f \in X$, and the model or operator $\mathcal{A} : X \to Y$, where $\mathcal{A}f = g$.

An important property of the problem $(\mathcal{A}, X, Y)$ is its *well-posedness*. Hadamard [46] defined it as follows.

Definition 2.3 (Well-posedness)**.** *Let $\mathcal{A} : X \to Y$ be a forward operator with Hilbert spaces X and Y, the problem $\mathcal{A}f = g$ is well-posed if it satisfies the following conditions:*

- *There exists a solution f.*
- *That solution is unique.*
- *The inverse operator $\mathcal{A}^{-1}$ is continuous, implying the stability of $\mathcal{A}^{-1}$ under minor data variations.*

Therefore, an inverse problem is deemed ill-posed if any of these properties are not met. Most often, stability of the inverse operator cannot be guaranteed.

Let us revisit the example of Grandma's cake recipe. No cake is always the exact replica of the previous one. Sometimes the ingredients differ, or sometimes Grandma modifies the recipe. Maybe the eggs are smaller or the sugar is finer; maybe Grandma did not have enough butter, so she added oil to the dough. All these adjustments represent errors in Emily's data collection. This means that Emily's recipe (her model $\mathcal{A}$) needs to be stable under small changes in the cake (her data g).

The focus of this research is computerized tomography. Here, the perturbations, or noise, can originate from several sources: one caused by the probabilistic nature of X-ray photons and one caused by movements between object and scanner. The latter will be discussed in the next section.
The number of photons that arrive at each pixel of the detector varies randomly. Therefore, even without an object to be scanned, the X-ray radiation process is inflicted with noise, the so-called *quantum noise* [24].
Let the data affected by quantum noise be defined as

$$g^\delta := g + \xi, \quad \xi \in Y,$$

with noise level δ, where

$$\|g - g^\delta\|_Y = \|\xi\|_Y \leq \delta \in \mathbb{R}_{>0}. \tag{2.14}$$

The noise in the measurement data appears as graininess in the image and is attributed to the statistical principles underlying photon generation in X-ray tomography and their attenuation, which manifest regardless of the object being scanned [24].
This noise is affected by various factors in the scanning parameters, including the source voltage, the duration of the exposure, and the dimensions of the pixels [97]. Other types of noise originate from the object itself; this includes, for example, object placement, geometry, and *movement*. To accurately reconstruct f, it is imperative to account for this noise.

Definition 2.4 (Inverse Problem)**.** *Let $(X, ||\cdot||_X)$ and $(Y, ||\cdot||_Y)$ be Hilbert spaces and $\mathcal{A}$ be a linear bounded operator between X and Y. Let g^δ be a perturbed measurement in Y. The inverse problem then delivers the quantity of interest $f \in X$, which solves*

$$\mathcal{A}f = g,$$

given noisy data g^δ with $\|g - g^\delta\|_Y \leqslant \delta$.

If the inverse operator $\mathcal{A}^{-1}$ is not continuous, the problem is ill-posed. Data variations in g^δ can cause the noise in the reconstruction to be amplified

$$\|f - \mathcal{A}^{-1}g^\delta\|_X \gg 0. \tag{2.15}$$

In this case, or when $\mathcal{A}$ is not invertible, it is possible to substitute the inverse of $\mathcal{A}$ with its *pseudo-inverse* $\mathcal{A}^+ : \text{ran}(\mathcal{A}) \oplus \text{ran}(\mathcal{A})^\perp \to X$ [84]. This pseudo-inverse operator maps every $g \in \text{ran}(\mathcal{A}) \oplus \text{ran}(\mathcal{A})^\perp$ to the unique element f^+ with minimal norm. Then $f^+ = A^+g$ is a solution of

$$\mathcal{A}^*\mathcal{A}f^+ = \mathcal{A}^*g, \tag{2.16}$$

where $\mathcal{A}^*$ is the *adjoint operator*, which satisfies

$$\langle \mathcal{A}^*y, x\rangle_X \;=\; \langle y, \mathcal{A}x\rangle_Y, \forall x \in X, y \in Y. \tag{2.17}$$

Equation (2.16) is called *normal equation*. Its derivation will be discussed in the context of the *Landweber* method in Section 2.4.1.

However, the operator $\mathcal{A}^+$ is only continuous if the image of $\mathcal{A}$ is closed, that is, $\text{ran}(\mathcal{A}) = \overline{\text{ran}(\mathcal{A})}$ [84]. If this is not satisfied, the pseudo-inverse does not guarantee a suitable approximation to f^+, if only noisy data g^δ are available. In this case regularization methods as explained in Section 2.3.2 can be applied to guarantee stability with respect to data noise.

2.3.2 Regularization of inverse problems

Let $\mathcal{A}f = g$ be a linear inverse problem (see (2.4)), where only noisy data g^δ with a noise level $\delta > 0$ (see (2.14)) are available. If the pseudo-inverse operator $\mathcal{A}^+$ is not continuous, the solution $f^\delta = \mathcal{A}^+ g^\delta$ is typically meaningless due to the large error in the reconstruction; see (2.15). In this case, $\mathcal{A}^+$ can be approximated by a family of continuous operators $\{R_t\}_{t>0}$. This is called *regularization* of ill-posed problems. An element in $\{R_t g^\delta\}_{t>0}$ needs to be found that approximates the true solution $f^+ = \mathcal{A}^+ g$ appropriately.

Definition 2.5 (Regularization)**.** *Let $\mathcal{A} \in \mathcal{L}(X, Y)$ and let $\{R_t\}_{t>0}$ be a family of continuous (possibly non-linear) functions $Y \to X$ with $R_t 0 = 0$. The pair $(\{R_t\}_{t>0}, \gamma)$ is called a* regularization *if a function $\gamma : \mathbb{R}_{>0} \times Y \to \mathbb{R}_{>0}$ exists so that for every $g \in ran(\mathcal{A})$*

$$\lim_{\delta \to 0} \sup\{\|\mathcal{A}^+ g - R_{\gamma(\delta, g^\delta)} g^\delta\|_X : g^\delta \in Y, \|g - g^\delta\|_Y \leqslant \delta\} = 0. \tag{2.18}$$

The value $\gamma(\delta, g^\delta)$ is called a regularization parameter*. If γ depends only on δ it is called an* a priori parameter choice *otherwise it is called an* a posteriori parameter choice.

The reconstruction error $\|\mathcal{A}^+ g - R_t g^\delta\|_X$ can be estimated by

$$\|\mathcal{A}^+ g - R_t g^\delta\|_X \leqslant \underbrace{\|\mathcal{A}^+ g - R_t g\|_X}_{\text{approximation error}} + \underbrace{\|R_t(g - g^\delta)\|_X}_{\text{data error}}.$$

Typically, the approximation error tends towards 0 with $t \to 0$. However, in that case, the data error explodes if the noise $g - g^\delta$ is not in the domain of $\mathcal{A}^+$. With $t \to \infty$, the data error approaches 0, but the approximation error explodes. Thus, an optimal regularization parameter $t_{\text{opt}} \approx \gamma(\delta, g^\delta)$ balances both errors to find the minimal reconstruction error.

When an a priori parameter choice is not possible, for example, because the choice depends on information from f^+ that is not available, the *discrepancy principle* can be applied as an a posteriori choice [80]. The idea is that a regularization parameter $\gamma(\delta, g^\delta)$ can be chosen so that

$$\|\mathcal{A} f^\delta_{\gamma(\delta,g^\delta)} - g^\delta\|_Y \approx \delta.$$

The discrepancy of $f^\delta_{\gamma(\delta,g^\delta)}$, that is, the residual, has the same magnitude as the data noise.

Definition 2.6 (Discrepancy principle [80])**.** *Let $\{t_k\}_{k\in\mathbb{N}}$ be a strictly monotonically decreasing null sequence and $\tau > 1$ fixed. Choose k^* so that*

$$\|\mathcal{A} f^\delta_{t_{k^*}} - g^\delta\|_Y \leqslant \tau\delta < \|\mathcal{A} f^\delta_{t_i} - g^\delta\|_Y$$

for $i = 1, \dots k^* - 1$. Choose $\gamma(\delta, g^\delta) \coloneqq t_{k^*}$.

The discrepancy principle finds application in several iterative reconstruction methods discussed in Section 2.4. Specific regularization methods are the content of Section 2.4.

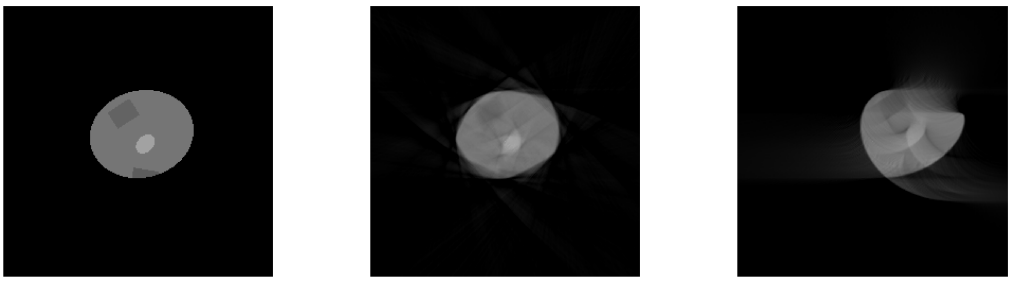
2.3.3 Dynamic inverse problems

During a CT scan, each projection is recorded at a different time; thus, there exists a one-to-one relationship between the angles φ of the tomograph and the times at which the projections are taken. In this sense, the angle φ can be interpreted as a temporal variable. This is particularly relevant if the object is moving in time. For the sake of simplification in modeling, it is often assumed that the target quantity f is static and therefore does not vary over time. However, this assumption does not hold in numerous cases where the subject of investigation may change throughout the data collection process. In medical CT, it is evident that a patient's body undergoes various transformations during a scan: rhythmic deformations due to respiration and cardiac activity, as well as other organ movements. Unfortunately, scanning of inanimate objects is not immune to the effects of motion either. It is also evident in non-destructive testing contexts, such as tracking the propagation of liquid fronts [16] or performing elasticity evaluations to determine material characteristics during scanning [61].

Movement can also be caused by fluctuations in ambient conditions, by the heat generated by the device, called *thermal drift*, or by vibrations caused by the CT

scanner itself or its surroundings. These drifts cause a misalignment between the X-ray projections and the object [87, 74].
Several approaches have been investigated to solve this problem. The authors in [63, 47, 94, 57] provide solutions that estimate or correct the motion of the object from the measurement data. Other research has been conducted to measure the motion of the object during the scanning process [22, 28]. Applications in nano-CT are heavily affected by this dynamic component. Due to the small size of the scanned objects, the signal-to-noise ratio is reduced correspondingly, thereby increasing the impact of noise on the final image quality.

The dynamic nature of such data conflicts with the static model $\mathcal{A}$. Consequently, employing a conventional inversion technique on dynamic measurements leads to motion artifacts within the reconstructed images, thereby diminishing the quality of the reconstruction. Subsequently, this will be referred to as *temporal noise* or *dynamic noise*. In Section 5.1.1, we present two datasets that have been generated for the purpose of this dissertation. In Figure 2.8 the reconstructions of two samples from these datasets by *filtered back projection* (FBP) [84] are shown. The artifacts resulting from the motion of the object are clearly visible, as the FBP is not equipped to deal with dynamic noise.



(a) (b) (c)

Fig. 2.8.: The left image shows an unperturbed phantom, the one in the middle is the FBP reconstruction of this phantom where a vibration movement was introduced. The right image shows the FBP reconstruction of a phantom drifting from the center of the field of view to the right.

Therefore, it is crucial to develop specialized strategies that accommodate the dynamic characteristics of the subject to effectively address these dynamic inverse problems. Several of these methods will be discussed in Section 2.4.

This temporal aspect can be included in the conventional model $\mathcal{A}f = g$ by refining the inverse problem to be time dependent; see [14]. As mentioned above, the time

spent rotating the subject would reflect this time component. The equation can then be written as

$$(\mathcal{A}f)(t) = g(t), \quad \text{for all } t \in [0, T].$$

There are two potential approaches for reconstructions. The first involves reconstructing the moving object, thus $f(t)$ for all time steps $t \in [0, T]$. This approach leads to $g(t)$ being sparse, since only a single projection angle is available at any time step t. The second approach involves reconstructing a fixed state of f at the time instance $t = 0$. In this scenario, all projection angles are used; however, in general, all measurement data $g(t \neq 0)$ are inconsistent with the reconstruction object $f(t = 0)$, therefore $g(t \neq 0)$ corresponds to a perturbed version of $f(t = 0)$ [23].

We consider inverse problems exclusively for CT applications. The searched-for quantity f is contained in the field of view, here defined as $\mathcal{B}^d$, a bounded subset of $\mathbb{R}^d$ with $d = \{2, 3\}$, see (2.7).
Blanke et al. [14] describe the nature of the behavior of the object in time as a *deformation model* $\Gamma : [0, T] \times \mathcal{B}^d \to \mathcal{B}^d$. Then Γ is a function that, for each time step $t \in [0, T]$, maps a location in $x \in \mathcal{B}^d$ to its deformed location $\hat{x} \in \mathcal{B}^d$. Thus, Γ introduces a time component to f, with

$$(f \circ \Gamma)(t, x) = f(\hat{x}),$$

where $\hat{x} = \Gamma(t, x)$, $t \in [0, T]$ and $x, \hat{x} \in \mathcal{B}^d$. Note that $\hat{x} \in \mathcal{B}^d$, as it is assumed that $\operatorname{supp}(f \circ \Gamma(t, \cdot)) \subset \mathcal{B}^d$.

Based on this deformation, the exact forward model can be defined as

$$(\mathcal{A}_\Gamma f)(t) := \mathcal{A}^{static}(f \circ \Gamma)(t) = g(t), \tag{2.19}$$

where $\mathcal{A}^{static}$ denotes the static case of the forward operator, that is, the standard Radon transform, and $\mathcal{A}_\Gamma$ includes the unknown deformation model Γ.

This definition assumes that complete knowledge of $\mathcal{A}_\Gamma$ is available. However, in practical scenarios, accurately computing the deformation model Γ or the respective operator $\mathcal{A}_\Gamma$ is a significant challenge. There are different solutions to solve this issue. Motion can, for example, be estimated by tracking markers [47]. Other methods reconstruct both the function f and the motion [6, 22, 13], which is particularly difficult when $g(t)$ is sparse [18]. When the deformation model is not available, reconstruction methods must recognize that the forward operator is, at best, an approximation and inherently imprecise.
This approximation is called the *inexact forward operator* and is labeled $\mathcal{A}^\eta$. For $\mathcal{A}^\eta \approx \mathcal{A}_\Gamma$ the parameter η indicates a modeling error, similar to the noise level δ of the

measured data (see 2.14). This error is also known as *global model inexactness* [14] and is estimated by

$$\|\mathcal{A} - \mathcal{A}^\eta\| \leqslant \eta \in [0, \infty). \tag{2.20}$$

Let $f \in \overline{\mathcal{B}_\rho(0)} \subset X$, with $\overline{\mathcal{B}_\rho(0)}$ being the closed ball with radius ρ around 0. The norm of f is then bounded by ρ and it follows that

$$\|\mathcal{A}f - \mathcal{A}^\eta f\| = \|(\mathcal{A} - \mathcal{A}^\eta)f\| \leqslant \|\mathcal{A} - \mathcal{A}^\eta\|\|f\| \leqslant \eta\rho. \tag{2.21}$$

A foundational method for static CT applications is the Radon transform (see (2.5)). However, the Radon transform does not incorporate a deformation model and is therefore an inexact operator when applied to dynamic CT. Thus, $\mathcal{R} = \mathcal{A}^\eta$ and the Radon transform can therefore be viewed as an approximation of the operator including the deformation model. Consequently, we can write $\mathcal{R} \approx \mathcal{A}_\Gamma$. For given measurement data g, the static operator can only provide an approximation. Thus,

$$\mathcal{A}^{static} f \approx g$$

and as g is perturbed, it generally holds that

$$g \notin \text{ran}(\mathcal{A}^{static}).$$

For the purpose of this research, the focus is on reconstructing the object at a single time point from measurement data based on the dynamic object. To avoid confusion, we distinguish between noise-free measurement data g from a static object; measurement data inflicted with quantum noise, resulting in g^δ; and data created from a dynamic object with an unknown deformation model, labeled g^η. This notation should highlight that g^η is inconsistent with the static Radon transform. The data generated for this investigation will consist exclusively of motion-related data.

The subsequent section will present a variety of reconstruction techniques. Among these methods, some specifically address the issue of operator inexactness.

2.4 Reconstruction methods

This section presents the fundamental concepts of various reconstruction algorithms, each of which is applicable to CT reconstruction tasks. Nevertheless, it should be

noted that not all algorithms presented are particularly tailored for the reconstruction of dynamic inverse problems.

2.4.1 Sequential subspace optimization – SESOP

Before introducing the *sequential subspace optimization (SESOP)* we introduce the closely related *Landweber method* [67]. SESOP generalizes and accelerates the fundamental principles behind the Landweber iteration.

The Landweber method This algorithm is an iterative method to solve the minimization problem $\min_{f\in X} \Psi(f)$ for the functional $\Psi : X \to \mathbb{R}$ with

$$\Psi(f) \coloneqq \frac{1}{2}\|\mathcal{A}f - g\|_Y^2. \tag{2.22}$$

In the following, the iteration formula of the Landweber method is derived.
For $\lambda \in \mathbb{R}$ and for all $\varphi \in X$ we get

$$\Psi(f+\lambda\varphi) = \frac{1}{2}\|\mathcal{A}(f+\lambda\varphi) - g\|_Y^2 \tag{2.23}$$

$$= \frac{1}{2}\langle \mathcal{A}f - g, \mathcal{A}f - g\rangle + \lambda\langle \mathcal{A}f - g, \mathcal{A}\varphi\rangle + \frac{\lambda^2}{2}\langle \mathcal{A}\varphi, \mathcal{A}\varphi\rangle \tag{2.24}$$

Calculating the total derivative of Ψ at f in the direction φ by using the Gâteaux differential

$$\begin{aligned} D_\varphi\Psi(f) &= \lim_{\lambda\to 0} \frac{\Psi(f+\lambda\varphi) - \Psi(f)}{\lambda} \\ &= \lim_{\lambda\to 0} \frac{\lambda\langle \mathcal{A}f - g, \mathcal{A}\varphi\rangle + \frac{\lambda^2}{2}\langle \mathcal{A}\varphi, \mathcal{A}\varphi\rangle}{\lambda} \\ &= \lim_{\lambda\to 0} \langle \mathcal{A}f - g, \mathcal{A}\varphi\rangle + \frac{\lambda}{2}\langle \mathcal{A}\varphi, \mathcal{A}\varphi\rangle \\ &= \langle \mathcal{A}f - g, \mathcal{A}\varphi\rangle \end{aligned}$$

and then applying the adjoint operator $\mathcal{A}^*$ (2.17) leads to

$$\begin{aligned} D_\varphi\Psi(f) &= \langle \mathcal{A}f - g, \mathcal{A}\varphi\rangle \\ &= \langle \mathcal{A}^*(\mathcal{A}f - g), \varphi\rangle. \end{aligned}$$

Due to the definition of the directional derivative

$$D_\varphi\Psi(f) = \langle \nabla\Psi(f), \varphi\rangle \quad \text{for all } \varphi \in X$$

we can derive that

$$\langle \mathcal{A}^*(\mathcal{A}f - g), \varphi \rangle = \langle \nabla\Psi(f), \varphi \rangle \quad \text{for all } \varphi \in X$$

and thus

$$\nabla\Psi(f) = \mathcal{A}^*(\mathcal{A}f - g).$$

The *normal equation* is then found by setting $\nabla\Psi(f) = 0$.

The Landweber method corresponds to the gradient descent for the functional Ψ. The iteration step of this gradient descent reads as

$$\begin{aligned} f_{n+1} &= f_n - \omega\nabla\Psi(f_n) \\ &= f_n - \omega\mathcal{A}^*(\mathcal{A}f_n - g), \end{aligned} \tag{2.25}$$

where $\omega > 0$ is a relaxation parameter. In the case of noisy measurement data, this iteration is performed until a stop criterion is satisfied. A more detailed explanation of the gradient descent algorithm can be found in Section 3.3.2.

SESOP is a method that solves large-scale smooth unconstrained problems [83]. In contrast to conventional gradient descent techniques, such as Landweber method, which usually adhere to the steepest descent path dictated by one gradient direction, SESOP utilizes gradient information along multiple search directions simultaneously, which improves convergence.
First, several key concepts required by SESOP will be introduced. The following overview is based on [14, 119, 77].

Let $(X, ||\cdot||_X)$ and $(Y, ||\cdot||_Y)$ be Hilbert spaces with their respective norms. Let $\mathcal{A} : X \to Y$ be a bounded linear operator with $\mathcal{A}f = g$, for $f \in X$ and $g \in Y$. $\mathcal{A}^*$ represents the adjoint operator of $\mathcal{A}$. Let $\langle\cdot,\cdot\rangle_X$ and $\langle\cdot,\cdot\rangle_Y$ be the inner products in X and Y, respectively. For improved readability of the notation, we shall omit specifying the respective domains for the inner product and the norm.

The *solution set* for the equation $\mathcal{A}f = g$ is defined as

$$\mathcal{M}_{\mathcal{A},g} \coloneqq \{f \in X : \mathcal{A}f = g\}.$$

Then we can define several important sets in X:
For $u \in X\backslash\{0\}$, $\alpha \in \mathbb{R}$

$$\mathcal{H}(u, \alpha) \coloneqq \{f \in X : \langle u, f \rangle = \alpha\}$$

is called a *hyperplane* in X.

A *stripe* is given by

$$\mathcal{H}(u, \alpha, \xi) \coloneqq \{f \in X : |\langle u, f\rangle - \alpha| \leq \xi\}, \tag{2.26}$$

with $\xi \in \mathbb{R}, \xi > 0$, whose boundary is given by $\mathcal{H}(u, \alpha + \xi)$ and $\mathcal{H}(u, \alpha - \xi)$.
For every $x \in \mathcal{M}_{\mathcal{A},g}$ and any $w \in Y$ we get

$$\langle \mathcal{A}^* w, x\rangle = \langle w, \mathcal{A}x\rangle = \langle w, g\rangle.$$

Therefore, we know that for any $w \in Y$, the hyperplane

$$\mathcal{H}(\mathcal{A}^* w, \langle w, g\rangle) = \{f \in X : \langle \mathcal{A}^* w, f\rangle = \langle w, g\rangle\}$$

contains the solution set $\mathcal{M}_{\mathcal{A},g}$.

SESOP is an iterative method that can apply several search directions in each iteration step. The iteration is defined as in (2.27).

Definition 2.7 (Sequential Subspace Optimization (SESOP) [83])**.** *The iteration step of SESOP reads*

$$f_{n+1} = f_n - \sum_{i \in I_n} t_{n,i} \mathcal{A}^* w_{n,i}, \tag{2.27}$$

where f_n is the current iterate, I_n a finite index set, $w_{n,i} \in Y$ for all $i \in I_n$. $n \in \mathbb{N}$ is the iteration index *and $t_{n,i} \in \mathbb{R}$. $\mathcal{A}^* w_{n,i}$ is called a* search directions*.*

The sequence $t_n \coloneqq (t_{n,i})_{i \in I_n}$ minimizes the convex function

$$h_n(t) \coloneqq \frac{1}{2}\left\| f_n - \sum_{i \in I_n} t_i \mathcal{A}^* w_{n,i} \right\|_X^2 + \sum_{i \in I_n} t_i \langle w_{n,i}, g\rangle. \tag{2.28}$$

If $\mathcal{A}^* w_{n,i}$ are linearly independent, h is strictly convex, and thus t_n is unique.

Minimizing h_n is equivalent to calculating the metric projection

$$f_{n+1} = \mathcal{P}_{\mathcal{H}_n}(f_n)$$

of the current iterate f_n onto $\mathcal{H}_n$ [106], which we define as the intersection of hyperplanes

$$\mathcal{H}_n \coloneqq \bigcap_{i \in I_n} \mathcal{H}(\mathcal{A}^* w_{n,i}, \langle w_{n,i}, g\rangle),$$

for $w_{n,i} \in Y$.

We already know that

$$\mathcal{M}_{\mathcal{A},g} \subseteq \mathcal{H}(\mathcal{A}^* w_{n,i}, \langle w_{n,i}, g \rangle).$$

Therefore, the iteration approximates a solution $f \in \mathcal{M}_{\mathcal{A},g}$ by projecting sequentially onto intersections of these hyperplanes. In Figure 2.9 this sequential projection is visualized for $X = \mathbb{R}^2$ with three hyperplanes.

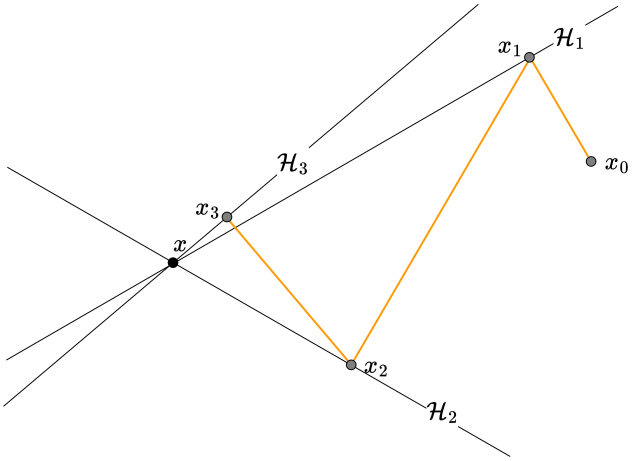


Fig. 2.9.: For $X = \mathbb{R}^2$, projections of x_0 onto hyperplanes $\mathcal{H}_i$, $i \in \{1, 2, 3\}$.

In Hilbert spaces, the metric projection of $f \in X$ onto $\mathcal{H}(u, \alpha)$ can be expressed as

$$\mathcal{P}_{\mathcal{H}(u,\alpha)}(f) = f - \frac{\langle u, g \rangle - \alpha}{||u||^2} u.$$

Algorithm 2.1 (SESOP). *Let $f_0 \in X$. For each $n \in \mathbb{N}$, define a finite index set I_n and search directions $\mathcal{A}^* w_{n,i}$ with $w_{n,i} \in Y$ and $i \in I_n$. The next iterate f_{n+1} is computed as the metric projection*

$$f_{n+1} \coloneqq \mathcal{P}_{\mathcal{H}_n}(f_n)$$

of the previous iterate f_n, onto the intersection of hyperplanes

$$\mathcal{H}_n := \bigcap_{i \in I_n} \mathcal{H}(\mathcal{A}^* w_{n,i}, \langle w_{n,i}, g \rangle).$$

In the presence of noisy data denoted by g^δ and an inexact forward operator $\mathcal{A}^\eta$ SESOP can also be applied as a regularization technique.

The SESOP iteration is regularized by projecting onto stripes instead of hyperplanes whilst including a stop criterion providing a finite stop index n^*, where the corresponding iterate f_{n^*} represents the regularized solution of $\mathcal{A}^\eta f = g$ [105, 119].

The stripes $\mathcal{H}(u, a, \xi)$ are chosen so that ξ depends on the noise level, while the solution set $\mathcal{M}_{\mathcal{A},g}$ is still contained in the stripe.
Let us define the stripe

$$\mathcal{H}^{\eta,\delta,\rho} := \mathcal{H}\left((\mathcal{A}^\eta)^* w, \langle w, g^\delta\rangle, (\delta + \eta\rho)\|w\|\right) \tag{2.29}$$

$$= \{f \in X : \|\langle (\mathcal{A}^\eta)^* w, f\rangle - \langle w, g^\delta\rangle\| \leqslant (\delta + \eta\rho)\|w\|\}, \tag{2.30}$$

where $\rho > 0$, see (2.21). With

$$\mathcal{M}^\rho_{\mathcal{A},g} := \mathcal{M}_{\mathcal{A},g} \cap B_\rho(0) \tag{2.31}$$

we can show that $\mathcal{M}^\rho_{\mathcal{A},g} \subset \mathcal{H}^{\eta,\delta,\rho}$, since for all $z \in \mathcal{M}^\rho_{\mathcal{A},g}$ we have

$$\begin{aligned}
\left\|\langle (\mathcal{A}^\eta)^* w, z\rangle - \langle w, g^\delta\rangle\right\| &= \left\|\langle w, (\mathcal{A}^\eta - \mathcal{A})\, z\rangle + \langle w, \mathcal{A}z\rangle - \langle w, g^\delta\rangle\right\| \\
&= \left\|\left\langle w, (\mathcal{A}^\eta - \mathcal{A})\, z + g - g^\delta\right\rangle\right\| \\
&\leqslant \|w\| \cdot \left(\|\mathcal{A}^\eta - \mathcal{A}\|\, \|z\| + \left\|g - g^\delta\right\|\right) \\
&\leqslant (\eta\rho + \delta)\|w\|.
\end{aligned} \tag{2.32}$$

This also provides the discrepancy principle. For all $z \in \mathcal{M}^\rho_{\mathcal{A},g}$

$$\|\mathcal{A}^\eta z - g^\delta\| = \|(\mathcal{A}^\eta - \mathcal{A})z + \mathcal{A}z - g^\delta\| \leqslant \tau(\eta\rho + \delta),$$

where $\tau > 1$ to guarantee that the error can fall below the noise level. Through the relation of the discrepancy principle to errors η and δ it is ensured that the algorithm remains stable and converges to a meaningful approximation of the true solution. It has been shown in [105, 107] that the discrepancy principle produces a finite stopping index n^*, such that

$$\|\mathcal{A}^\eta f_{n^*} - g^\delta\| \leqslant \tau(\eta\rho + \delta)$$

with $\tau > 1$.

We can now write the regularized SESOP algorithm as follows.

Algorithm 2.2 (RESESOP [14]). *Let $f_0 \in X$. For each $n \in \mathbb{N}$, define a finite index set I_n and search directions $(\mathcal{A}^\eta)^* w_{n,i}$ with $w_{n,i} \in Y$ and $i \in I_n$.*
If the discrepancy principle

$$\|\mathcal{A}^\eta f_n - g^\delta\| \leqslant \tau(\eta\rho + \delta) \tag{2.33}$$

with $\tau > 1$ is satisfied, we can stop iterating. Otherwise, the next iterate f_{n+1} is computed as the metric projection

$$f_{n+1} \coloneqq \mathcal{P}_{H_n^{\eta,\delta,\rho}}(f_n)$$

of the previous iterate f_n , where

$$\mathcal{H}_n^{\eta,\delta,\rho} \coloneqq \bigcap_{i \in I_n} \mathcal{H}\left(u_{n,i}^{\eta,\delta}, \alpha_{n,i}^{\eta,\delta}, \xi_{n,i}^{\eta,\delta,\rho}\right),$$

with

$$\begin{aligned} u_{n,i}^{\eta,\delta} &\coloneqq (\mathcal{A}^\eta)^*\, w_{n,i}^{\eta,\delta}, \\ \alpha_{n,i}^{\eta,\delta} &\coloneqq \langle w_{n,i}^{\eta,\delta}, g^\delta \rangle, \\ \xi_{n,i}^{\eta,\delta,\rho} &\coloneqq (\delta + \eta\rho)\|w_{n,i}^{\eta,\delta}\|. \end{aligned}$$

A good choice for the search directions $(\mathcal{A}^\eta)^* w_{n,i}$ is to define them similarly to the Landweber method as the gradient of the functional Ψ defined in (2.22). Therefore, we set $w_{n,i} \coloneqq w_i \coloneqq \mathcal{A}^\eta f_i - g^\delta$, and thus

$$(\mathcal{A}^\eta)^* w_{n,i} = (\mathcal{A}^\eta)^*(\mathcal{A}^\eta f_i - g^\delta). \tag{2.34}$$

The algorithm with two search directions has been explored by [119, 105] and is sketched for $X = \mathbb{R}^2$ in Figure 2.10.

For stripes

$$\mathcal{H}_i = \mathcal{H}((\mathcal{A}^\eta)^* w_i, \langle w_i, g^\delta \rangle, (\delta + \eta\rho)\|w_i\|), \text{ with } i \in \{1, 2\}$$

and $x_0 \in \mathcal{H}_2$, let x_1 be the metric projection of x_0 onto the intersection of $\mathcal{H}_1$ and $\mathcal{H}_2$ calculated by

$$x_1 = \mathcal{P}_{\mathcal{H}_1 \cap \mathcal{H}_2}(x_0).$$

The metric projection of x_0 onto the intersection of two stripes can be calculated by a maximum of two metric projections onto the bounding hyperplanes or their

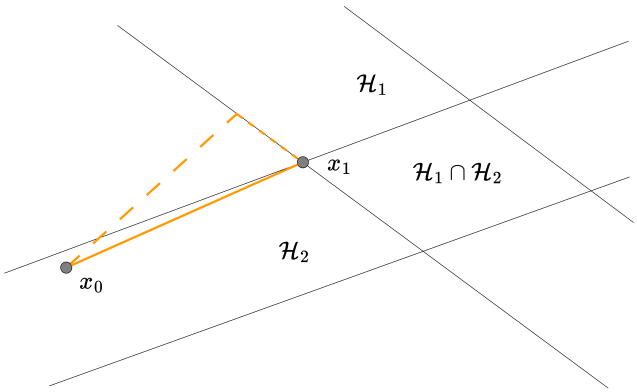


Fig. 2.10.: For $X = \mathbb{R}^2$ and $\mathcal{H}_1$ and $\mathcal{H}_2$ stripes and $x_0 \in \mathcal{H}_1$. The metric projection $x_1 = \mathcal{P}_{\mathcal{H}_1 \cap \mathcal{H}_2}(x_0)$ projects x_0 onto the intersection of stripes $\mathcal{H}_1 \cap \mathcal{H}_2$.

intersections [14, 119]. To this end, x_0 is first projected onto one of the bounding hyperplanes of $\mathcal{H}_1$. Thus,

$$\mathcal{P}_{\mathcal{H}_1}(x_0) \equiv \mathcal{P}_{\mathcal{H}((\mathcal{A}^\eta)^* w_1, \langle w_1, g^\delta \rangle \pm (\delta + \eta\rho)\|w_1\|)}(x_0) = \hat{x}_1$$

Then, $\hat{x}_1$ is projected onto the intersection of the bounding hyperplane of $\mathcal{H}_2$ and $\mathcal{H}_1$ by

$$x_1 = \mathcal{P}_{\mathcal{H}_1 \cap \mathcal{H}_2}(\hat{x}_1).$$

The convergence of RESESOP has been discussed in [106, 105, 107].

2.4.2 Kaczmarz method

An iterative algorithm for solving semi-discrete linear problems of the form $\mathcal{A}_j f = g_j$ is the *Kaczmarz method*. We continue to use the same notation as in the previous section and reuse some of the definitions.
Similarly to SESOP, this method also applies metric projections sequentially to iterates f_i for $i \in \{0, 1, \dots\}$. The Kaczmarz method in the context of fully discrete CT is also known as the *algebraic reconstruction technique (ART)*. This section follows again Natterer's book on "The Mathematics of Computerized Tomography" [84].
We revisit the notation introduced in (2.11), where $f^{[M]}$ is an approximation of f using the matrix F with M pixels. The Kaczmarz method computes the approximation $f^{[M]}$, by computing F as a point in the intersection of hyperplanes

$$\mathcal{H}_j := \{F \in \mathbb{R}^M : \langle a_j, F \rangle = g_j\}, \text{ for } j = 0, \dots J-1.$$

Analogously to SESOP this is done via sequential metric projections; here, however, the projections are dependent on sub-problems with $a_j F = g_j$, $j = 0, \dots J-1$. Let $\mathcal{P}_{\mathcal{H}_j} : \mathbb{R}^M \to \mathcal{H}_j \subset \mathbb{R}^M$ be the metric projection on $\mathcal{H}_j$. For $F \in \mathbb{R}^M$, we get

$$\mathcal{P}_{\mathcal{H}_j}(F) = F + t\, a_j,$$

for a $t \in \mathbb{R}$. The value t is calculated as follows

$$\langle \mathcal{P}_{\mathcal{H}_j}(F), a_j \rangle = \langle F + t\, a_j, a_j \rangle = \langle F, a_j \rangle + t\, \langle a_j, a_j \rangle = g_j,$$

which is, as long as $a_j \neq 0$, equivalent to

$$t = \frac{g_j - \langle F, a_j \rangle}{\|a_j\|^2}.$$

Therefore, we can write the projection as

$$\mathcal{P}_{\mathcal{H}_j}(F) = F + \frac{g_j - \langle F, a_j \rangle}{\|a_j\|^2}\, a_j.$$

Let $\omega > 0$ be a relaxation parameter and

$$\mathcal{P}^\omega := \mathcal{P}^\omega_{\mathcal{H}_N} \cdot \mathcal{P}^\omega_{\mathcal{H}_{N-1}} \cdot \ldots \cdot \mathcal{P}^\omega_{\mathcal{H}_1}.$$

with

$$\mathcal{P}^\omega_{\mathcal{H}_j} := (1-\omega)\, I + \omega\, \mathcal{P}_{\mathcal{H}_j}$$

be the sequential projection onto all hyperplanes $\mathcal{H}_j$ for $j = 0, \dots, J-1$.

Using these projections, the *Kaczmarz* algorithm can be written as follows.

Algorithm 2.3 (Kaczmarz [84]). *Let $F^{(0)} \in \mathbb{R}^M$. For each $n \in \mathbb{N}$, the next iterate $F^{(n+1)}$ is computed as the metric projection*

$$F^{(n+1)} := \mathcal{P}^\omega(F^{(n)}), \tag{2.35}$$

of the previous iterate F_n, where

$$\mathcal{P}^\omega := \mathcal{P}^\omega_{\mathcal{H}_N} \cdot \mathcal{P}^\omega_{\mathcal{H}_{N-1}} \cdot \ldots \cdot \mathcal{P}^\omega_{\mathcal{H}_1},$$

with $\omega > 0$,

$$\mathcal{P}^\omega_{\mathcal{H}_j} := (1-\omega)\, I + \omega\, \mathcal{P}_{\mathcal{H}_j}$$

and hyperplanes

$$\mathcal{H}_j := \{F \in \mathbb{R}^M : \langle a_j, F\rangle = g_j\}, \text{ for } j = 0, \dots J-1.$$

The algorithm with its explicit formulation of the metric projection can be found in Algorithm 1 and a visualization of the projections is provided in Figure 2.11. The Kaczmarz iteration can also be formalized for semi-discrete problems with F

Algorithm 1 Kaczmarz pseudocode

Initialize: $F^{(0)} \in \mathbb{R}^M$, $\omega > 0$
1: **for** n = 0, 1,... **do**
2: $\quad F_0^{(n)} \leftarrow F^{(n)}$
3: $\quad$ **for** j = 0, ...J-1 **do**
4: $\quad\quad F_{j+1}^{(n)} \leftarrow F_j^{(n)} + \omega \frac{g_j - \langle F_j^{(n)}, a_j\rangle}{\|a_j\|^2} a_j$
5: $\quad$ **end for**
6: $\quad F^{(n+1)} := F_{J-1}^{(n)}$
7: **end for**
8: **return** $F^{(n+1)}$

in its continuous realization. A detailed description can be found in [84]. Within the scope of this dissertation, the explicit discrete form of the iteration step is of importance as it serves us as a building block to the RESESOP-Kaczmarz algorithm, which will be introduced in the next section.

2.4.3 RESESOP-Kaczmarz method

This method is a combination of SESOP and Kaczmarz and was first introduced in [14]. By incorporating multiple search directions $(\mathcal{A}^\eta)^*(\mathcal{A}^\eta f_n - g^\delta)$ in one iteration, SESOP accelerates the reconstruction process. However, it cannot take into account a semi-discrete formulation of the forward mapping. This is easily incorporated by the Kaczmarz method, which allows us to combine the respective sub-problems $a_j F = g_j$, $j = 0, \dots J-1$, as seen in Section 2.4.2. The RESESOP-Kaczmarz method can then efficiently incorporate the local characteristics of the inverse problem, specifically, the local modeling errors, into the regularization process. These local errors depend on geometry parameters, which are the scan angles φ and the detector points s. Thus, the semi-discrete formulation of the inverse problem allows us to separate these modeling errors into sub-problems. Therefore, the combination of both methods can be used to employ multiple search directions and the semi-discrete realization of the forward operator $\mathcal{A}$ in the same iteration. For this reason, this

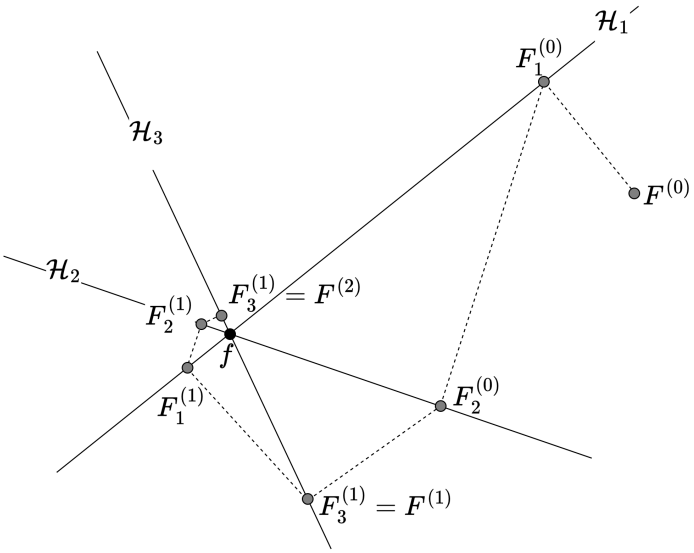


Fig. 2.11.: For $X = \mathbb{R}^2, N = 3$ two iterations are shown. An initial $F^{(0)}$ is given and then iteratively projected onto hyperplanes $\mathcal{H}_i$ for $i \in \{1, 2, 3\}$, which results in $F^{(1)}$. Repeating this process for another iteration shows that $F^{(2)}$ is approximating f.

method is particularly well suited for dynamic CT as the model error of the dynamic forward operator varies over time and is therefore dependent on the scan angles φ.

The global model inexactness η, see (2.20), can now be incorporated in the semi-discrete form

$$\|\mathcal{A}_{k,l}^{\eta} - \mathcal{A}_{k,l}\| \leqslant \eta_{k,l} \in [0, \infty),$$

where the operators and the model inexactness are now semi-discrete with $k \in \mathcal{K}$ and $l \in \mathcal{L}$. Equivalently, the quantum noise, see (2.14), has a semi-discrete form $\delta_{k,l}$ and can be formulated as

$$\|g_{k,l}^{\delta} - g_{k,l}\| \leqslant \delta_{k,l} \in [0, \infty).$$

Instead of incorporating both indices k, l into the notation, it is common practice to use an index function as in (2.10).

Rather than writing $\mathcal{A}_{k,l}$, we write $\mathcal{A}_j$ for an iteration index $j \in \{0, 1, \dots J-1\}$, where $J = K \cdot L$, that cycles through all $k \in \mathcal{K}$ and $l \in \mathcal{L}$. Furthermore, it is common to use the *modulo operator* to reduce the number of iteration indices by using the iteration index n of the algorithm to cycle through $j(k, l)$. For example, for $n \in \{0, 1, \dots\}$, we write $\mathcal{A}_{[n]} = \mathcal{A}_{n \mod J}$. When $[n]$ iterates through $\{0, \dots, J-1\}$

for r full cycles, then $n = r \cdot J$, with $r \in \mathbb{N}_{>0}$.
In the following, we refrain from indexing inexactness $\eta = (\eta_{[n]})_{n\in\{0,1,\dots\}}$ or data noise $\delta = (\delta_{[n]})_{n\in\{0,1,\dots\}}$ when they are used as indices themselves. For example, for noisy measurement data $g_{[n]}^{\delta_{[n]}}$, we write $g_{[n]}^{\delta}$.

The Kaczmarz iteration can be implemented with respect to indices k and l or by iterating over one and averaging over the other. With the two discretization indices k, l encoded in $[n]$, the version implemented for the purpose of this dissertation iterates explicitly over $\mathcal{K}$ and $\mathcal{L}$.
The iteration step can be found in Equation 2.36.

Definition 2.8 (SESOP-Kaczmarz [14]). *The iteration step of SESOP-Kaczmarz reads*

$$f_{n+1} = f_n - \sum_{i\in I_n} t_{n,i}\mathcal{A}_{[i]}^* w_{n,i}, \tag{2.36}$$

where f_n is the current iterate, I_n a finite index set and $w_{n,i} \in Y$ for all $i \in I_n$. $n \in \mathbb{N}$ is the iteration index *and $t_{n,i} \in \mathbb{R}$. $\mathcal{A}_{[i]}^* w_{n,i}$ is called a* search direction*, where $[i] \coloneqq i \mod J$.*

The vector t_n now minimizes the convex function h_n (see (2.28)), where the operator $\mathcal{A}$ is in its discrete form $\mathcal{A}_{[n]}$. In terms of metric projections $\mathcal{P}$, the individual hyperplanes are now defined as

$$\mathcal{H}(\mathcal{A}_{[n]}^* w_{n,i}, \langle w_{n,i}, g_{[n]}\rangle).$$

For this method, we adjust the definition of the solution set for the semi-discrete problem to

$$\mathcal{M}_{\mathcal{A}_j,g_j} \coloneqq \mathcal{M}_{\mathcal{A},g}^j \coloneqq \{f \in X : \mathcal{A}_{j(k,l)} f = g_{j(k,l)} \text{ for all } k \in \mathcal{K}, l \in \mathcal{L}\}.$$

Similarly to (2.31) and (2.32) we can now define the bounded solution set

$$\mathcal{M}_{\mathcal{A},g}^{j,\rho} \coloneqq \mathcal{M}_{\mathcal{A},g}^j \cap B_\rho(0) \tag{2.37}$$

and for $z \in \mathcal{M}_{\mathcal{A},g}^{j,\rho}$ obtain the estimate

$$\begin{aligned}\left\|\mathcal{A}_j^\eta z - g_j^\delta\right\| &= \left\|(\mathcal{A}_j^\eta - \mathcal{A}_j)\, z + \mathcal{A}_j\, z - g_j^\delta\right\| \\ &\leqslant \delta_j + \eta_j \rho.\end{aligned}$$

Regularization can be achieved through the same steps as for RESESOP, by projecting onto stripes instead of hyperplanes, and by terminating the iteration once a stop criterion is satisfied. See (2.39) for the definition of RESESOP-Kaczmarz stripes and (2.38) for the discrepancy principle.

Algorithm 2.4 (RESESOP-Kaczmarz [14]). *Let $f_0 \in \overline{\mathcal{B}_\rho(0)} \subset X$. For each $n \in \mathbb{N}$, define a finite index set I_n and search directions $(\mathcal{A}^\eta_{[i]})^* w_{n,i}$ with $w_{n,i} \in Y$ and $i \in I_n$.*
If the discrepancy principle

$$\|\mathcal{A}^\eta_{[i]} f_n - g^\delta_{[i]}\| \leqslant \tau(\eta_{[i]}\rho + \delta_{[i]}) \tag{2.38}$$

with $\tau > 1$ is satisfied, we stop iterating. Otherwise, the next iterate f_{n+1} is computed as the metric projection

$$f_{n+1} := \mathcal{P}_{H_n^{\eta,\delta,\rho}}(f_n)$$

of the previous iterate f_n , where

$$\mathcal{H}_n^{\eta,\delta,\rho} := \bigcap_{i \in I_n} \mathcal{H}\left(u_{n,i}^{\eta,\delta}, \alpha_{n,i}^{\eta,\delta}, \xi_{n,i}^{\eta,\delta,\rho}\right), \tag{2.39}$$

with

$$\begin{aligned} u_{n,i}^{\eta,\delta} &:= (\mathcal{A}^\eta_{[i]})^* \, w_{n,i}^{\eta,\delta}, \\ \alpha_{n,i}^{\eta,\delta} &:= \langle w_{n,i}^{\eta,\delta}, g^\delta_{[i]} \rangle, \\ \xi_{n,i}^{\eta,\delta,\rho} &:= (\delta_{[i]} + \eta_{[i]}\rho)\|w_{n,i}^{\eta,\delta}\|. \end{aligned}$$

For proof of the convergence of RESESOP-Kaczmarz and its regularization, we refer to Blanke et al. [14].

The RESESOP-Kaczmarz method has been used in this research as a comparison to the network architectures introduced in Chapter 4. For this purpose, the algorithm was implemented for two search directions following closely the instructions in [14]. The pseudocode for this can be found in Algorithm 2. The method iterates until either the discrapancy principle, that is, the stopping criterium is satisfied or a maximum number of N iterations have been performed. Here, we explicitly iterate through both the scan angles and the detector pixels.

Algorithm 2 RESESOP-Kaczmarz [14]

Initialize: $\text{scannerAngles} \leftarrow 0 \dots K-1$,
$\text{detectorPixels} \leftarrow 0 \dots L-1$,
$f_0 \leftarrow$ empty phantom
Given: $\mathcal{A} \leftarrow$ weight matrix
$\eta, \delta \leftarrow$ global model inexactness and noise level
$g^\delta \leftarrow$ perturbed sinogram
k, l indicate a specific scan angle and detector point
$N \leftarrow$ maximum number of iterations

while ¬stoppingCriterion or $n < N$ **do**
  for angle index k in scannerAngles **do**
    for detector index l in detectorPixels **do**
      $w_{k,l} \leftarrow \mathcal{A}_{k,l} \cdot f_n - g^\delta_{k,l}$
      if $\|w_{k,l}\| \leqslant \tau \cdot (\delta + \eta_{k,l})$ **then**
        ▷ discrepancy principle is satisfied for k, l
        $\text{stoppingCondition}_{k,l} \leftarrow$ True
        continue
      end if
      $\alpha \leftarrow w_{k,l} \cdot g_{k,l}$
      $\xi \leftarrow (\delta_{k,l} + \eta_{k,l}) \cdot \|w_{k,l}\|$
      ▷ compute first search direction u_{new}
      $u_{\text{new}} \leftarrow (\mathcal{A}_{k,l})^* \cdot w_{k,l}$
      ▷ projection w.r.t. first search direction
      $\tilde{f} \leftarrow f_n - \frac{\|w_{k,l}\| \cdot (\|w_{k,l}\| - (\delta_{k,l} + \eta_{k,l}))}{\|u_{\text{new}}\|^2} \cdot u_{\text{new}}$
      if is first iteration **then**
        $f_{n+1} \leftarrow \tilde{f}$ ▷ solely one search direction in first iteration
      else
        ▷ consider second search direction u_{old}
        if $\langle u_{\text{old}}, \tilde{f} \rangle > \alpha_{\text{old}} + \xi_{\text{old}}$ **then**
          $t \leftarrow \frac{\langle u_{\text{old}}, \tilde{f} \rangle - (\alpha_{\text{old}} + \xi_{\text{old}})}{\|u_{\text{new}}\|^2 \cdot \|u_{\text{old}}\|^2 - \langle u_{\text{new}}, u_{\text{old}} \rangle^2}$

        else if $\langle u_{\text{old}}, \tilde{f} \rangle < \alpha_{\text{old}} - \xi_{\text{old}}$ **then**
          $t \leftarrow \frac{\langle u_{\text{old}}, \tilde{f} \rangle - (\alpha_{\text{old}} - \xi_{\text{old}})}{\|u_{\text{new}}\|^2 \cdot \|u_{\text{old}}\|^2 - \langle u_{\text{new}}, u_{\text{old}} \rangle^2}$
        else
          $t \leftarrow 0$
        end if
        ▷ both search directions for iterations $n > 1$
        $f_{n+1} \leftarrow \tilde{f} + \langle u_{\text{new}}, u_{\text{old}} \rangle \cdot t \cdot u_{\text{new}} - \|u_{\text{new}}\|^2 \cdot t \cdot u_{\text{old}}$
      end if
      ▷ save variables from this iteration
      $u_{\text{old}} \leftarrow u_{\text{new}}$
      $\alpha_{\text{old}} \leftarrow \alpha$
      $\xi_{\text{old}} \leftarrow \xi$
      $n \leftarrow n + 1$

    end for
  end for
end while
return f_{n+1}

2.4.4 Dremel method

Kilian Dremel first introduced this method in his dissertation [35]. Its German title translates to "Modeling of the measurement process and implementation of a model-based iterative solution procedure for sectional image reconstruction for X-ray computed tomography". We will refer to this as the *Dremel method*. The algorithm is based on the Kaczmarz iteration but interlaces a correction step that adjusts the operator and its adjoint. This correction step inherently identifies shifts within the detector plane; however, it is also capable of estimating movements of the object (or of the source) since these shifts can be approximately translated into each other, as elaborated in [35]. Thus, in each main iteration step, the algorithm iterates over all scan angles with index $k \in \{0, \dots K-1\}$ and determines the necessary shift of the operator $\mathcal{A}_k$ for all detector points. For each iteration step, there is a current assumed operator $\mathcal{A}_k$ and a current reconstruction $f^{(n)}$ for iteration $n = 0, 1, \dots$. The correction process involves modifying the operator $\mathcal{A}_k$ for the upcoming iteration $n+1$. This is achieved by optimizing the cross-correlation between the measured data g_k^δ for angle k and the forward projection of the intermediate reconstruction $\mathcal{A}_k f^{(n)}$, considering all potential detector shifts.

Algorithm 3 Dremel [35]

Initialize: $f^{(0)} \in \mathbb{R}^M$, $\omega > 0$
Given: Operator for each scan angle with no shift: $\mathcal{A}_0, \dots \mathcal{A}_{K-1}$,
1: Perturbed data g^δ
2: **for** $n = 0, 1, \dots, N-1$ **do**
3: $\quad f \leftarrow f^{(n)}$
4: $\quad$ **for** $k = 0, \dots K-1$ **do**
5: $\quad\quad g_k \leftarrow \mathcal{A}_k f$
6: $\quad\quad f \leftarrow f - \omega\, \mathcal{A}_k^* (g_k - g_k^\delta)$
7: $\quad\quad s_k \leftarrow s_k - \lambda \frac{\Delta d}{m} \text{shift\_cross\_corr}(g_k, g_k^\delta)$
8: $\quad\quad \mathcal{A}_k \leftarrow$ operator at angle k with shift s_k
9: $\quad$ **end for**
10: $\quad f^{(n+1)} \leftarrow f$
11: **end for**
12: **return** $f^{(N)}$

At line 7 of Algorithm 3, the function $\text{shift\_cross\_corr}(g_k, g_k^\delta)$ calculates the displacement along the detector axis that maximizes the cross-correlation between the single-angle projection vectors g_k and $g_k^\delta \in \mathbb{R}^L$. Here, L denotes the total number of detector points. The computed shift is multiplied by the detector pixel size, Δd, and then divided by the magnification m, which is derived from the distances between the source and detector, as well as the source and object (considering the geometry of the fan beam). This yields an estimation of the object's displacement

perpendicular to the current scan angle, which is further adjusted by an optional relaxation factor, $\lambda \in (0, 1]$. In order to achieve subpixel accuracy in the shifts, [35] suggests up-sampling the projections prior to computing the cross-correlation. The shift computation is given by

$$\text{shift\_cross\_corr}(y, y^\delta) = \frac{1}{r_{\text{SR}}} \cdot \Bigg(- \left\lfloor \frac{n_{\text{SR}}}{2} \right\rfloor + \left(\left\lfloor \frac{n_{\text{SR}}}{2} \right\rfloor + \arg\max_l \ \text{cross\_corr} \left(\text{up}_{r_{\text{SR}}}(y), \text{up}_{r_{\text{SR}}}(y^\delta) \right)_l \right) \bmod n_{\text{SR}} \Bigg),$$

where

$$\text{cross\_corr}(y_{\text{SR}}, y^\delta_{\text{SR}}) = \ \text{post}\Big(\mathcal{F}^{-1} \Big(\mathcal{F}(\text{pre}(y^\delta_{\text{SR}}))^* \mathcal{F}(\ \text{pre}(y_{\text{SR}})) \Big) \Big),$$

and $\text{up}_{r_{\text{SR}}}(\cdot)$ is the up-sampling operator, $n_{\text{SR}} = r_{\text{SR}} \cdot L$ is the dimension of the up-sampled projection vectors $y_{\text{SR}}, y^\delta_{\text{SR}} \in \mathbb{R}^{n_{\text{SR}}}$, and $\mathcal{F}$ is the discrete Fourier transform; the functions $\text{pre}(\cdot)$ and $\text{post}(\cdot)$ are pre- and post-processing functions. In [35], an edge filter has been suggested as a preprocessing when correlating low-contrast signals.
Detailed parameter settings for evaluation of the Dremel method on the datasets presented in (5.1.1) are described in (5.2.2).

Unlike SESOP and Kaczmarz, there is no convergence analysis available for the Dremel method yet.

2.4.5 Filtered Back Projection

In CT, *filtered back projection* (FBP) is one of the most widely used reconstruction algorithms. It relies on the *Fourier slice theorem* and the Radon transform. The following is based on the respective sections in the book "The Mathematics of Computerized Tomography" by Natterer [84] and is supplemented by insights from [116].

A projection was previously defined as the intensities of all X-rays on the detector for a certain angle. We continue with the notation from (2.1), where $\omega := \omega(\varphi) \in S^1$ and $s \in \mathbb{R}$. Let $p_\varphi(s)$ be the projection function that, for an angle φ, maps each point s to the integral of the objective function over all lines with angle φ. This is

equivalent to the Radon transform (see Definition 2.1) evaluated at $\omega(\varphi)$ and s, that is,

$$p_\varphi(s) \coloneqq (\mathcal{R}f)(\omega, s).$$

The backprojection of angle φ would then map each point in the object domain to the intensity value of the point on the detector to which it was projected. Intuitively speaking, the intensity value of the detector is smeared over the object domain. This is repeated for all projection angles $\varphi \in [0, 2\pi]$, and we integrate over all values, yielding

$$f_{bp}(x) = \int p_\varphi(\langle x, \omega \rangle) d\varphi.$$

However, such a backprojection cannot achieve sharp images. To correct for the blur of the backprojection, the projection function p_φ is first translated into the Fourier space by applying the *2-d Fourier transform*.

Definition 2.9 (n-dimensional Fourier transform [84])**.** *The* Fourier transform *of an n-dimensional function $f \in L^1(\mathbb{R}^n)$ is defined as*

$$(Ff)(\xi) = \frac{1}{\sqrt{2\pi}^n} \int_{\mathbb{R}^n} f(x) e^{-i\xi x} dx. \tag{2.40}$$

The inverse Fourier transform *is defined as*

$$(F^{-1}F)(x) = \frac{1}{\sqrt{2\pi}^n} \int_{\mathbb{R}^n} F(\xi) e^{i\xi x} d\xi. \tag{2.41}$$

However, the object function f is unknown, and only the projection function p_φ is available. Fortunately, the *Fourier slice theorem* solves this problem.

Theorem 2.4.1 (Fourier slice theorem [84])**.** *For $f \in L^1(\mathbb{R}^n)$, $\omega \in S^1$ and $\xi \in \mathbb{R}$, the* Fourier slice theorem *states that*

$$F(\mathcal{R}f)(\xi) = (2\pi)^{\frac{n-1}{2}} F(f)(\xi\omega). \tag{2.42}$$

Proof. For $x \in \mathbb{R}^n$, $y \in \{\omega^\perp\}$ and $s \in \mathbb{R}$, we can substitute $x = s\omega + y$, leading to $s = \langle x, \omega \rangle$ and $dx = dy\, ds$. Then the following holds true:

$$\begin{aligned} F(\mathcal{R}f)(\xi) &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} (\mathcal{R}f)(\omega, s) e^{-is\xi}\, ds \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{-is\xi} \int_{\omega^\perp} f(s\omega + y)\, dy\, ds \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}^n} e^{-i\xi\langle x,\omega\rangle} f(x)\, dx \\ &= (2\pi)^{\frac{n-1}{2}}\, F(f)(\xi\omega). \end{aligned}$$

□

The theorem states that a 1-d Fourier transform in s of the projection function for angle φ, that is, the left side of (2.42), is the same as a single line through the 2-d Fourier representation of function f in $x \in \mathbb{R}^2$, that is, the right side of (2.42). This line crosses the origin at the angle φ. Using all projection angles $\varphi \in [0, 2\pi]$, the full Fourier representation of f can be constructed. Subsequently, the original function f can be retrieved by using the inverse 2-d Fourier transform.

However, when solving the backprojection numerically, the Fourier transform is applied to discrete projection data with a finite number of angles φ_k, defined as in (2.8), and a finite number of detector pixels s_l (2.9). The sampling of angles and detector points leads to problems in the Fourier domain.

The transformed projection data for an angle φ is arranged on a line through the center of the Fourier domain of f and can be written as

$$F\left(\mathcal{R}f\right)(\xi) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} (\mathcal{R}f)(\omega, s) e^{-i\xi s} ds. \tag{2.43}$$

Thus, the transformed data for all angles and detector points are arranged in circles around the origin. Due to the tomographic setup to acquire projections, the Fourier domain of the function f is sampled in such a way that the low frequencies are located in the center and the high frequencies are at the edges. Consequently, the sampling density is higher for low frequencies compared to high frequencies. These higher frequencies are important for sharp edges and details, which is why another step is required for sharp reconstructions, that is, applying a *filter*. This filter can be expressed as a function w and is applied in the Fourier domain to suppress low frequencies and increase the relative importance of higher frequencies. The *ramp* filter is the fundamental filter for FBP; however, other filters, such as Shepp-Logan,

and Cosine filters are possible [85]. The frequencies in the Fourier domain are scaled by multiplying the transformed projection data by the Fourier transform of the filter. The *inverse Fourier transform* can then be applied to the filtered frequencies. All these components can be summarized in this term:

$$\underbrace{\frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \underbrace{\overbrace{F\left(\mathcal{R}f\right)(\xi)}^{\substack{\text{Fourier transform} \\ \text{of projection data}}} \quad \overbrace{F(w)(\xi)}^{\substack{\text{Fourier transform} \\ \text{of filter}}}}_{\substack{\text{filtered Fourier} \\ \text{transform}}} e^{i\xi s} d\xi}_{\text{inverse Fourier transform}} \tag{2.44}$$

Applying these transformations for all scan angles φ, the filtered backprojection can be written as

$$f_{\text{fbp}}(x) := \int_0^{\pi} F^{-1}\left(F(\mathcal{R}f)(\xi)\, F(w)(\xi)\right)\left(\langle x, \omega \rangle\right) d\varphi, \tag{2.45}$$

where $\omega = \omega(\varphi)$ and w is the filter function.

2.4.6 Further Reading

There are many other reconstruction methods for inverse problems. Let us consider some of those that are specialized for dynamic inverse problems, as discussed in Section 2.3.3. Arridge et al. [6] introduced a joint reconstruction coupled with a low-rank decomposition approach for such problems. They postulate that temporal disturbances can be represented by a few key features, allowing the reconstruction to be split into the spatial and temporal aspects of these features. The authors then propose to incorporate the feature extraction into the reconstruction process. One benefit of this approach is that it allows separate regularization of temporal and spatial features, potentially enhancing reconstruction quality. This method has been numerically evaluated for different numbers of projection angles per time step. The results show that stable reconstruction quality can be achieved with five or more angles per time step. Returning to the dynamic inverse problem relevant to this dissertation, discussed in Section 2.3.3, we are interested in dynamic inverse problems, where projection angles and time steps are in a one-to-one relationship. This method has not been evaluated for this case.

The iterative RESESOP-Kaczmarz method, which has been introduced in Section 2.4.3, has been applied to dynamic CT problems in several forms [77, 104,

102]. Using the operator inexactness, this method can reconstruct dynamic objects even when only an estimate of the modeling error is available [77, 14]. However, this method is very time-consuming. For this reason, Sarnighausen et al. [104] combine this iterative method with a deep learning approach and a dynamic version of the FBP. First, RESESOP-Kaczmarz is performed for a small number of iterations to reconstruct the object at the first and last time steps. Landmark detection is then used to detect the corners of the object at both time steps and by calculating the average of the shifts between corresponding corners the total shift of the object is estimated. Finally, this information is used in the dynamic FBP. The numerical evaluation of this hybrid method on a rectangular object showed improved reconstructions compared to static FBP and 30 iterations of RESESOP-Kaczmarz. In [77] the RESESOP-Kaczmarz method has been evaluated in combination with several neural network post-processing models. The results show that this method is a good foundation for data-driven post-processing; however, the model inexactness is still a requirement for this combined reconstruction approach. This research inspired the development of one of the components of the RESESOP-Kaczmarz method, that is, SESOP, into a fully learned reconstruction network, as presented in Chapter 4. The goal is to build a more flexible approach without a priori knowledge of the sample-specific model inexactness. Another hybrid approach with RESESOP-Kaczmarz and machine learning is presented in [102] and will be further discussed in 3.5.

CT reconstructions face challenges not only because of dynamic factors but also due to low-dose radiation and limited-angle tomography. In this context, many learned reconstruction methods have been proposed. In Section 3.4, some of these will be discussed in more detail.

3 Deep Learning

Artificial intelligence (AI) has become ubiquitous in various aspects of modern society, including computer tomography. It attempts to imitate human intelligence by monitoring its surroundings, learning from them, and producing outputs that improve its likelihood of achieving a specified goal [101]. An early approach to AI was to use *knowledge bases*, a collection of knowledge in a formal language that computers can understand. These systems can then use logical inference rules to solve tasks [43]. This process was never successful because accurately describing the world with formal rules proved to be too complex. The defeat of knowledge bases suggested that systems must learn and acquire knowledge independently, which is what *machine learning* (ML) achieves [43].

Machine learning is a subset of AI. One of the most influential parts of the success of AI is the data provided and its representation. ML models receive data in the form of features that have been established to be important to the goal and a label that represents the data of interest. For example, to determine the value of a house, i.e. the label, an ML model would need a dataset with input features such as property size, number of rooms, location, and year of build.

It is not always feasible to extract high-level abstract features from raw data. Instances of such raw data could be natural language processing, speech recognition, or facial recognition from images or videos.

When training a machine learning model, the parameters are learned through an optimization algorithm to reduce the error between the output of the model and the provided label, thus solving a minimization problem.

Returning to the subject of this dissertation, we reconstruct the scanned object from its sinogram, a form of image processing. There is no viable option to translate all the information of a sinogram to a set of features equivalent to the house price example. This is where *neural networks* come in, a subset of machine learning, see Figure 3.1. Neural networks represent a special type of machine learning algorithm. The initial aim was to simulate the learning processes of the brain, which led to the term *artificial neural network* [43]. In neural networks, the architectures are designed to emulate neurons, which take in signals, process them, and then transmit the modified signals to the subsequent neuron. *Deep learning* describes a category of

neural networks that are especially large and attempt to solve particularly complex problems.

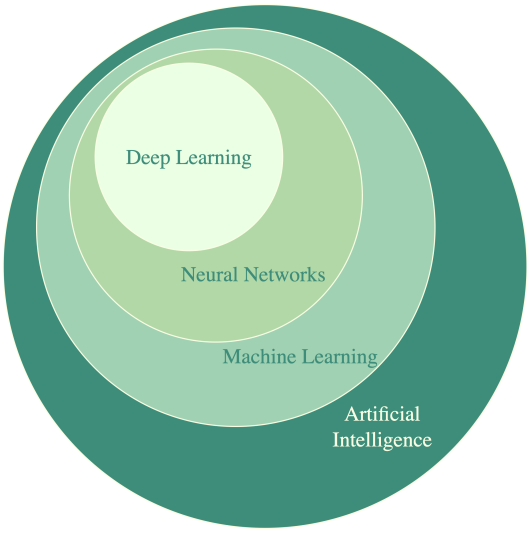


Fig. 3.1.: The connection between artificial intelligence, machine learning, neural networks and deep learning.

A neural network is called *deep* when it consists of many successive layers. These layers make it possible to change the representation of the input data into more abstract information. In image processing, it is common practice to use convolutional neural networks (see Section 3.2.2). Such convolutional layers can decompose an object in an image into its edges and corners and then into contours until the model recognizes whole parts of the object [43].

The first part of this chapter will discuss basic concepts from probability theory and statistical learning (3.1) that build the foundation for machine learning training algorithms. Section 3.2 discusses fundamental network architectures: fully connected networks and convolutional networks. In Section 3.3 the concepts of statistical learning find application in training algorithms for neural networks. Several optimization algorithms are explained as well as possible loss functions and regularization techniques. Section 3.4 summarizes multiple deep learning frameworks in the context of CT reconstructions and focuses on unrolled iterative methods.

3.1 Fundamentals of Probability Theory and Statistical Learning

Before diving into the specific network architectures crucial for this dissertation, it is important to examine why probability theory is elementary to AI.

AI systems typically interact with noisy, incomplete, or ambiguous datasets. Probability theory offers a mathematical framework for quantifying and managing that uncertainty. By representing inputs as random variables within probabilistic models, AI systems can evaluate and consider multiple *hypotheses* simultaneously. This approach prevents systems from prematurely committing to a single interpretation, which may be incorrect, thus enhancing their capacity for accurate and flexible problem solving. Instead of embedding fixed rules, modern AI systems identify patterns by estimating the underlying distribution of the data they see. Using statistical estimation methods, such as *empirical risk minimization* and *maximum likelihood estimation*, these models are able to infer which patterns represent authentic signals and which are random fluctuations. This process ensures that artificial intelligence is capable of adapting to novel inputs based on the true structure of the data it processes. In scenarios where a network model must decide between various possible actions, such as suggesting a product or finding the best route, it evaluates expected results using probability. By linking probabilities to different events with associated *cost functions*, the model can determine which action will maximize the expected benefit or minimize the expected loss.
This section will introduce basic concepts from probability theory and statistical learning that will find application in the training and optimization algorithms in Section 3.3.

3.1.1 Some Fundamentals of Probability Theory

As mentioned above, probability theory is concerned with assigning the probability of specific outcomes to random experiments. Here, a few concepts are introduced that lay the foundation for statistical learning.

Definition 3.1 (σ-algebra [65])**.** *Let Ω be a non-empty set, then $\mathcal{S} \subset 2^{\Omega}$ is a σ-algebra, if*

- $\Omega \in \mathcal{S}$,
- *if $S \in \mathcal{S}$, then $S^c \in \mathcal{S}$, where $S^c := \Omega \backslash S$,*

- *for a countable number of $S_i \in \mathcal{S}$, then $\bigcup_{i\in\mathbb{N}} S_i \in \mathcal{S}$.*

Definition 3.2 (Probability measure [65])**.** *For a non-empty set Ω and $\mathcal{S} \in 2^\Omega$ a σ-algebra, a function $\mathcal{P} : \mathcal{S} \to [0, 1]$ is called a* probability measure *on $(\Omega, \mathcal{S})$ if*

- $\mathcal{P}(\Omega) = 1$,
- *for each countable sequence of pairwise disjoint sets $S_1, S_2, \dots \in \mathcal{S}$ the following is true*

$$\mathcal{P}(\bigcup_{i\in\mathbb{N}} S_i) = \sum_{i\in\mathbb{N}} \mathcal{P}(S_i).$$

Now we can define a *probability space*.

Definition 3.3 (Probability space)**.** *Let the triple $(\Omega, \mathcal{S}, \mathcal{P})$ denote the* probability space *where*

- *the* sample space *Ω defines a non-empty set of all possible outcomes of a random experiment,*
- *$\mathcal{S} \subset 2^\Omega$ is a σ-algebra of Ω which is the space of possible results of the experiment, the so-called* event space*,*
- *and $\mathcal{P} : \mathcal{S} \to [0, 1]$ defines the* probability measure *on the event space $\mathcal{S}$.*

For a set Ω' and a σ-algebra $\mathcal{S}'$, $(\Omega', \mathcal{S}')$ is called a *measurable space*. From now on $(\Omega, \mathcal{S}, \mathcal{P})$ is a probability space and $(\Omega', \mathcal{S}')$ is a measurable space.

A function V from $(\Omega, \mathcal{S})$ to $(\Omega', \mathcal{S}')$ with $V^{-1}(S') \in \mathcal{S}$ for all $S' \in \mathcal{S}'$ is called *measurable function* from $(\Omega, \mathcal{S})$ to $(\Omega', \mathcal{S}')$ or a *random variable* from $(\Omega, \mathcal{S}, \mathcal{P})$ to $(\Omega', \mathcal{S}')$. When a random variable is applied to a sample $\omega \in \Omega$, resulting in $v = V(\omega)$, we obtain a realization of V.
The *probability distribution* of the random variable V is defined as $\mathcal{P}_V = \mathcal{P} \circ V^{-1}$ with respect to $(\Omega, \mathcal{S}, \mathcal{P})$, and $(\Omega', \mathcal{S}', \mathcal{P}_V)$ is also a probability space. If the random variable V follows the distribution $\mathcal{P}$, we write $V \sim \mathcal{P}$.

A *hypothesis* is a statement that aims to describe the relationship between variables or provide an explanation for an observed event. Hypotheses can be tested with a *loss function*. Let $\mathcal{H}$ be the set of hypotheses, the function

$$\ell : \mathcal{H} \times \Omega \to \mathbb{R} \tag{3.1}$$

is called a *loss function*, which evaluates the selection of a hypothesis h for an element within the sample space Ω. In the stochastic framework, it is necessary to consider the expected loss relative to the distribution $\mathcal{P}$.
The *expectation value* is defined as

$$\mathbb{E}[V] = \int_{\Omega} V(\omega) d\mathcal{P}(\omega),$$

see [65]. In the context of statistical learning, we are interested in the expectation value of a function of the realizations, not the realizations themselves. For example, the loss function might be applied to the realizations. Thus, we consider the expectation value

$$\mathbb{E}[f(V)] = \int_{\Omega} f(V(\omega))\ d\mathcal{P}(\omega),$$

see [43].
To emphasize that ω is drawn from the distribution $\mathcal{P}$ the notation $\mathbb{E}_{\omega\sim\mathcal{P}}(f(V))$ is preferred.

The *risk function* for $h \in \mathcal{H}$ is given by

$$\mathbf{L}_{\mathcal{P}}(h) = \mathbb{E}_{\omega\sim\mathcal{P}}[\ell(h,\omega)]. \tag{3.2}$$

This function denotes the expectation value of the loss ℓ given the hypothesis h.

3.1.2 Some Fundamentals of Statistical Learning

The core concept of machine learning is to "learn" from the data. For a clearer grasp of this concept, the two primary methods through which machine learning models *learn* are presented here: *supervised learning* and *unsupervised learning*.
In addition, there are methods such as self-supervised [9], semi-supervised [12], and reinforcement learning [101, 43].

Learning methods In *supervised learning*, a *training dataset* $D_s = \{(x_i, y_i)\}_{i=1}^{N}$ for $N \in \mathbb{N}$ is a set of pairs from $X \times Y$, where every y_i is the label corresponding to the respective input features x_i. In the scenario of predicting house prices, the features serve as input data while the price of the houses represents the label. In the context of image classification, images constitute the input data and the output might, for example, consist of a set of labels that identify the objects within the images.
An unknown function f describes the relationship between data pairs: $f(x_i) = y_i$.

Let $\mathcal{F}_\theta$ be the machine learning model with parameters $\theta \in \Theta$ that is trained to approximate f. During training, the parameters are *learned* such that $\mathcal{F}_\theta(x_i) = \hat{y}_i$, where $\hat{y}_i$ should be close to y_i. A good model can then predict accurate labels from previously unseen samples $T_s = \{(x_j, y_j)\}_{j=1}^{K}$ with $K \in \mathbb{N}$, the so-called *test data*, where $\hat{y}_j$ approximates y_j sufficiently well [101]. Generally, a third dataset is used as *validation data*. During training, the model can be evaluated on the validation data to test its generalizability. Preferably, the test and validation datasets should be different, so that after the training process is finished, the model can be tested on entirely unseen data. A classic example of supervised learning is email filtering. Such a model analyzes data from a set of emails, including the sender's address, subject, and content, to find patterns and determine whether the message is legitimate or spam. The labels are provided by manually annotating which emails are spam for a sufficient number of samples.

Unsupervised learning does not work with annotated data, but only with input data. The dataset can therefore be defined as $D_{us} = \{x_i\}_{i=1}^{N}$ with $N \in \mathbb{N}$. Deep learning was previously described as a way to change the representation of the input data. Unsupervised learning has a similar purpose. Instead of learning a predefined output label, the model develops a new representation of the input data [41]. This can be applied in clustering tasks to detect anomalies that do not fit any group of data samples or to segment datasets by shared characteristics [39]; or in dimensionality reduction for recommender systems [91].

Empirical risk

Let $\Omega = X \times Y$ be a sample space and $\mathbf{F}_\Theta := \{\mathcal{F}_\theta : X \to Y \text{ with } \theta \in \Theta\}$ the space of hypotheses.

For random variables $\mathbb{X}$ and $\mathbb{Y}$ with realizations in X and Y, respectively, there is an unknown distribution $\mathcal{P}$, such that $(\mathbb{X}, \mathbb{Y}) \sim \mathcal{P}$, called *true distribution*. We are interested in evaluating how well a hypothesis $\mathcal{F}_\theta \in \mathbf{F}_\Theta$ describes the relationship $x \mapsto y$ between variables for $x \in X$ and $y \in Y$. For this, the *loss function* is defined as a mapping ℓ:

$$Y \times Y \to \mathbb{R}, \quad (\hat{y}, y) \mapsto \ell(\hat{y}, y). \tag{3.3}$$

The loss function is defined so that $\ell(\hat{y}, y)$ is small when $\hat{y}$ is close to y.

The *risk* evaluates how well a hypothesis $\mathcal{F}_\theta$ describes the relationship between the random variables $\mathbb{X}$ and $\mathbb{Y}$ that follow the distribution $\mathcal{P}$. It is defined as

$$L(\mathcal{F}_\theta) := \mathbb{E}_{(\mathbb{X},\mathbb{Y})\sim\mathcal{P}}[\ell(\mathcal{F}_\theta(\mathbb{X}), \mathbb{Y})].$$

The true distribution $\mathcal{P}$ is unknown, however, a set of realizations $\{(x_i, y_i)\}_{i=1}^N$ is available. These pairs are assumed to be independent and identically distributed (i.i.d.) following the *empirical distribution* $\mathcal{P}_D$, also called *data distribution*.

The empirical risk evaluates the performance of a hypothesis on a given dataset.

Definition 3.4 (Empirical Risk [43])**.** *Let $D = \{(x_i, y_i)\}_{i=1}^N \in (X \times Y)^N$ be a dataset of N i.i.d. samples from the distribution $\mathcal{P}_D$. The data pairs (x_i, y_i) are realizations of the random variables $\mathbb{X}$ and $\mathbb{Y}$, respectively. The function $\mathbf{L}_D : \mathbf{F}_\Theta \to \mathbb{R}$ is called* empirical risk *and is defined as*

$$\mathbf{L}_D(\mathcal{F}_\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(\mathcal{F}_\theta(x_i), y_i), \tag{3.4}$$

where $\ell : Y \times Y \to \mathbb{R}$ is a loss function.

The empirical risk is an approximation of the true risk:

$$\mathbb{E}_{(\mathbb{X},\mathbb{Y})\sim\mathcal{P}}[\ell(\mathcal{F}_\theta(\mathbb{X}), \mathbb{Y})] \approx \frac{1}{N} \sum_{i=1}^{N} \ell(\mathcal{F}_\theta(x_i), y_i).$$

Depending on the function $\mathcal{F}_\theta$ and its respective spaces X and Y, there are many possible choices for the loss function. For example, if $\mathcal{F}_\theta$ is trying to solve a classification problem such as filtering spam for legitimate emails, then $Y = \{0, 1\}$, where 0 indicates that the input is not spam. Then ℓ could be a binary cross-entropy loss, which returns a value between 0 and 1 indicating the probability that an email is spam. However, if $\mathcal{F}_\theta$ solves a regression problem, for example, estimating house prices, and $Y = \mathbb{R}^d$ with $d \in \mathbb{N}_{>0}$, then the loss function must reflect that. Popular loss functions for regression tasks are mean squared error (see (3.3.3)) or the mean absolute error (see (3.3.3)).

The goal in machine learning is to find a specific hypothesis $\mathcal{F}_\theta \in \mathbf{F}_\Theta$ that minimizes the risk $\mathbf{L}_D$ of a given dataset D. This concept is referred to as *empirical risk minimization* [82].

Definition 3.5 (Empirical Risk Minimization (ERM) [82])**.** *In the setting of (3.4) the ERM problem is defined as*

$$\mathcal{F}_\theta^+ \in \underset{\mathcal{F}_\theta \in \mathbf{F}}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \ell\left(\mathcal{F}_\theta(x_i), y_i\right),$$

where $\mathbf{F}$ *is restricted to a specific hypothesis class. This restriction is called an* inductive bias.

It is important to note that $\mathcal{F}_\theta^+$ corresponds to the minimal *empirical* risk, rather than the minimum risk across the unknown distribution $\mathcal{P}$. This constraint frequently results in overfitting of $\mathcal{F}_\theta^+$ to the dataset D, meaning that the model captures the relationships between x_i and y_i in D too precisely. Consequently, this may hinder the model's ability to generalize to unseen data. To combat this problem, the hypothesis class is restricted by incorporating prior knowledge [110]. Consider the dataset shown in Figure 3.2, where training and test samples are plotted, both of the same unknown distribution. Three functions of varying degrees are shown, the linear function underfits the training data as it does not represent a good fit for training or testing samples. However, the polynomial of 15 degrees overfits the data. It closely represents the training data, but it cannot fit the test data well. The line for the polynomial of degree 4 is a compromise. It can fit the training and test data appropriately. Thus, restricting the set of possible hypotheses $\mathbf{F}$ to polynomials of smaller degree would combat the problem of overfitting.
The empirical risks associated with the training data and the test data can be used to assess the performance of these three models. The model should perform well on the training data and should be able to generalize well on unknown data. Thus, the goal is to have a low training error as well as a low test error, the so-called *generalization error*.

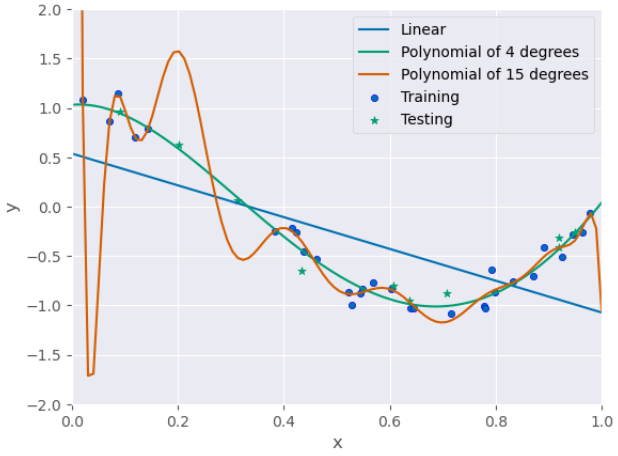


Fig. 3.2.: Graphic representation of overfitting vs underfitting. Training and test data are generated using the function $f(x) = \cos(1.5\,\pi\,x)$. On the interval between 0 and 1 the polynomial of degree 4 fits the training and test data best. The polynomial of 15 degrees overfits the data and the linear model underfits the data.

Maximum likelihood estimation

Until now, there have been no presumptions regarding the empirical distribution $\mathcal{P}_D$ of the dataset D. However, we will now consider this distribution to be of a parametric nature. This approach allows us to apply the *maximum likelihood estimation* to determine the parameters that result in the maximum joint probability of the observed data. Therefore, rather than applying a broad hypothesis space within the framework of ERM, we can determine the parameter $\theta \in \Theta$ that facilitates the creation of a distinct hypothesis. The following is based on Goodfellow et al. [43, p.128].

Let us continue with the dataset D and the unknown but true distribution $\mathcal{P}$. The data pairs $(x_i, y_i) \in D$ are realizations of the random variables $\mathbb{X}$ and $\mathbb{Y}$ that follow the true distribution $\mathcal{P}$. The goal of this method is to find a function that estimates the probability that data (x, y) are from the true distribution $\mathcal{P}$. Thus, the family of parametric functions $p_{(\mathbb{X},\mathbb{Y})}((x, y); \theta)$ is introduced, that evaluates the probability that x is a realization of $\mathbb{X}$ and y is a realization of $\mathbb{Y}$ for some parameter $\theta \in \Theta$. For better readability, we write $p((x, y); \theta)$. The goal of maximum likelihood estimation is to find a θ that maximizes this probability.

Definition 3.6 (Maximum likelihood estimation)**.** *In the setting of Definition 3.4. The maximum likelihood estimation for $\theta \in \Theta$ is defined as*

$$\theta^* \in \arg\max_{\theta} \prod_{i=1}^{N} p\left((x_i, y_i); \theta\right). \tag{3.5}$$

Applying the logarithm to this term does not alter the underlying result; however, it provides an expression that is more convenient to calculate and analyze. Thus, the above can be written as

$$\arg\max_{\theta} \prod_{i=1}^{N} p\left((x_i, y_i); \theta\right) = \arg\max_{\theta} \sum_{i=1}^{N} \log p\left((x_i, y_i); \theta\right).$$

Furthermore, we can divide the right side of this expression by the number of samples in the dataset and receive an expression that is based on the expectation with respect to the distribution of the dataset D. Thus, we receive

$$\arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \log p\left((x_i, y_i); \theta\right) = \arg\max_{\theta} \mathbb{E}_{(\mathbb{X},\mathbb{Y})\sim\mathcal{P}}[\log p\left((\mathbb{X}, \mathbb{Y}); \theta\right)]. \tag{3.6}$$

The probability that some data (x, y) are drawn from the true distribution is given by $p_{true}((x, y))$. The dissimilarity between the true probability and the parametric function that models the true probability can be measured by the *Kullback-Leibler* (KL) [43] divergence, which is given by

$$\begin{aligned} D_{KL}(p_{true}||p) &= \mathbb{E}_{(\mathbb{X},\mathbb{Y})\sim\mathcal{P}}\left[\log p_{true}((\mathbb{X},\mathbb{Y})) - \log p((\mathbb{X},\mathbb{Y});\theta)\right] \\ &= \mathbb{E}_{(\mathbb{X},\mathbb{Y})\sim\mathcal{P}}\left[\log p_{true}((\mathbb{X},\mathbb{Y}))\right] - \mathbb{E}_{(\mathbb{X},\mathbb{Y})\sim\mathcal{P}}\left[\log p((\mathbb{X},\mathbb{Y});\theta)\right]. \end{aligned}$$

To find the best parameter θ such that the function $p(\cdot, \theta)$ models the true probability, we must minimize this term. Because the left side does not depend on the parametric function $p(\cdot, \theta)$ it is sufficient to minimize the right side. The true distribution $\mathcal{P}$ is unknown, however, this term can be approximated using the available data distribution $\mathcal{P}_D$. Thus, maximizing in (3.5) is equivalent to minimizing

$$-\mathbb{E}_{(\mathbb{X},\mathbb{Y})\sim\mathcal{P}_D}\left[\log p((\mathbb{X},\mathbb{Y});\theta)\right] = -\frac{1}{N}\sum_{i=1}^{N}\log p((x_i, y_i);\theta). \tag{3.7}$$

Thus, maximum likelihood estimation serves as an approach to align the true distribution with the empirical data distribution $\mathcal{P}_D$. The true goal is to match the parameters to the true distribution $\mathcal{P}$ which is unknown.

Corollary 3.1.1. *For parameters $\theta \in \Theta$ the maximum likelihood estimation is equivalent to the minimization problem*

$$\theta^* \in \arg\min_{\theta\in\Theta}\frac{1}{N}\sum_{i=1}^{N}\log p((x_i, y_i);\theta).$$

Consequently, maximum likelihood estimation can be viewed as a special case of empirical risk minimization, where a parametric family of probability distributions and a parameter space θ are used.
Next, the foundational ideas of neural networks will be presented. In Section 3.3, the discussion will turn to training algorithms, at which point the concepts of statistical learning will once again be relevant.

3.2 Neural Networks

There are many forms of machine learning models, for example k-nearest neighbors [32], decision trees [92], or support vector machines [117]. This dissertation focuses

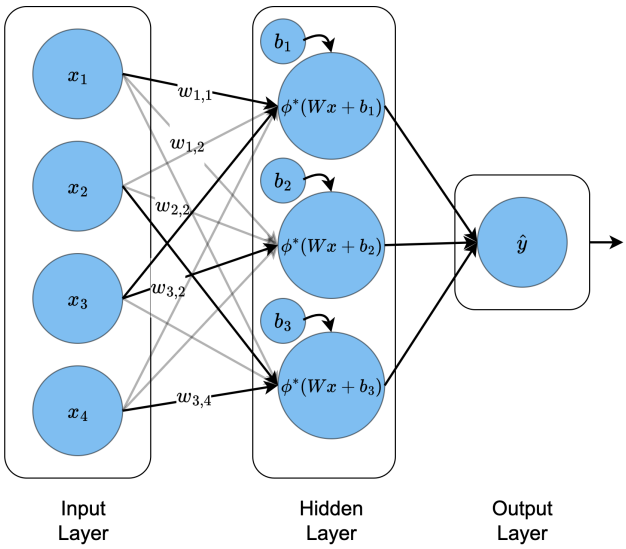


Fig. 3.3.: Fully connected neural network with four input neurons, one hidden layer with three neurons and one output neuron. The data x in the hidden layer is a vector of the input data $(x_1, x_2, x_3, x_4)^T$ and W is a matrix consisting of all $w_{i,j}$ with $i = 1, 2, 3$ and $j = 1, 2, 3, 4$.

on *deep neural networks*. The simplest form of such a network is the *fully connected* neural network. Given that our problem involves image data, *convolutional* neural networks (see Section 3.2.2) are especially relevant.

3.2.1 Fully Connected Neural Network

The fundamental form of a neural network is represented by the *fully connected neural network* (FCNN), illustrated in Figure 3.3. This type of network consists of an input layer, an output layer, and a number of hidden layers. Each layer consists of a number of neurons, for input and output, this number is determined by the dimensions of input data x and output data y. This type of network is called fully connected because the neurons in one layer are connected to all neurons in the next layer.

The connections between neurons are called *weights*. The input of a neuron is multiplied by a set of weights before being processed by a non-linear transformation in the neuron. This operation is done for each neuron in a layer and can be summarized in Definition 3.7.

Definition 3.7 (Fully connected layer)**.** *Let $x \in \mathbb{R}^m$, $W \in \mathbb{R}^{n \times m}$ be a weight matrix. A fully-connected layer is then defined as*

$$\mathcal{L}_{W,b}(x) := \phi(Wx + b),$$

where $b \in \mathbb{R}^n$ is a bias vector and $\phi : \mathbb{R}^n \to \mathbb{R}^n$ an activation function defined component wise by a non-linear operation $\phi^* : \mathbb{R} \to \mathbb{R}$ with

$$\phi(z) := (\phi^*(z_i))_{i \in \{1,\dots m\}}. \tag{3.8}$$

The set of all the weights and biases of a network is defined as the parameter $\theta \in \Theta$. Following this, the complete network can be written as the concatenation of such layers.

Definition 3.8 (Fully connected neural network)**.** *Let X be the input space, Y the output space, l be the number of layers, and W_k, b_k be the parameters of layer $k \in \{1, \dots, l\}$. The fully connected neural network is then defined as*

$$\mathcal{F}_\theta(x) := \mathcal{L}^{(l)}_{W_l,b_l} \circ \mathcal{L}^{(l-1)}_{W_{l-1},b_{l-1}} \circ \dots \mathcal{L}^{(1)}_{W_1,b_1}(x) = \hat{y},$$

where $\theta = \{W_1, \dots, W_l, b_1, \dots, b_l\}$.

Activation Functions

In the definition of the fully connected layer (see (3.7)) the *non-linear* activation function was introduced. This function provides a crucial component to the learning capabilities of neural networks. For each neuron, this function influences the impact that the signal has on the next layer. Non-linearity gives the network the ability to learn complex relationships from the data. If the activation functions were linear, any combination of neurons would only be able to model a linear mapping. According to the *universal approximation theorem*, a neural network composed of merely two layers, one non-linear layer followed by a linear layer, is capable of approximating any continuous function arbitrarily well [101].
Continuous differentiability is desirable for gradient-based optimization methods. The *backpropagation*, which will be introduced in Section 3.3.1 and Algorithm 5, requires the derivative of the activation function. However, there are popular activation functions that are only piecewise differentiable. In practice, at the points where the function is not differentiable, an arbitrary value is assigned to the derivative.

This section introduces some of the most popular activation functions and briefly discusses their use in neural networks. A visual comparison of those functions can be found in Figure 3.4.

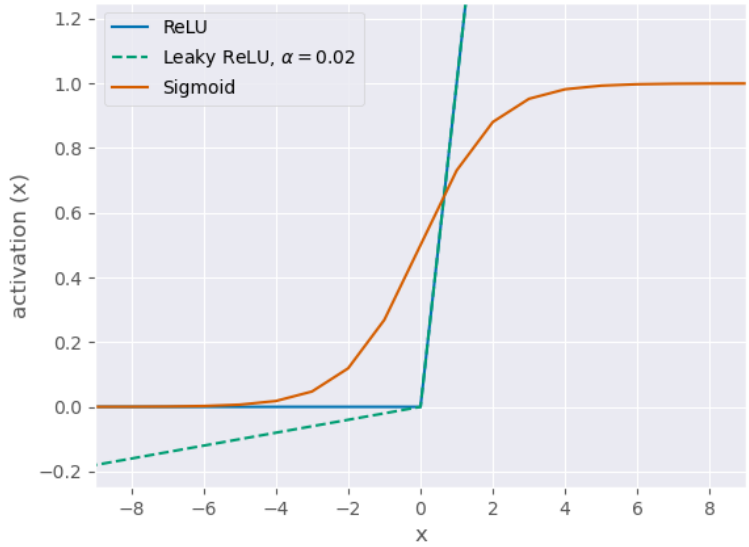


Fig. 3.4.: Sigmoid, ReLU and Leaky ReLU activation functions are shown on the interval $[-10, 10]$

Definition 3.9 (Sigmoid)**.** *The sigmoid activation function is defined for* $x \in \mathbb{R}$ *as*

$$\mathit{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \in (0, 1).$$

The output values are in the interval of $(0, 1)$ which makes this function particularly useful for classification tasks, where, for example, the output could signify the probability of belonging to a certain class. It is therefore often used in the output layer to scale the values to their required range.
The sigmoid function has areas of saturation where the gradients come close to zero. When the activation values reach these areas, changes in the input have very little impact on the activation values, thus training becomes inefficient. This is called *vanishing gradient problem* and can significantly affect the network.

The *rectified linear unit* (ReLU) is a very popular choice for deep neural networks due to the ease of computation.

Definition 3.10 (ReLU)**.** *The ReLU activation function is defined for* $x \in \mathbb{R}$ *as*

$$\mathit{ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \in [0, \infty).$$

This function is partially linear and piecewise differentiable as it is differentiable everywhere except $x = 0$. Popular implementations, such as PyTorch [88], use zero as the gradient at this point. Negative values are mapped to zero, which simplifies calculations; however, this modification can result in weights ceasing to be trained once neurons reach a state of zero, a phenomenon often referred to as *dying neurons*, due to the fact that their gradients are consequently also zero. This can introduce sparsity into the network, which can enhance the generalization abilities of the model [42]; however, it may also impede its learning progression. Consequently, maintaining balance is essential, leading to the development of variations of ReLU.

One of these variations is the *Leaky ReLU*, where for $x < 0$ a small constant is multiplied by the input, thus eliminating the problem of dying neurons.

Definition 3.11 (Leaky ReLU)**.** *The leaky ReLU activation function is defined for* $x \in \mathbb{R}$ *and a predefined parameter* $\alpha \in \mathbb{R}_{>0}$ *as*

$$\mathit{LReLU}(x, \alpha) = \begin{cases} x, & \text{if } x \geqslant 0 \\ \alpha x, & \text{otherwise} \end{cases} \in (-\infty, \infty).$$

The parameter α must be set to a small positive value before training and ensures that the activations do not become zero and that their gradients can still have an impact during training. Compared to the standard ReLU activation, this can lead to faster training times, however, it adds another hyperparameter, which needs to be selected carefully. The following activation function goes one step further, by learning this parameter.

The *parametric Rectified Linear Unit* (PReLU) behaves similarly to the leaky ReLU, with the distinction that the parameter α is incorporated within the set of learnable parameters θ. This results in a minimal increase in computational load during the training process, while simultaneously eliminating the need for additional training iterations to determine an optimal value for α. For the problem addressed in this dissertation, only activation functions applicable to regression tasks are discussed. In Chapter 4, the activation functions of the sigmoid and PReLU are of interest.

FCNN in Context of Image Reconstruction

The advantage of fully connected neural networks is that they are structure agnostic [95] meaning that input data does not have to be adapted to the network. For example, image data with $m \times n$ pixels can be fed to the network as an array of length $m \cdot n$. However, there are also major disadvantages to applying FCNNs to image data like that. The dataset introduced in Section 5.1.1 contains images of phantoms with 255 x 255 pixels and sinograms with 133 x 723 pixels. Should these images be processed by an FCNN with only one layer, the input layer would require 96,159 neurons, a hidden layer would require 96,159 neurons, and the output layer would require 65,025 neurons. Consequently, the total number of weights would reach 15,499,292,256. Despite such a huge parameter space, such a network would not be effective for predictions on image data.
Another disadvantage of FCNN in this context, is that the location of pixels within the image is not important to the network. Randomly shuffling the pixels of an image would not have an effect on the outputs; however, it is obvious that the relative location of a pixel to its neighboring pixels can be very important [101]. For these reasons, it is crucial to apply specialized layers that are more efficient in learning from image data, while also taking into account the structural characteristics of such inputs.

3.2.2 Convolutional Neural Network

A solution to these problems is the concept of *convolution*. Instead of considering each pixel individually, a weight w, the so-called *kernel*, is applied to all groups of neighboring pixels in an image x.

Definition 3.12 (Discrete 2-d convolution [43])**.** *Let $x \in \mathbb{R}^{H \times B}$ and $w \in \mathbb{R}^{N \times M}$, with $H, B, N, M \in \mathbb{N}_{>0}$.The discrete 2-d convolution of x with w is defined as*

$$(x * w)(i,j) := \sum_{n=1}^{N} \sum_{m=1}^{M} x_{s \cdot i - n, s \cdot j - m} \, w_{n,m} \, , \tag{3.9}$$

where $s \in \mathbb{N}_{>0}$ is called stride *and i, j are such that $0 < s \cdot i - n \leqslant H$ and $0 < s \cdot j - n \leqslant B$.*

The parameters i, j are restricted in Definition 3.12 so that the indexing operations on x are valid. The kernel cannot be applied beyond the edges of x.

Figure 3.5 shows the graphic representation of a single convolution. The input image in blue is convolved with a 3×3 kernel in green by applying Equation 3.9. The result is a single value stored in the highlighted pixel of the purple output image. Applying this equation to all possible values i, j results in the full purple output.

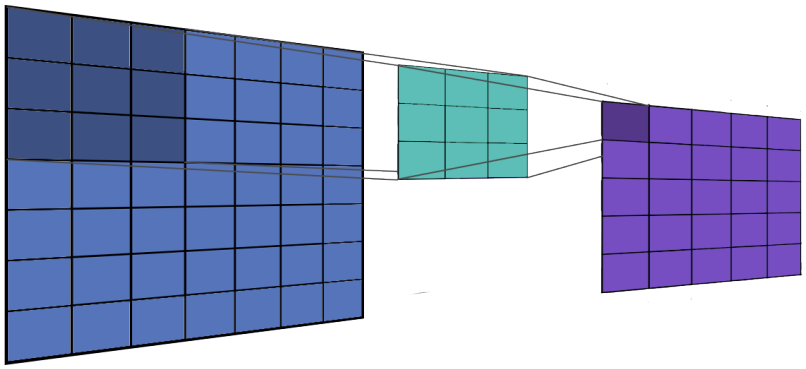


Fig. 3.5.: Convolution of a 7×7 sample (in blue) with a 3×3 kernel (in green).

In the concept of convolutional neural networks, the three core parameters are *kernel size*, *stride*, and *padding*. The kernel size (N, M) is usually set to be a small odd number, such as 3, 5 or 7 [73], in most cases $N = M$. Stride defines the step size between convolutions and padding adds zero-valued pixels to the perimeter of the input.

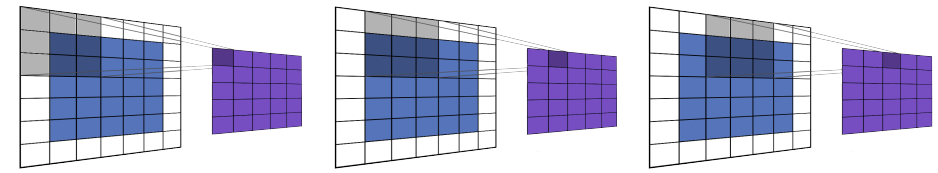


Fig. 3.6.: Three convolutions on a 5×5 image with a kernel of size 3×3, padding $= 1$ and stride $= 1$. The output is of size 5×5 as well. (a) To calculate the first pixel of the output, the convolution starts in the top left corner. Due to the zero-padding, 5 out of 9 pixels are zero. The first pixel of the input image will be used in 4 separate convolutions and will therefore impact 4 pixels in the output image. (b) The second convolution after shifting by $stride = 1$ uses the highlighted pixels from the input. (c) For the third convolution the first input pixel is not relevant anymore. A total of 25 convolutions have to be performed to create the output image.

Figure 3.6 shows an example of stride $= 1$ and padding $= 1$ as the highlighted section of the input image moves one pixel per convolution across the input image. This is a common choice for this parameter [101]. If sub-sampling of the input is required, the stride is chosen to achieve the desired dimensions. Figure 3.7 shows

an example with a stride of 2. The output image now only has a size of 3×3 pixels. In both images, shifts are illustrated in just one dimension, while convolutions in the other dimension are carried out in a similar manner.
Scanning the input in such a way implies that pixels on the edges are included less frequently in the convolutions than pixels farther away from the edges. By using padding, the kernel can be applied beyond the original perimeter of the input, and thus edge pixels appear more often in the convolutions; see Figure 3.6. Another effect of padding is that the dimensions of the output can be adjusted independently of the size and stride of the kernel [43]. Both padding and stride can be defined differently for either dimension. For input dimensions $H \times B$, kernel size $N \times M$, stride $s = (s_1, s_2)^T$ and padding $p = (p_1, p_2)^T$, the output dimensions $\hat{H} \times \hat{B}$ can be calculated as

$$\hat{H} = \left\lfloor \frac{H + 2 \cdot p_1 - (N-1) - 1}{s_1} \right\rfloor + 1, \quad \hat{B} = \left\lfloor \frac{B + 2 \cdot p_2 - (M-1) - 1}{s_2} \right\rfloor + 1.$$

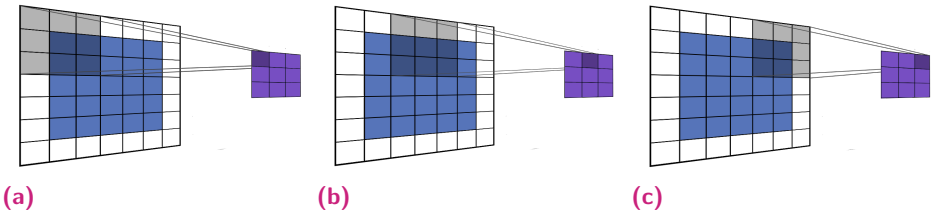
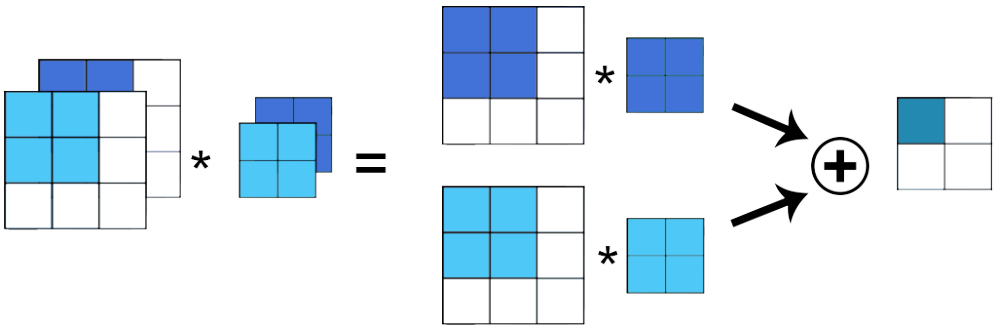


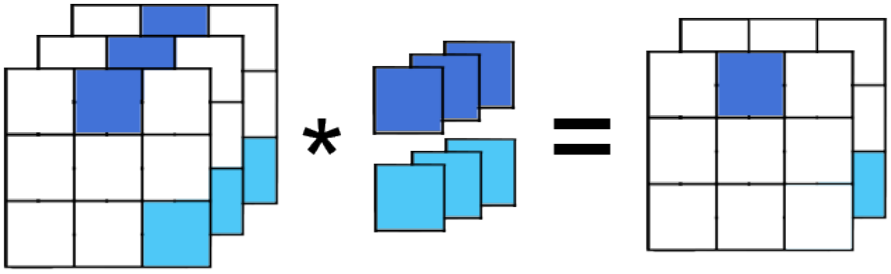
Fig. 3.7.: Three convolutions on a 5×5 image with a kernel of size 3×3, padding $= 1$ and stride $= 2$. Due to the sub-sampling through the stride parameter, the output is of size 3×3. (a) To calculate the first pixel of the output, the convolution starts in the top left corner. The first pixel of the input image will be used in only one convolution and will therefore impact one pixels in the output image. (b) The second convolution after shifting by $stride = 2$ uses the highlighted pixels from the input. The first pixel of the image is not relevant anymore. (c) After performing three convolutions, the complete row has been processed. For the complete output image only 9 convolutions are required.

Convolutional layers use *filters* of four dimensions, which are an extension of kernels. The first and second dimensions are used for the *input* and *output channels*, C_{in} and C_{out}, respectively, while the last two dimensions define the size of the kernel. Input channels permit different representations of input data. For example, colored images can be represented as data with three channels, according to the RGB values of the image. With output channels, it is possible to apply several kernels to an input within the same layer to generate several representations of the data.

Figure 3.8a shows a convolution of an image of size 3×3 with $C_{in} = 2$, $C_{out} = 1$ and a kernel of size 2×2, the filter is therefore of dimension $\mathbf{2} \times \mathbf{1} \times 2 \times 2$. As shown



(a) $C_{in} = \mathbf{2}$ and $C_{out} = \mathbf{1}$: input size $\mathbf{2} \times 3 \times 3$, kernel size 2×2 and therefore filter size $\mathbf{2} \times \mathbf{1} \times 2 \times 2$, output size $\mathbf{1} \times 2 \times 2$.



(b) $C_{in} = \mathbf{3}$ and $C_{out} = \mathbf{2}$: input size $\mathbf{3} \times 3 \times 3$, kernel size 1×1 and therefore filter size $\mathbf{3} \times \mathbf{2} \times 1 \times 1$, output size $\mathbf{2} \times 3 \times 3$.

Fig. 3.8.: Two examples of channel combinations for input and output channels. For both examples, let stride $= 1$ and padding $= 0$.

in the figure, the input and filter channels are paired and convolved. The results of each convolution are then summed up and provide the value of an individual pixel in the output. In this example, no padding is applied and therefore the result is of dimension $C_{out} \times \hat{H} \times \hat{B} = 1 \times 2 \times 2$.
The convolution process of 2 output channels is shown in Figure 3.8b. This time the input has 3 channels and the kernels are of size 1×1. Therefore, the dimensions of the filter are $\mathbf{3} \times \mathbf{2} \times 1 \times 1$. The different colors show where C_{out} is relevant. For each color the same process as in Figure 3.8a is applied. Due to the kernel size, the resulting output is of dimension $2 \times 3 \times 3$.

To include the concept of channels, Equation 3.9 can then be rewritten as follows.

Definition 3.13 (Multi-channel 2-d convolution [43]). *Let $x \in \mathbb{R}^{C_{in} \times H \times B}$ and $W \in \mathbb{R}^{C_{in} \times C_{out} \times N \times M}$ and let $\circledast$ be a convolution that also computes the aggregation of multiple channels:*

$$(x \circledast W)_{c_o}(i,j) := \sum_{c_i=1}^{C_{in}} \sum_{n=1}^{N} \sum_{m=1}^{M} x_{c_i, s\cdot i-n, s\cdot j-m} W_{c_i, c_o, n, m}, \tag{3.10}$$

where $s \in \mathbb{N}_{>0}$ is the stride, $c_o \in \{1, \dots, C_{out}\}$ and i, j such that the indexing operations inside the summation are valid.

Further

$$\begin{aligned}(x \circledast W)(i,j) &:= (x \circledast W)_{c_o \in \{1,\dots C_{out}\}}(i,j) \\ &:= \left(\sum_{c_i=1}^{C_{in}} (x_{c_i,\cdot,\cdot} * W_{c_i,c_o,\cdot,\cdot})(i,j) \right)_{c_o \in \{1,\dots C_{out}\}}\end{aligned}$$

and

$$(x \circledast W) := \Big((x \circledast W)(i,j) \Big)_{i,j} \tag{3.11}$$

Finally, a convolutional layer can be defined.

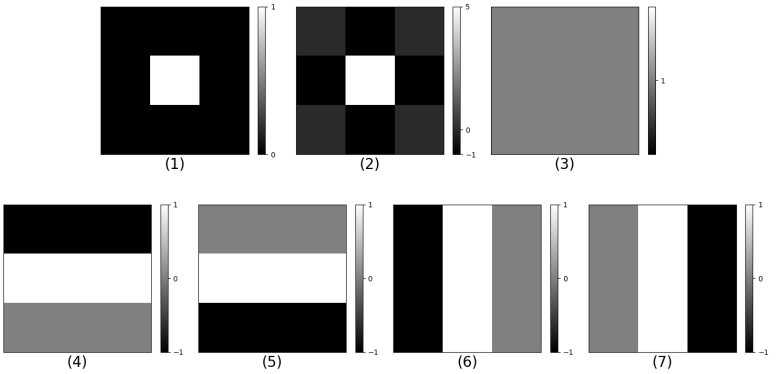
Definition 3.14 (Convolutional layer). *In the setting of Definition 3.13. A convolutional layer is defined as*

$$\mathcal{L}_{Conv,W,b}(x) := \phi\left((x \circledast W) + b\right), \tag{3.12}$$

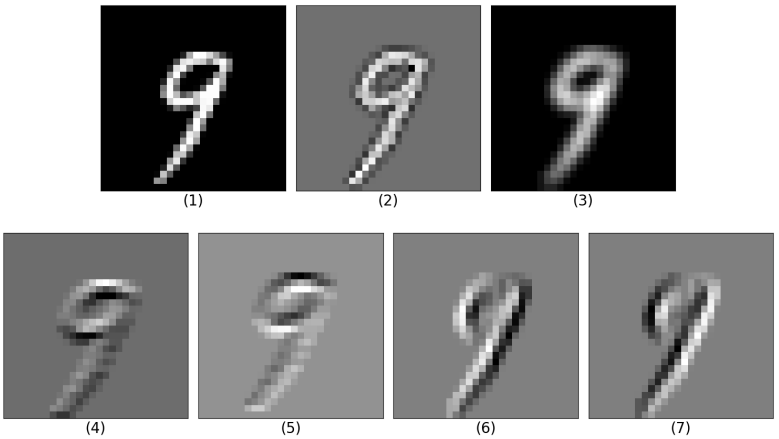
where $b \in \mathbb{R}^{C_{out} \times \hat{H} \times \hat{B}}$ is the bias and $\phi : \mathbb{R}^{C_{out} \times \hat{H} \times \hat{B}} \to \mathbb{R}^{C_{out} \times \hat{H} \times \hat{B}}$ an activation function, defined component-wise by

$$\phi(z) := \Big(\phi^*(z_{c,h,b}) \Big)_{c,h,b}.$$

Now that convolutional neural networks have been defined, the effect of different kernels will be demonstrated by applying a convolutional layer to a sample from the famous MNIST dataset of handwritten digits [33]. Seven different kernels have been selected, as can be seen in Figure 3.9a. The first row of this figure shows an identity kernel, a sharpening kernel, and a blurring kernel. The values of the individual pixels can be found in the respective color bars.



(a) These kernels have following effects: (1) identity, (2) sharpening, (3) blurring, (4) - (7) edge highlighting.



(b) Results of one convolution with above kernels and a sample from the MNIST dataset.

Fig. 3.9.: Kernels and their effects on a sample from the MNIST dataset.

Figure 3.9b shows the effects that each kernel has on the number 9 of the MNIST dataset. The first image in row one shows the original image, where the identity kernel has been applied. The second image shows more pronounced edges caused by a sharpening kernel. The last image in the row shows the effect of the blurring kernel. The second row shows how different sides of the number have been highlighted, caused by the second row of kernels.

Goodfellow et al. [43] highlight the efficiency of CNNs over FCNNs by considering the *sparsity* of weights and *weight sharing*.

Sparisty of weights In FCNNs, each value in one layer is connected to each element in the previous layer. To reduce the computational effort needed to calculate all these

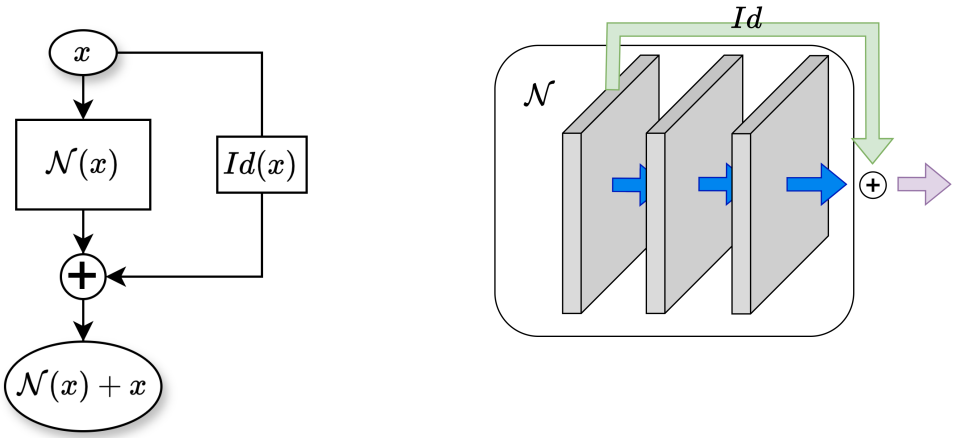
matrix multiplications, convolutions use small kernels; thus, pixels in the output are only connected to a small number of pixels in the input. For the case of a 2-d image with one channel, a pixel in the output of a convolution is only connected to $N \cdot M$ pixels in the input. Whereas in an FCNN a pixel is connected to all pixels in the previous layer, thus $H \cdot B$ multiplications are necessary. N and M are small numbers, while H and B can be very large. This leads to sparse connections between layers and therefore to faster computations and smaller model memory requirements.

Weight sharing This is a consequence of the convolutions; see (3.9), where w is fixed for all computations within this convolution. Consequently, all the pixel values in x are paired with the same $N \cdot M$ values in w. In contrast, FCNNs do not share any weights between nodes; each pixel has its unique weight connection to each pixel in the next layer.

3.2.3 Noteworthy Architectures

Two specific architecture types worth highlighting are the concepts of *residual networks* and *U-Net*. These have been combined with many other network architectures and have proven to be highly effective.

Residual network Convolutional neural networks have already been demonstrated to be more efficient in learning from image data than fully connected layers. However, the depth of the network is highly relevant. Several studies underline the importance of network depth [112, 113, 50, 58], where the best performing models leverage architectures with 16 to 30 layers [112, 113]. He et al. [51] raise the question whether the performance of a model can be improved simply by increasing its depth. It has been reported that there is an upper limit to the depth of the network beyond which its performance suffers. Furthermore, overfitting is not the cause of this "degradation problem" [51] as the training error increases when the depth increases after a certain threshold.
Given a well-performing shallow model, it should theoretically be possible to construct a deeper architecture by using this shallow model and adding identity mappings to it. However, according to [51], current optimization algorithms are incapable of finding a suitable solution for such deep architectures.
He et al. [51] propose the *deep residual learning framework*. A standard set of network layers learns a new representation $H(x)$ of the input x [101]. As the name suggests, instead of training a set of layers to learn the direct mapping: $x \to H(x)$,



(a) Residual building block according to [51].

(b) CNN residual block with three CNN layers in blue and a shortcut in green.

Fig. 3.10.: General structure of residual network block as well as possible design of CNN version of such a network.

the residual mapping $x \to N(x) := H(x) - x$ is learned. The output $N(x)$ is then recast as $N(x) + x$ as seen in Figure 3.10a. This type of layer block is known as the *ResNet* block.

The identity mapping in Figure 3.10a is also called a shortcut connection. Hypothetically, if x is already the optimal representation of the data, the model could set the weights of the residual part to zero and, therefore, replace the layer stack with the identity mapping. Experiments in [51] suggest that non-linear layers in deep architectures struggle to approximate an identity mapping, which leads to the degradation of very deep models.

The implementation of such identity connections depends on the dimensions of x and $N(x)$. When dimensions change, the identity mapping needs to replicate that change. This can be achieved by applying a convolution layer with a kernel of size 1×1 and suitable strides. These shortcut connections are computed at very little additional cost. The basic building block of a residual network, as suggested by [51], consists of two blocks of a convolutional layer followed by batch normalization and an activation function. Figure 3.10b shows a ResNet block with three CNN layers, visualized as blue arrows. The shortcut output is then added to the output of these CNN layers.

U-Net The *U-Net* is a very popular network architecture and was first introduced in [98] to segment biomedical images. This model was successful in limiting the required data set size and reducing the size of the model. The architecture is

based on CNN layers that form a contracting path and a symmetric expanding path, resembling the letter *U* and giving the model its name. This architecture is similar to an encoder-decoder model [30], with the addition of skip connections (also known as shortcuts) between the contracting and the expanding path. Chen et al. [30] combined this architecture with residual connections to reduce the noise found in low-dose CT images. There have been several methods using U-Nets as post-processing to reduce artifacts [77, 60, 59] or in combination with other methods as fully learned reconstruction methods [75].

3.3 Network Training

Having defined the structure of a neural network $\mathcal{F}_\theta : X \to Y$ and the dataset $D = \{(x_1, y_1), \dots (x_N, y_N)\} = (X \times Y)^N$, this section addresses neural network training algorithms. *Training* refers to finding the optimal parameter $\theta \in \Theta$ for a fixed network $\mathcal{F}_\theta$ for the dataset D.
The *loss* function ℓ, defined in (3.3), can also be viewed as evaluating the performance of the network parameter θ for the network $\mathcal{F}_\theta$ on a dataset D. The empirical risk minimization then turns into minimizing the risk function

$$\mathbf{E}_D(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(\mathcal{F}_\theta(x_i), y_i). \tag{3.13}$$

In the context of machine learning, this is called *objective function* or *cost function*. Following the definition of ERM (see (3.5)) and maximum likelihood estimation (see (3.1.1)), training refers to finding $\theta^* \in \Theta$, such that

$$\theta^* \in \arg\min_{\theta \in \Theta} \mathbf{E}_D(\theta).$$

However, for most deep learning problems, this minimization is not feasible. Instead, training algorithms restrict themselves to finding a local minimum. For this purpose, the gradient of $\mathbf{E}_D$ in direction θ is considered. The parameter θ^+ is a local minimum when

$$\nabla_\theta \mathbf{E}_D(\theta^+) = 0.$$

As shown in Figure 3.2, it is crucial that machine learning models have a low training error as well as a low *generalization error*. For this purpose, a *regularization* term can be included in the objective function, which now reads

$$\tilde{\mathbf{E}}_D(\theta) = \mathbf{E}_D(\theta) + \lambda \mathcal{R}(\theta),$$

where $\lambda \geqslant 0$ controls the intensity of the regularization. For individual samples x, y from the training data, this is written as

$$\tilde{\mathbf{E}}_{(x,y)}(\theta) = \ell(\mathcal{F}_\theta(x), y) + \lambda \mathcal{R}(\theta). \tag{3.14}$$

Further details of regularization techniques will be discussed in Section 3.3.4. In cases where it is clear from context which dataset is being utilized in the objective function, we avoid using specific indices for the data set. For example, we use $\tilde{\mathbf{E}}(\theta)$ instead of $\tilde{\mathbf{E}}_D(\theta)$.

Finding θ^+ for $\mathcal{F}$ is referred to as *training* and consists of several steps:

1. **Forward propagation**: The network with its parameters is evaluated on the training data.
2. **Calculate empirical risk**: The loss function is applied to calculate the empirical risk on the data.
3. **Backward propagation**: The gradient of the risk function is calculated for all parameters.
4. **Optimization of parameters**: The gradients are used to update the parameters.

Performing these steps on the training dataset is referred to as an *epoch*. Typically, training a neural network requires multiple repetitions of this process, that is, numerous epochs.

3.3.1 Forward and Backward Propagation

The first step of the training algorithm is to evaluate the current parameter θ of the network on the samples of the training data. We consider a fully connected neural network with l layers, indexed by $k \in \{1, \ldots, l\}$. The parameter θ consists of weights $W^{(k)} \in \mathbb{R}^{n \times m}$ and bias $b^{(k)} \in \mathbb{R}^n$ for each layer k. Let us examine two consecutive layers $k-1$ with m neurons and k with n neurons. Thus, the input

sample x is applied to all layers of the network starting at layer 1 and ending at layer l. A detailed representation of the layers $k-1$ and k is shown in Figure 3.11.

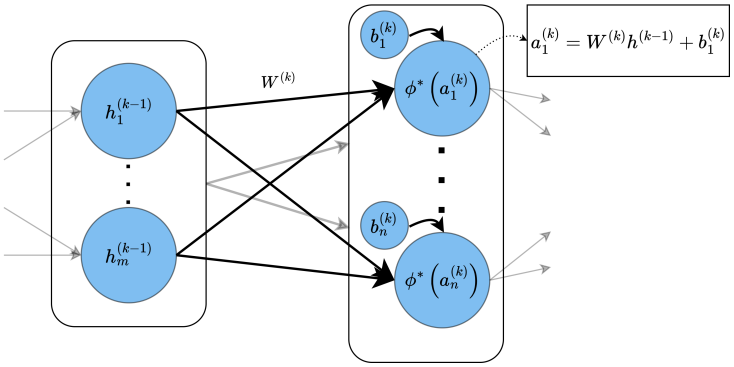


Fig. 3.11.: Following the notation in Algorithm 4, layers $k-1$ and k are shown. Layer k consists of n neurons, while layer $k-1$ consists of m neurons. The activation function ϕ^* is applied to each neuron, specifically to $a_i^{(k)} = W^{(k)}h^{(k-1)} + b_i^{(k)}$ and $h_i^{(k)} = \phi^*\left(a_i^{(k)}\right)$, with $i = 1, \ldots, n$.

Following the Definitions 3.7 and 3.8, the algorithm can be written as in (4), where the equations on the lines (3) and (4) define the forward propagation of the input data x through the network with parameters $W^{(k)}, b^{(k)}$. The calculations are summarized for all neurons in a layer, thus $h^{(k-1)} = (h_1^{(k-1)}, \ldots, h_m^{(k-1)})^T$, $a^{(k)} = (a_1^{(k)}, \ldots, a_n^{(k)})^T$, and $b^{(k)} = (b_1^{(k)}, \ldots, b_n^{(k)})^T$, as seen in Figure 3.11.

Algorithm 4 Forward Propagation [43]

Given: network depth l, weight matrices $W^{(i)}$, bias parameters $b^{(i)}, i \in \{1, \ldots, l\}$, the set of all parameters $\theta = \{(W^{(1)}, b^{(1)}), \ldots, (W^{(l)}, b^{(l)})\}$, input data $x \in X$, output data $y \in Y$, activation function ϕ.

1: $h^{(0)} \leftarrow x$
2: **for** $k = 1, \ldots, l$ **do**
3: $\quad a^{(k)} \leftarrow b^{(k)} + W^{(k)}h^{(k-1)}$
4: $\quad h^{(k)} \leftarrow \phi(a^{(k)})$
5: **end for**
6: $\hat{y} = h^{(l)}$
7: $\tilde{\mathbf{E}} = \ell(\hat{y}, y) + \lambda\mathcal{R}(\theta)$
8: **return**

After forward propagation and calculation of the regularized loss for samples x, y, the gradients are propagated from the last layer to the first.
The purpose of backpropagation is to tune all weights in θ so that the network $\mathcal{F}_\theta$ optimizes the risk $\tilde{\mathbf{E}}_D(\theta)$ in the training data. This tuning corresponds to minor changes in all parameters $W^{(k)}$ and $b^{(k)}$ for all layers k and considering all training samples $(x_i, y_i) \in D$. Thus, the gradients of all parameters for all training samples

have to be calculated.

The backpropagation is based on the *chain rule*.
For $x \in \mathbb{R}^m, y \in \mathbb{R}^n$ and $g : \mathbb{R}^m \to \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$. Then

$$\frac{\partial f(g(x))}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial f(g(x))}{\partial g(x)_j} \frac{\partial g(x)_j}{\partial x_i},$$

or written in vector notation as

$$\nabla_x f(g(x)) = \left(\frac{\partial g(x)}{\partial x}\right)^T \nabla_{g(x)} f(g(x)).$$

The backpropagation algorithm can be found in (5). Starting with the gradient $\nabla_{\hat{y}} \tilde{\mathbf{E}}$, the algorithm calculates the gradients of $\tilde{\mathbf{E}}$ in the directions $a^{(k)}, W^{(k)}, b^{(k)}$ for all layers $k = l, l-1, \ldots, 1$.

Algorithm 5 Backward Propagation [43]

Given: network depth l, weight matrices $W^{(i)}$, bias parameters $b^{(i)}, i \in \{1, \ldots, l\}$, the set of all parameters $\theta = \{(W^{(1)}, b^{(1)}), \ldots, (W^{(l)}, b^{(l)})\}$, predicted output $\hat{y} \in Y$, output data $y \in Y$, activation function ϕ.

1: $g \leftarrow \nabla_{\hat{y}} \tilde{\mathbf{E}} = \nabla_{\hat{y}} \ell(\hat{y}, y)$
2: **for** $k = l, l-1, \ldots 1$ **do**
3: $\quad g \leftarrow \nabla_{a^{(k)}} \tilde{\mathbf{E}} = g \odot \phi'(a^{(k)})$
4: $\quad \nabla_{W^{(k)}} \tilde{\mathbf{E}} = g\, h^{(k-1)^T} + \lambda \nabla_{W^{(k)}} \mathcal{R}(\theta)$
5: $\quad \nabla_{b^{(k)}} \tilde{\mathbf{E}} = g + \lambda \nabla_{b^{(k)}} \mathcal{R}(\theta)$
6: $\quad g \leftarrow \nabla_{h^{(k-1)}} \tilde{\mathbf{E}} = W^{(k)^T} g$
7: **end for**

Gradients are then used in an optimization algorithm to update the parameters. Thus, in each epoch, all parameters are updated once considering all samples in the training data.

3.3.2 Optimization Algorithms

Following the computation of all gradients of the network via forward and backward propagation, the next step involves modifying the parameters according to these gradients. The primary algorithm used for this task is called *gradient descent*. Several optimization algorithms used for optimization in modern deep learning applications derive from gradient descent. This section will provide an introduction to some of these algorithms.

To concentrate on the mechanics of these algorithms, we use the empirical risk without regularization as defined in (3.13). When using the regularized empirical risk, the algorithms have to be adjusted to include the gradient of the regularizer $\mathcal{R}$.

Gradient descent This section focuses on finding a minimum of the objective function $\mathbf{E}$ with respect to θ, for this $\mathbf{E}$ is assumed to be continuously differentiable in θ. The goal is to find $\theta^+ \in \Theta$, such that $\nabla_\theta \mathbf{E}(\theta^+) = 0$. In each iteration, this algorithm updates θ by incorporating a scaled version of the negative direction of the gradient.

Definition 3.15. *Let $\mathbf{E} : \Theta \to \mathbb{R}_{>0}$ be a continuously differentiable function. For starting value $\theta^{(0)} \in \Theta$, the gradient descent iteration step is given by*

$$\theta^{(k+1)} = \theta^{(k)} - \epsilon \nabla_\theta \mathbf{E}(\theta^{(k)}), \tag{3.15}$$

for $k = 0, 1, \ldots$ and learning rate *$\epsilon > 0$ [43].*

The gradient is computed as

$$\nabla_\theta \mathbf{E}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \left(\ell(\mathcal{F}_\theta(x_i), y_i) \right) = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\hat{y}} \ell(\mathcal{F}_\theta(x_i), y_i) \nabla_\theta \mathcal{F}_\theta(x_i). \tag{3.16}$$

Note that for the gradient of the risk function the gradient of $\ell(\hat{y}, y)$ (see (3.3)) in direction $\hat{y}$ is important, as y are the true data and therefore fixed. Thus, for all N samples in the dataset, the gradient of the network and the network itself have to be evaluated. The training set is referred to as a *batch*, the algorithm is therefore also referred to as *batch gradient descent*.

Stochastic gradient descent For modern datasets, which can easily contain millions of samples, it is computationally unfeasible to perform this gradient descent step for all samples in the dataset. An adaptation of the gradient descent, the so-called *stochastic gradient descent*, mitigates this problem.

In the iteration step of the stochastic gradient descent, the gradient is evaluated on a single sample. With an initial $\theta^{(0)}$, the iteration step reads

$$\theta^{(k+1)} = \theta^{(k)} - \epsilon^{(k)} \nabla_\theta \, \mathbf{E}_{(x_{i_k}, y_{i_k})}(\theta) \tag{3.17}$$

with $k = 0, 1, \ldots, N$ and $i_k \in \{1, \ldots, N\} =: \bar{N}$. $\bar{N}$ denotes the set of indexes of the training data [43, 15]. The risk function is not evaluated on the entire dataset, but on individual samples (x_{i_k}, y_{i_k}).
The calculation of this iteration step is significantly faster compared to the gradient descent step. However, utilizing a single sample results in substantial fluctuations in $\mathbf{E}$ at each iteration, which poses challenges when approximating a local minimizer.

Stochasic gradient descent with mini-batches Instead of performing the iteration on the complete training data (as in (3.15)), or only on a single sample (as in (3.17)), a hybrid approach is introduced, called *stochastic gradient descent with mini-batches*. Here, smaller subsets of the dataset are created, called *mini-batches*.

Definition 3.16 (Stochastic gradient descent with mini-batches [43])**.** *In the setting of Definition 3.15. Let I_k be a subset of the training data index set $\bar{N}$ with $|I_k| \ll N$, then*

$$\mathbf{E}_k(\theta) = \frac{1}{|I_k|} \sum_{i \in I_k} \ell(\mathcal{F}_\theta(x_i), y_i)$$

and the iteration step of the stochastic gradient descent with mini-batches reads

$$\theta^{(k+1)} = \theta^{(k)} - \epsilon^{(k)} \nabla_\theta \mathbf{E}_k(\theta^{(k)}), \tag{3.18}$$

where for $k = 0, 1, \ldots$ and the learning rate *$\epsilon^{(k)} > 0$.*

The gradient of $\mathbf{E}_k(\theta)$ then approximates $\nabla_\theta \mathbf{E}(\theta)$. The gradient is calculated analogously to (3.16), but is now feasible to compute because $|I_k| \ll N$.
Notice that the learning rate $\epsilon^{(k)}$ in stochastic versions of gradient descent depends on the iteration index k. This is due to the random sampling of subsets of the training data, which introduces a source of noise. The true gradient $\nabla_\theta \mathbf{E}(\theta)$ becomes 0 when it is at a minimum using the full training data. For smaller subsets the gradient of $\mathbf{E}_k(\theta^+)$ does not vanish, even when θ^+ is a minimum of $\mathbf{E}$.

The use of mini-batches has become the standard method for parameter optimization in machine learning. Consequently, instead of specifying stochastic gradient descent with mini-batches, we now refer to it simply as gradient descent.

An analysis of the convergence of stochastic gradient descent can be found in [15]. Convergence often relies in part on the learning rate satisfying

$$\sum_{k=1}^{\infty} \epsilon^{(k)} = \infty \qquad \text{and} \qquad \sum_{k=1}^{\infty} (\epsilon^{(k)})^2 < \infty.$$

Gradient descent with momentum This adaptation of gradient descent takes inspiration from physics, namely the force that moves particles through space: *momentum*. In physics this is defined as mass times velocity; in our parameter space we assume unit mass, which leaves velocity as our momentum. It defines the direction and rate at which the parameters move through the parameter space.
The purpose of this momentum algorithm is to accelerate learning for parameter spaces with large gradients, small but consistent gradients, or noisy gradients. The method calculates an exponentially decaying moving average of previous gradients and continues to follow their direction.

Definition 3.17 (Gradient descent with momentum [43, 90])**.** *In the setting of Definition 3.16. For an initial velocity* $v^{(0)}$ *and a momentum parameter* $\alpha \in [0, 1)$*, the iteration step of gradient descent with momentum reads*

$$v^{(k+1)} = \alpha v^{(k)} - \epsilon^{(k)} \nabla_\theta \mathbf{E}_k(\theta) \tag{3.19}$$
$$\theta^{(k+1)} = \theta^{(k)} + v^{(k+1)}, \tag{3.20}$$

where for $k = 0, 1, \ldots$ *and the* learning rate $\epsilon^{(k)} > 0$.

The momentum parameter α controls the impact of previous gradients on the current iteration.
For gradient descent without momentum, the step size, that is, the size of the parameter update, is only determined by the learning rate and the gradient of the current iteration. When momentum is used, it is also influenced by the impact and direction of previous gradients. The step size is highest when many previous gradients point in the same direction and lowest when the gradients point in opposing directions. The benefit of employing gradient descent with momentum lies in its ability to reach the minimum more quickly and is less likely to be trapped in a local minimum.
The *Nesterov* momentum takes a slightly different approach by applying the current velocity to the parameter $\theta^{(k)}$ before evaluating the gradient at those parameters.

Definition 3.18. *In the same setting as in (3.17), the Nesterov iteration reads as*

$$\tilde{\theta}^{(k)} = \alpha v^{(k)} + \theta^{(k)} \tag{3.21}$$
$$v^{(k+1)} = \alpha v^{(k)} - \epsilon^{(k)} \nabla_\theta \mathbf{E}_k(\tilde{\theta}(k)) \tag{3.22}$$
$$\theta^{(k+1)} = \theta^{(k)} + v^{(k+1)}, \tag{3.23}$$

Using the momentum method on the current parameters is viewed as a correction factor to the standard momentum method (see, (3.17)).

Algorithms with Dynamic Learning Rates

Several versions of gradient descent have been presented, yet the determination of the learning rate, which plays a crucial role in the algorithm's effectiveness, has been largely overlooked. The momentum algorithm has somewhat mitigated the impact of the learning rate, but introduces another parameter, which also needs to be selected carefully.
In machine learning models, it is often the case that some features are sparse. When optimizing such models, the average gradient of sparse features is small, resulting in the parameters of these features updating at slower rates.
Here, a few algorithms are presented that incrementally adapt the learning rate to the model parameters.

Adaptive gradient algorithm (AdaGrad) This algorithm offers a solution to the problem of adjusting the learning rate depending on the impact of the gradients. During each iteration, the sum of the squares of all previous gradient values is calculated and used to scale the learning rate, by dividing it by the square root of the calculated sum [43].

Definition 3.19. *In the setting of Definition 3.16. Let $\delta > 0$ be a small constant, the iteration step of the AdaGrad algorithm [37] reads*

$$\begin{aligned} r^{(k+1)} &= r^{(k)} + \left(\nabla_\theta \mathbf{E}_k(\theta^{(k)}) \odot \nabla_\theta \mathbf{E}_k(\theta^{(k)})\right) \\ \Delta^{(k+1)} &= -\left(\epsilon \oslash \left(\delta + \sqrt{r^{(k+1)}}\right)\right) \odot \nabla_\theta \mathbf{E}_k(\theta^{(k)}) \\ \theta^{(k+1)} &= \theta^{(k)} + \Delta^{(k+1)}, \end{aligned}$$

where $\odot$, $\oslash$ and $\sqrt{\cdot}$ are the component-wise multiplication, division and square root, respectively and $\epsilon > 0$ is the learning rate.

Note that ϵ is the base learning rate, which stays fixed throughout the iterations. However, the *effective* learning rate is adjusted according to the size of the respective gradient.
Because the algorithm incorporates the squared value of the gradient, it avoids saddle points more easily. In comparison, gradient descent, whether accompanied by momentum or not, initially travels in the direction of the steepest gradient, potentially becoming trapped in local minima. In contrast, AdaGrad progresses along a more direct trajectory, adjusted by the scaled gradients.
A major downside of this algorithm is that computation of the sum of all squared gradients is expensive and slow, rendering the algorithm unusable for modern architectures and datasets. Nevertheless, it lays the foundation for the subsequent algorithms introduced.

Root mean square propagation (RMSProp) This algorithm is based on AdaGrad, but additionally employs an exponentially decaying average to the aggregated squared gradients, allowing the gradients from initial iterations to be disregarded over time. This addresses the issue found in AdaGrad, where the learning rate has the potential to decrease significantly, because the entire gradient history is taken into account.

Definition 3.20 (Root Mean Square Propagation (RMSProp))**.** *In the setting of Definition 3.19. Let $\rho \in [0, 1)$ be the decay rate and $r^{(0)}$ is initialized by zero, the iteration step of the RMSProp algorithm [43, 54] reads*

$$r^{(k+1)} = \rho\, r^{(k)} + (1-\rho)\left(\nabla_\theta \mathbf{E}_k(\theta^{(k)}) \odot \nabla_\theta \mathbf{E}_k(\theta^{(k)})\right) \tag{3.24}$$

$$\Delta^{(k+1)} = -\left(\epsilon \oslash \left(\delta + \sqrt{r^{(k+1)}}\right)\right) \odot \nabla_\theta \mathbf{E}_k(\theta^{(k)}) \tag{3.25}$$

$$\theta^{(k+1)} = \theta^{(k)} + \Delta^{(k+1)}, \tag{3.26}$$

The parameter ρ controls how much of the gradient history is used in each iteration, making the algorithm more efficient and preventing the learning rate from vanishing. According to [43] this algorithm is recognized as an efficient and practical optimization method for deep neural networks.

Adaptive moment estimation (Adam) The final and one of the most robust algorithms presented here is the *adaptive moment estimation*, or more commonly referred to as *Adam*. This approach combines the benefits of gradient descent with momentum and RMSProp's moving averages, leading to fast convergence and robustness to noisy or sparse gradients.

Definition 3.21 (Adam optimizer, [64]). *Let $\rho_1, \rho_2 \in [0, 1)$ be the decay rates and $\delta > 0$ be a small constant and $v^{(0)}, r^{(0)}$ are initialized by zero, the iteration step of the Adam algorithm [43, 64] reads*

$$
\begin{aligned}
g^{(k)} &= \nabla_\theta \mathbf{E}_k(\theta^{(k)}) \\
v^{(k+1)} &= \rho_1\, v^{(k)} + (1-\rho_1) g^{(k)} && (3.27)\\
r^{(k+1)} &= \rho_2\, r^{(k)} + (1-\rho_2)\left(g^{(k)} \odot g^{(k)}\right) && (3.28)\\
\hat{v}^{(k+1)} &= v^{(k+1)} \oslash (1-\rho_1^{k+1}) && (3.29)\\
\hat{r}^{(k+1)} &= r^{(k+1)} \oslash (1-\rho_2^{k+1}) && (3.30)\\
\Delta^{(k+1)} &= -\epsilon\, \hat{v}^{(k+1)} \oslash \left(\delta + \sqrt{\hat{r}^{(k+1)}}\right) \\
\theta^{(k+1)} &= \theta^{(k)} + \Delta^{(k+1)},
\end{aligned}
$$

where $\oslash$, $\odot$ and $\sqrt{\cdot}$ are the element-wise division, multiplication and square root.

Let us consider the individual steps of this algorithm closer. $v^{(k+1)}$ calculates exponential moving averages of the gradient and $r^{(k+1)}$ its squared counterpart; see (3.27), (3.28). $\rho_1, \rho_2 \in [0, 1)$ are hyperparameters which control the decay rates for these averages. These moving averages serve as estimates of the first moment (mean) and the second moment (uncentered variance) of the gradient. These averages are initialized by setting $v^{(0)} = r^{(0)} = 0$, which introduces a bias towards zero, particularly during the early iterations and when the decay rates are small (i.e. ρ_1 and ρ_2 are close to 1). This is why in Equations 3.29, 3.30 a bias correction is performed resulting in estimates $\hat{v}^{(k+1)}$ and $\hat{r}^{(k+1)}$. This bias towards zero is also a problem for RMSProp; however, no bias correction is performed in this algorithm.

RMSProp and Adam are only some of the currently most popular optimization algorithms. Unfortunately, there is no universally best algorithm. According to [43] the user's preference of algorithm lies more with the choice of hyperparameters than the advantages of the individual algorithm.

3.3.3 Loss Functions

When choosing a loss function for a particular application, several properties must be considered. Terven et al. [114] list them as

1. Convexity: Optimization methods based on gradient descent perform better on convex functions.

2. Differentiability: This is desirable for gradient-based optimization; see Definition 3.15.

3. Robustness: A loss function should not be strongly affected by outliers.

4. Smoothness: A continuous gradient without abrupt changes or discontinuities makes optimization easier.

5. Sparsity: For high-dimensional data with a small number of important features, it can be advantageous to use a loss that promotes sparse parameters.

6. Monotonicity: According to [114] a loss function is considered monotonic if the value of the loss function decreases as the prediction approaches the true value. This ensures that the negative gradient always points in the direction of the optimal solution, and thus optimization methods perform well.

This section considers possible loss functions that are applicable for image reconstruction tasks, such as CT, and how they compare to the list of properties. This list is by no means exhaustive, given the existence of numerous loss functions tailored for specific problems like object detection, image classification, segmentation or generation, and many more. For a complete list, we refer to [114]. As mentioned in Section 3.2.1, while differentiability is ideal, algorithms in real-world applications can function with piecewise differentiability.
Let $y_{(i,j)}$ and $\hat{y}_{(i,j)}$, with $i, j \leq 0 < N$ denote the ground truth and the reconstructed CT images, respectively.

Mean Squared Error

The most widely known metric is the *mean squared error (MSE)*, which is defined as

$$\mathrm{MSE}(y, \hat{y}\,) \coloneqq \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \big(y_{(i,j)} - \hat{y}_{(i,j)}\big)^2. \tag{3.31}$$

It calculates the average squared difference per pixel between the reconstructed image and the actual image.
The advantages of the MSE are its simplicity, fast computation, convexity, smoothness, differentiability, and monotonicity in the sense of loss functions. Furthermore, it can be directly interpreted in terms of Hounsfield units. However, it is sensitive to outliers and in terms of image reconstruction limitations include that it tends to overlook specific structures and borders; it shows weak correlation with perceived quality [123]. An example of this is discussed in Section 5.4.

Mean Absolute Error

Another well known metric is the *mean absolute error (MAE)*. The MAE is defined as

$$\mathrm{MAE}(y, \hat{y}) \coloneqq \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left| y_{(i,j)} - \hat{y}_{(i,j)} \right|. \tag{3.32}$$

This loss shares many of the properties of MSE, such as simplicity, fast computation, convexity, and monotonicity. It is differentiable and smooth everywhere, except for $y = \hat{y}$. MAE creates a less smooth loss surface compared to MSE, creating problems for optimization algorithms. In contrast to MSE it is less sensitive to outliers, because larger errors have the same impact as small errors.

3.3.4 Regularization Techniques

The primary goal in ML is to create a model that performs well on previously unseen data, not just training data. When training models with a large amount of parameters, they have the capacity to learn patterns in the training data so effectively that they cannot *generalize* well to new data. The validation error, i.e. the error on previously unseen data is high, whereas the training error is very low. There are many strategies that reduce the validation error, even at the expense of an improved training error. This is referred to as *regularization*. In this section, some regularization methods for neural networks are introduced.

Parameter Norm Penalties

This type of regularization has already been mentioned in the context of forward and backprojection. It introduces a penalty term $\lambda \mathcal{R}(\theta)$ to the objective function, where $\lambda \geqslant 0$ controls the influence of the norm penalty term $\mathcal{R}(\theta)$.

$$\tilde{\mathbf{E}}_D(\theta) = \mathbf{E}_D(\theta) + \lambda \mathcal{R}(\theta), \tag{3.33}$$

During optimization of the objective function $\tilde{\mathbf{E}}_D$, not only the original objective $\mathbf{E}_D$ is minimized for the training data D, but also some measure $\mathcal{R}$ with respect to the parameter θ. When applying this function $\mathcal{R}$ to the parameters, only the weights are regularized. The bias controls only the output of the activation function, whereas the weights are applied directly to the input of the neuron. Regularization of the bias would lead to underfitting [43].

The two most popular penalty functions are the L^2 norm and the L^1 norm.

For L^2 regularization, the objective function reads

$$\tilde{\mathbf{E}}(\theta) = \mathbf{E}(\theta) + \lambda \frac{1}{2}\|w\|_2^2$$

and the gradient descent step according to (3.15) reads

$$\begin{aligned} w^{(k+1)} &= w^{(k)} - \epsilon(\nabla_w \mathbf{E}(\theta) + \lambda\, w^{(k)}) \\ &= w^{(k)} - \epsilon\lambda\, w^{(k)} - \epsilon\nabla_w \mathbf{E}(\theta), \end{aligned}$$

with $\epsilon > 0$ as the learning rate. Thus, in each iteration, the value of $w^{(k)}$ is further reduced by a scaled version of itself. Generally, ϵ and α are chosen as small values; otherwise, the weight update can become unstable.
In the case of L^1 regularization, the objective function reads

$$\tilde{\mathbf{E}}(\theta) = \mathbf{E}(\theta) + \lambda \frac{1}{2}\|w\|_1$$

and the gradient descent step is written as

$$\begin{aligned} w^{(k+1)} &= w^{(k)} - \epsilon(\nabla_w \mathbf{E}(\theta) + \lambda\ \mathrm{sign}(w^{(k)})) \\ &= w^{(k)} - \epsilon\lambda\ \mathrm{sign}(w^{(k)}) - \epsilon\nabla_w \mathbf{E}(\theta). \end{aligned}$$

Contrary to L^2 regularization, where each $w^{(k)}$ is linearly scaled by $\epsilon\lambda$, here, it is a constant factor with the sign of $\mathrm{sign}(w^{(k)})$ and thus L^1 regularization pushes the weights to zero faster than L^2 regularization.

Batch Normalization

This method was originally introduced to reduce the internal covariance shifts in each layer [58]. The authors define this shift as the change in the distribution of network activations resulting from changes in network parameters throughout the training process. To correct for this shift, the activation values of a mini-batch $B = \{x_1, \ldots, x_m\}$ are normalized, where every $x_i \in B$ has dimensions d, thus

$x_i = (x_i^{(1)}, \ldots, x_i^{(d)})$, hence the name *batch normalization* (BN). The mean and variance of the mini-batch are calculated as

$$\mu_B^{(k)} = \frac{1}{m}\sum_{i=1}^{m} x_i^{(k)}$$
$$\sigma_B^{(k)^2} = \frac{1}{m}\sum_{i=1}^{m}(x_i^{(k)} - \mu_B^{(k)})^2$$

and used to shift values $x_i \in B$ through

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)^2} + \delta}},$$

where δ is a small constant added for numerical stability. This transformation changes what the network layer can express. Therefore, two learnable parameters γ and β are introduced, which transform the normalized activation values by applying

$$X_i^{(k)} = \gamma^{(k)}\,\hat{x}_i^{(k)} + \beta^{(k)}.$$

Consequently, the network could learn the parameters γ and β such that $\gamma = \sqrt{\sigma^2}$ and $\beta = \mu$ to recreate the original input x if that were the ideal activation.
Despite batch normalization gaining popularity because of its strong empirical results, the underlying mechanism of the method remains not yet well-understood. The original intention was to reduce covariant shifts [58], however, since then it has been shown that a network with BN, where a noise is introduced after BN layers, still performed better than the original network without BN or noise [103]. This noise was created to simulate a covariant shift, which implies that this shift is not the reason for poor performance of the model. Instead, [103] suggests that BN creates a smoother optimization problem which in turn improves gradient descent-based optimization methods. Finally, [76] provides a mathematical framework to show that batch normalization behaves as a regularizer for neural networks.

Early Stopping

When training large models, a method to prevent overfitting is to simply stop the training when the model performs well on training data and before performance on validation data decreases. This can be accomplished by evaluating the model performance on validation data at the end of each epoch and terminating the training process, once this performance deteriorates, even if the error on the training data still improves. Continuing the training after this point increases the risk that the model

learns patterns from the training data that reflect a noise instead of the underlying data generating distribution. This is analogous to the discrepancy principle applied in RESESOP (2.33) and RESESOP-Kaczmarz (2.38). For the discrepancy principle or early stopping, the iterative method is stopped when a certain criterion is met. In the case of RESESOP, the iteration terminates when the error in f_n, specifically $\|\mathcal{A}^\eta f_n - g^\delta\|$, falls below a predefined noise threshold. Therefore, the quality of f_n is inherently limited by the accuracy of the data and the operator used. In machine learning, the level of noise is unspecified; nevertheless, performance can be empirically assessed through the error of the model applied to validation data. The underlying intuition remains unchanged, only that early stopping depends on statistical estimates of the data distribution.

Algorithm 6 Early Stopping [43]

Initialize: θ network parameters, $i, j = 0,\ v = \infty$ as initial validation loss
Given: $p \in \mathbb{N}$ as patience

while $j < p$ **do** ▷ patience has not been reached and other stopping criteria for training are not met
    run_training(θ) ▷ run training algorithm for n steps
    $i \leftarrow i + n$
    $v' \leftarrow$ validation_loss(θ) ▷ evaluate the model on the validation set
    if $v' < v$ **then** ▷ validation loss decreases
        $j \leftarrow 0$
        $\theta^+ \leftarrow \theta$
        $i^+ \leftarrow i$ ▷ save currently best parameters and training steps
        $v' \leftarrow v$
    else ▷ count consecutive times validation loss does not decrease
        $j \leftarrow j + 1$
    end if
end while
return θ^+, i^+

This algorithm can be further refined by stopping training when the change in validation loss is less than a defined threshold Δ. In this case, the patience counter j increases when $v' \geqslant v$ or $|v' - v| < \Delta$.

For linear models with quadratic error functions, it can be shown that early stopping is equivalent to L^2 regularization [43]. For general error functions, it can be shown that L^2 regularization and early stopping are closely related in terms of regularization properties [70].

There are several other regularization techniques for neural networks, such as data augmentation and dropout [43] that will not be further discussed. However, it is

obvious that building a robust model architecture requires many complex decisions which cannot easily be derived from the type of problem or the data available.

3.4 Deep Learning in the context of CT

In the previous sections, the concepts of CT and machine learning have been introduced individually. In this context, we discuss deep learning approaches for image reconstruction in CT. In Section 3.4.1 different learning techniques are discussed, starting with supervised, unsupervised, and self-supervised learning. We then distinguish between fully learned models and post-processing models.

3.4.1 Learning Methods

Supervised and unsupervised learning have been introduced in Section 3.1.2. We now examine various applications of these learning methods for CT reconstructions, followed by an exploration of various strategies for employing machine learning models in reconstruction tasks.

Although there are research efforts to improve the reconstruction process in CT using unsupervised learning [125, 115, 8], many others are grounded in supervised learning [77, 48, 126, 60]. The learning approach to these reconstruction models is fundamentally different. Unsupervised methods often use other reconstruction methods, such as FBP, due to its fast computation, to support model training [125, 115], while supervised models are based on measurement data and are not influenced by other reconstruction processes. For successful learning, data quality is most crucial. In CT reconstruction, ground truth is not easily available and therefore datasets are often synthetic [60], require a large amount of human intervention [69], or are supplemented by other reconstruction methods, such as FBP [52].
For the purpose of this research, we applied supervised learning techniques. The problem of data generation will be further discussed in Section 5.1.

There have been approaches to *fully learn* solutions for inverse problems using fully connected networks. This has been proved successful by [128] where the AUTOMAP architecture was applied to MRI data. As discussed in Section 3.2, this naive solution leads to enormous parameter spaces. He et al. [49] used a more efficient approach with the iRadonMap network. This model is split into two parts, the first one reproducing the FBP structure and the second one using residual CNN

layers as examined in (3.2.3). The FBP structure has been reproduced by applying a so-called sinusoidal back-projection layer, which capitalizes on the fact that only a few radon-projected points affect a reconstruction point. Therefore, far fewer parameters are needed compared to a fully connected layer [49].

These fully learned models can be based on other processes but are usually *black-box* methods, which means that internal decisions are difficult or impossible to comprehend. This is where *hybrid models* have an advantage. These models are used to enhance other reconstruction methods. In [19] a hybrid deep learning-shearlet framework is used to improve the reconstructions of limited angle CT. This can be interpreted as a type of inpainting, where the missing data, that is, the data that cannot be reconstructed due to missing projection data, is learned by the network.

Post-processing models are utilized to improve the backprojection produced by another method. In CT, a very popular approach is to use learned post-processing models to improve the results of filtered backprojection [98] or other reconstruction techniques [77]. The goal is to reduce the noise and artifacts left in the reconstruction. A neural network is applied to the results of a process, such as the FBP. The result is thus the composition

$$x_{reco} = \mathcal{F}_\theta \circ \mathrm{FBP}(y^\delta), \tag{3.34}$$

where $\mathcal{F}_\theta : X \to X$. Such neural networks are trained to distinguish between reconstruction artifacts and the target image. There are also methods that first apply a preprocessing step to the measurement data. In [121] a filter is learned by a neural network, FBP is then applied to the results, before applying a U-Net architecture as post-processing.

3.5 Unrolled Iterative Methods

The approach to unrolling iterative algorithms originates from Gregor and LeCun [45] who applied this method to the *iterative shrinkage and thresholding algorithm* (ISTA). The goal was to improve the computational efficiency of sparse coding algorithms. ISTA is a method used in compressed sensing, image processing, and signal recovery to identify sparse solutions to linear inverse problems, issues such as image restoration and noise reduction. This method proves exceptionally useful in contexts involving noisy data, where the aim is to reconstruct a signal or image from limited information. The ISTA algorithm reads as in Algorithm 7, where $W \in \mathbb{R}^{n\times m}$, $g \in \mathbb{R}^n$ and $f_0 \in \mathbb{R}^m$ is an initial guess of the reconstruction f. The algorithm

iterates until the change between f_k and f_{k+1} falls below a predefined threshold. The operator h_α is a component-wise shrinkage operator with threshold parameters α, defined as

$$[h_\alpha(u)]_i := \text{sign}(u_i)\max\{|u_i| - \alpha_i, 0\}.$$

Algorithm 7 ISTA [45]

Given: $W \in \mathbb{R}^{n\times m}$, $g \in \mathbb{R}^n$, $L > 0$ largest eigenvalue of W^TW, $\lambda > 0$.
Initialize: $f_0 \in \mathbb{R}^m$
1: **for** $k = 0, \ldots$ **do**
2: $\quad f_{k+1} \leftarrow h_{\frac{\lambda}{L}}\Big(f_k - \frac{1}{L}W^T(Wf_k - g)\Big)$
3: **end for** ▷ Iterate until change in f_{k+1} is below a threshold
4: **return** f_{k+1}

Rearranging the argument of this operation leads to

$$f_{k+1} = h_{\frac{\lambda}{L}}\Big((I - \frac{1}{L}W^TW)f_k + \frac{1}{L}W^Tg\Big).$$

The right side of this equation can be translated to

$$h_\theta\Big(Sf_k + B\Big),$$

where the soft-thresholding operator h_θ is a non-linear activation function and $(I - \frac{1}{L}W^TW)$ is replaced with weights S and $\frac{1}{L}W^Tg$ with B. It becomes obvious how this algorithm can now be interpreted as a neural network with trainable parameters S, B, θ. In the original paper S and B are shared across all iterations, making this a recurrent neural network, called *learned ISTA* or *LISTA*, see Algorithm 8. This interpretation of ISTA has since been developed further. For example, Bubba

Algorithm 8 Learned ISTA [45]

Initialize: $f_0 \in \mathbb{R}^m$ and S, B, θ are trainable weights.
1: **for** $k = 0, \ldots, K-1$ **do**
2: $\quad f_{k+1} \leftarrow h_\theta\Big(Sf_k + B)\Big)$
3: **end for**
4: **return** f_K

et al. [20] have combined this approach with a convolutional architecture in the context of inverse problems. The integration of convolutions offers a considerable decrease in the number of parameters involved. This approach is suitable for limited-angle problems, where a full $360°$ rotation of the object is not possible. In [21], this

approach has been further developed by developing CNN filters that are specialized to mitigate streak artifacts in reconstructions. This considerably reduces the number of learnable parameters while preserving or marginally enhancing the quality of reconstructions from limited-angle and sparse-angle measurements.

Numerous approaches since ISTA have shown that algorithm unrolling can enhance performance in various applications [71, 59, 3].
For example, [29] has applied the unrolling methodology to a gradient descent scheme using fields of experts (FoE) as a regularizer [100]. Considering a regularized objective function

$$f = \arg\min_f E(f) \tag{3.35}$$

$$= \arg\min_f \frac{\lambda}{2}||Af - g||_2^2 + R(f), \tag{3.36}$$

with regularization term $R(f)$. FoE acts as a regularizer

$$R(f) = \sum_{l=1}^{L} \phi_l(G_l f),$$

where G_l is a transform matrix, acting as a linear filter on f and ϕ_l is a potential function. Under the assumption that FoE is differentiable, a gradient scheme can be applied to (3.36). An iteration step of the gradient descent scheme then reads

$$f_{k+1} = f_k - \epsilon \frac{\partial E}{\partial f_k}, \tag{3.37}$$

$$\frac{\partial E}{\partial f_k} = \lambda_k A^T(Af_k - g) + \sum_{l=1}^{L} (G^l)^T \gamma^l(G^l f_k), \tag{3.38}$$

where $\gamma(\cdot) = \phi'(\cdot)$ and learning rate $\epsilon > 0$. The intuition behind FoE is that several "experts", i.e. linear filters and non-linear functions, work together to detect different patterns in the data.

Line 2 in Algorithm 9 then describes the regularized gradient descent step, with γ_k^l as the gradient of a potential function in the l-th regularization term of the k-th iteration. The algorithm iterates until f_{k+1} satisfies a stopping criterion.

The term $\sum_{l=1}^{L}(G_k^l)^T \gamma_k^l(G_k^l f_k)$ originating from FoE can be interpreted as a classical CNN, where G represents a CNN filter and γ a non-linear activation function. The authors in [100] point out that FoE has the ability to learn universal priors for different applications, while a CNN implementation of FoE is limited to the data set on which it is trained.

Algorithm 9 Gradient Descent with FoE [29]

Given: $\lambda_k > 0$, $A : X \to Y, g \in Y$.
Initialize: $f_0 \in X$
1: **for** $k = 0, \dots$ **do**
2: $\quad f_{k+1} \leftarrow f_k - \left(\lambda_k A^T(Af_k - g) + \sum_{l=1}^{L}(G_k^l)^T \gamma_k^l(G_k^l f_k)\right)$
3: **end for**
4: **return** f_{k+1}

In the unrolled network, the FoE term was replaced by a CNN $\Psi_{\theta_k}(f_k)$ as shown in Algorithm 10. In the *LEARN* network the parameter λ_k is included in the set of

Algorithm 10 LEARN [29]

Given: $\lambda_k > 0$, $A : X \to Y, g \in Y$.
Initialize: $f_0 \in X$
1: **for** $k = 0, \dots K-1$ **do**
2: $\quad f_{k+1} \leftarrow f_k - \lambda_k A^T(Af_k - g) - \Psi_{\theta_k}(f_k)$
3: **end for**
4: **return** f_K

learned parameters $\Theta_k := \{\lambda_k, \theta_k\}$.

This network inspired the work in [59], where a similar approach was taken to unroll the superiorized version of the simultaneous algebraic reconstruction technique.
This algorithm is also based on a gradient descent scheme applied to the objective function in (3.36), with $\lambda = 1$. In the superiorization step, the gradient of the regularizer R is replaced by the normalized negative gradient of R. This step adds a perturbation to the iterative algorithm to improve its performance.
The algorithm then has a two-step iteration process, as can be seen in Algorithm 11. The iteration ends, when a stopping criterion is satisfied.

Algorithm 11 Superiorized SART [59]

Given: D and M are diagonal scaling matrices, R is a regularizer. $A : X \to Y, g \in Y$.
Initialize: $f_0 \in X$
1: **for** $k = 0, \dots$ **do**
2: $\quad f_k^+ \leftarrow f_k - \lambda_k D A^T M(Af_k - g)$
3: $\quad f_{k+1} \leftarrow f_k^+ - \beta_k \dfrac{\nabla R(f_k^+)}{\left\|\nabla R(f_k^+)\right\|}$
4: **end for**
5: **return** f_{k+1}

Here, D and M are diagonal scaling matrices, defined as the sums of columns and rows of A, respectively, and R is a regularizer. Line 2 performs a gradient descent step, line 3 then applies a superiorization to the previous results f_k^+.

Similarly to LEARN, the *unrolled SART* network replaces the regularization term with a neural network Ψ_{θ_k} in each iteration k, see Algorithm 12. In addition to the network parameters θ_k, λ_k is also a trainable parameter.

Algorithm 12 Unrolled superiorized SART [59]

Given: D and M are diagonal scaling matrices, $A : X \to Y, g \in Y, \lambda_k$ and θ_k are trainable parameters.
Initialize: $f_0 \in X$
1: **for** $k = 0, \dots, K-1$ **do**
2: $\quad f_k^+ \leftarrow f_k - \lambda_k D A^T M(Af_k - g)$
3: $\quad f_{k+1} \leftarrow f_k^+ - \Psi_{\theta_k}(f_k^+)$
4: **end for**
5: **return** f_K

Another iterative reconstruction technique is the primal dual hybrid gradient algorithm (PDHG) [27] which has been transformed into a *learned PDHG* and a *learned Primal-Dual* algorithm [3].

In large-scale problems, gradient descent requires the regularizer (used to ensure sparsity or smoothness) to be differentiable. This condition limits its applicability for optimization tasks involving non-smooth objectives. Proximal methods, unlike gradient descent, handle non-smooth objective functions. They incorporate a proximal step instead of a gradient step, which is specifically designed for problems with non-differentiable terms. The proximal operator is defined as

$$\text{prox}_{\tau\mathcal{G}}(f) = \underset{f' \in X}{\arg\min} \Big(\mathcal{G}(f') + \frac{1}{2\tau} \|f' - f\|_X^2 \Big). \tag{3.39}$$

Essentially, it modifies the solution by solving a regularized minimization problem.

Many proximal operators do not have a closed-form solution, which means that they cannot be computed directly in a single step. This makes traditional proximal methods less efficient for certain large-scale problems. Primal-dual algorithms, like PDHG, are designed to overcome the challenge of non-closed-form proximal operators. They work by decomposing the problem into simpler subproblems and iteratively solving these subproblems, using updates to both primal and dual variables.

By breaking the optimization process into smaller, manageable pieces, primal-dual methods ensure that each iteration improves both the primal and dual solutions. This iterative approach is computationally efficient and particularly suitable for large datasets.

In [3], this approach is modified to solve minimization problems of the form

$$\min_{f \in X} \Big(\mathcal{F}(\mathcal{K}(f)) + \mathcal{G}(f) \Big), \tag{3.40}$$

where $\mathcal{K} : X \to U$ is a (non-) linear operator and $\mathcal{G} : X \to \mathbb{R}$ and $\mathcal{F} : U \to \mathbb{R}$ are functionals on the primal space X and the dual space U, respectively.

The specialized form of the minimization problem in (3.40), namely, for inverse problems can be written as

$$\min_{f \in X} \Big(\mathcal{L}(\mathcal{A}(f), g) + \lambda \mathcal{R}(f) \Big), \text{ for a fixed } \lambda \geq 0,$$

where $\mathcal{F} \coloneqq \mathcal{L}(\cdot, g), \mathcal{K} \coloneqq \mathcal{A}$ and $\mathcal{G} \coloneqq \mathcal{R}$.

The algorithm then applies the proximal operator in (3.39) alternatively to the Fenchel conjugate $\mathcal{F}^*$ of $\mathcal{F}$ and $\mathcal{G}$. The Fenchel conjugate is defined as

$$\mathcal{F}^*(y) = \sup_{x \in \text{dom}(\mathcal{F})} (y^T x - \mathcal{F}(x)).$$

The Fenchel conjugate plays a role in establishing the dual problem and serves as a connection between primal and dual formulations [11].

Algorithm 13 Non-linear PDHG [3]

Given: $\sigma, \tau > 0$ s.t. $\sigma\tau||\mathcal{K}||^2 < 1$, $\gamma \in [0, 1]$ and $f_0 \in X, h_0 \in U$.

1: **for** $k = 0, \ldots$ **do**
2: $\quad h_{k+1} \leftarrow prox_{\sigma\mathcal{F}^*}(h_k + \sigma\mathcal{K}(\bar{f}_k))$
3: $\quad f_{k+1} \leftarrow prox_{\tau\mathcal{G}}(f_k - \tau[\partial\mathcal{K}(f_k)]^*(h_{k+1}))$
4: $\quad \bar{f}_{k+1} \leftarrow f_{k+1} + \gamma(f_{k+1} - f_k)$
5: **end for**
6: **return** f_{k+1}

Algorithm 13 shows the 3 steps for each iteration, where $\sigma, \tau > 0$ are the step lengths so that $\sigma\tau||\mathcal{K}||^2 < 1$, $\gamma \in [0, 1]$ and $[\partial\mathcal{K}(f_k)]^* : U \to X$ is the adjoint of the derivative of $\mathcal{K}$ in f_k. When a stop criterion is fulfilled, the iteration is stopped.
According to [3], the authors argue that it is possible to substitute proximal operators with those that are not necessarily proximal operators.

The learned PDHG algorithm, see Algorithm 14, replaces both proximal operators with parameterized operators, that is, network layers with trainable parameters.

Algorithm 14 Learned PDHG [3]

Initialize: $f_0 \in X$, $h_0 \in U$

for $k = 0, \dots, K-1$ **do**
  $h_{k+1} \leftarrow \Gamma_{\theta^d}(h_k + \sigma\mathcal{K}(\bar{f}_k), g)$
  $f_{k+1} \leftarrow \Lambda_{\theta^p}(f_k - \tau[\partial\mathcal{K}(f_k)]^*(h_{k+1}))$
  $\bar{f}_{k+1} \leftarrow f_{k+1} + \theta(f_{k+1} - f_k)$
end for
return f_K

The dual and primal proximals are encoded with parameters θ^d and θ^p, respectively, θ is an overrelaxation parameter. Including σ and τ, all these parameters are inferred from the training data.
This new algorithm adheres strictly to the structure of the PDHG algorithm. Going a step further, the authors provide only the individual components of the input terms to the networks in the learned Primal-Dual algorithm, see Algorithm 15.

Algorithm 15 Learned Primal-Dual [3]

Initialize: $f_0 \in X^{N_{primal}}$, $h_0 \in U^{N_{dual}}$

for $k = 0, \dots, K-1$ **do**
  $h_{k+1} \leftarrow \Gamma_{\theta^d_{k+1}}(h_k, \mathcal{K}(f_k^{(2)}), g)$
  $f_{k+1} \leftarrow \Lambda_{\theta^p_{k+1}}(f_k, [\partial\mathcal{K}(f_k^{(1)})]^*(h_{k+1}^{(1)}))$
end for
return $f_K^{(1)}$

In addition to giving the network the freedom to choose how to combine the primal and dual iteration steps, the authors also exploit CNN properties, namely channels. $f^{(1)}$ and $f^{(2)}$ refer to the first and second channels of the convolutional layer output $f = [f^{(1)}, f^{(2)}, \dots, f^{(N_{primal})}] \in X^{N_{primal}}$. Equivalently, $h \in U^{N_{dual}}$ has multiple channels, [3] refers to this as "memory". Finally, the first channel of f contains the output of the network.

Similarly to previous work, Adler et al. [2] suggested an approach to solve ill-posed inverse problems using a simple gradient descent scheme. The authors define an error functional $E : X \to \mathbb{R} \coloneqq d(f, f_{true})$, with $d : X \times X \to \mathbb{R}$ as a distance functional. As f_{true} is generally not available, E needs to be approximated. As previously established by earlier methods, the approximation

$$\arg\min_{f \in X} E(f) \approx \arg\min_{f \in X} \Big(\mathcal{L}(\mathcal{T}(f), g) + \lambda \mathcal{S}(f) \Big)$$

is used, where $\mathcal{L}$ is a data log-likelihood functional, $\mathcal{T} : X \to Y$ is the forward operator and $\mathcal{S} : X \to \mathbb{R}$ is a regularization functional with parameter λ. The corresponding gradient descent iteration step then reads as

$$f_{k+1} = f_k - \sigma\Big(\nabla[\mathcal{L}(\mathcal{T}(\cdot), g)](f_k) + \lambda[\nabla\mathcal{S}](f_k)\Big),$$

assuming the likelihood $\mathcal{L}$ and regularizer S are differentiable. The gradient term can then be translated into a neural network with inputs $\nabla[\mathcal{L}(\mathcal{T}(\cdot), g)](f_k), \nabla\mathcal{S}(f_k)$, as well as f_k and s_k. The authors suggest to improve the convergence rate of the gradient descent scheme by providing additional information from previous iteration steps, again named *memory*, defined as $s \in X^M$. The inclusion of the regularization parameter λ and the step length σ as part of the network parameters θ means that explicit selection of these values is unnecessary.

Algorithm 16 Partially learned gradient descent [2]

Initialize: $f_0 \leftarrow \mathcal{T}_{FBP}$.
1: initialize $s_0 \in X^M$.
2: **for** $k = 0, \dots, K-1$ **do**
3: $\quad (s_{k+1}, \Delta f_{k+1}) \leftarrow \Lambda_\theta\Big(s_k, f_k, \nabla[\mathcal{L}(\mathcal{T}(\cdot), g)](f_k), \nabla\mathcal{S}(f_k)\Big)$
4: $\quad f_{k+1} \leftarrow f_k + \Delta f_{k+1}$
5: **end for**
6: **return** f_K

The power of CNNs lies in their ability to extract value from neighborhoods of pixels. Generally, a pixel value is in some relation to its neighboring pixels, which is why CNNs employ kernels to utilize this relationship.
However, the ray transform in CT creates different relationships among pixels. All phantom pixels on the line of a ray are transformed to provide a single-pixel value in the sinogram. Therefore, the authors argue that this line-neighborhood can not be exploited with a traditional CNN kernel, and the architecture thus requires an explicit forward operator to make this connection.
The forward operator was defined as a line integral $\mathcal{P} : X \to Y$ and depending on the type of data, different loss functions were used. For example, their ellipses data set required a squared L^2 norm $\mathcal{L}(\cdot, g) \coloneqq \frac{1}{2}||\cdot - g||_Y^2$, leading to

$$\nabla_f[\mathcal{L}(\mathcal{P}(\cdot), g)](f) = \mathcal{P}^*(\mathcal{P}(f) - g).$$

For each iteration of the unrolled network, both the gradient of the loss as well as the gradient of the regularizer are calculated. The article continues to include a brief empirical comparison of the performance of the network with and without gradients provided and draws the conclusion that the gradients boost performance. However,

it is not surprising that the network performs better with more information provided. A thorough comparison would require adjusting the architecture to give the network the opportunity to compensate for the withheld gradients.

In [102] the iterative RESESOP method (see Section 2.4.1) has been partially unrolled. Considering (2.27), the value $t_{n,i} \in \mathbb{R}$ regulates the step size of each search direction in each iteration. Feinler et al. [102] proposes to learn this step size in combination with a regularization term at each iteration. The networks are individually trained at every iteration.

Given our exploration of various unrolled networks, the general principles of algorithm unrolling can be easily formalized.
Unrolled iterative methods are based on iterative functions $\Phi_k : X \times Y \to X$, which are partially or fully replaced by neural network layers Ψ_{θ_k}.
Assuming a full replacement of Φ_k and a fixed number of iterations K, the entire network is trained end to end to approximate the composition of Φ_0 to Φ_{K-1} by

$$\begin{aligned} f_K &= \Phi_{K-1} \circ \Phi_{K-2} \circ ... \circ \Phi_0(f_0, g) \\ &\approx \Psi_{\theta_{K-1}} \circ \Psi_{\theta_{K-2}} \circ ... \circ \Psi_{\theta_0}(f_0, g), \end{aligned}$$

where $f_0 \in X$ is some initial guess, g is the measurement data and θ_i for $i = 0, \dots K-1$ are trainable parameters.
In the case of a partial replacement of the iteration step Φ_k, as seen in most prior examples, the iteration can be approximated by

$$f_K \approx (\Psi_{\theta_{K-1}} \circ \hat{\Phi}_{K-1}) \circ (\Psi_{\theta_{K-2}} \circ \hat{\Phi}_{K-2}) \circ ... \circ (\Psi_{\theta_0} \circ \hat{\Phi}_0(f_0, g)),$$

where $\hat{\Phi}_k$ represents the non-network part of the iteration.

Part II

Methodology and Numerical Results

4 Unrolled Neural Network

In Section 3.5, it was discussed how different iterative methods can be translated into unrolled networks. In the context of this dissertation, two architectures have been developed that apply unrolling to the SESOP method (see Section 2.4.1). These architectures implement different approaches of unrolling SESOP. First, the fully learned operator network (see Section 4.1) is presented. This network separates each SESOP iteration into two parts, one to process the measurement data g^η and one to process the search directions. The second architecture is supported by the Radon transform and will be referred to as the Radon network, see Section 4.2. In this architecture, each search direction is processed in conjunction with the measurement data. The reconstruction is performed by the Radon transform, which is an inexact operator in the context of dynamic inverse problems. The network then attempts to transform the Radon reconstructions to learn the unperturbed reconstructions.
Both architectures are convolutional networks. As discussed in Section 3.2.2, such networks offer an effective framework for image data because the weights are sparse and shared.

Previously, it was suggested that an architecture based on convolutional kernels applied to CT measurement data is problematic [2]. A single pixel in the measurement data depends on the ray transform of all pixels on a line through the object. Thus, there are non-local dependencies encoded in the measurement. As seen in Figure 4.1, the pixels $A_{1,2}$ and $B_{1,2}$ in the measurement data (Figure 4.1b) originate from the line transform over the respective lines in the scan geometry (Figure 4.1a). The four pixels are therefore in a relationship with all pixels intersected by the lines through the scanned object, highlighted in purple. The measurement data therefore contain a lot of information from the scanned object; however, we argue that the pixels in the scanned object are still in a spatial relationship. It is correct that this spatial relationship is not as close as for most applications of CNNs; however, there is a relationship that the network might gain information from.

In the study [77], I collaborated with colleagues from the University of Bremen and the University of Göttingen to examine the potential of various post-processing architectures when applied to the filtered back projection, the Dremel method [35],

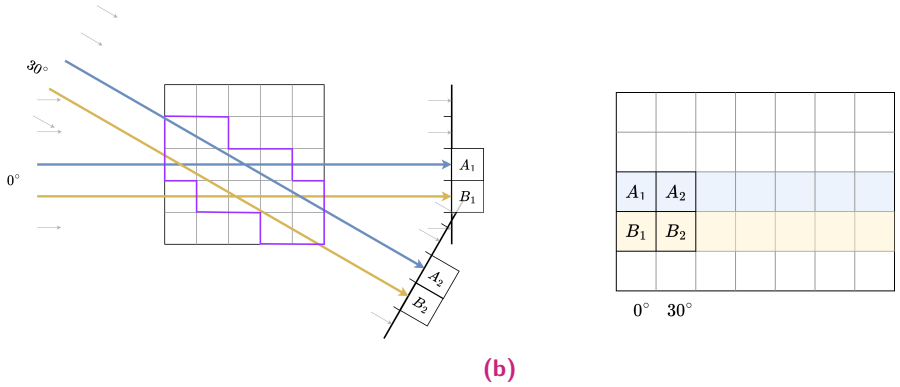


Fig. 4.1.: (a) Parallel beam geometry with two highlighted rays visualized for two angles:$0°$ and $30°$. On the right is the detector shown with five detector points. All pixels highlighted in purple have an influence on the values collected in $A_{1,2}$ and $B_{1,2}$ (b) The sinogram for this geometry is shown with two highlighted detectors. Pixels A_1 and B_1 correspond to the values of the line integrals over the blue and yellow rays for the $0°$ angle. A_2 and B_2 correspond to the rays for the $30°$ angle. Applying a 2×2 kernel to this, would imply grouping those pixels into a neighborhood which is related to the pixels in the purple enclosure.

and the RESESOP-Kaczmarz technique. The post-processing methods of interest where the well-known *U-Net*, the *conditional invertible U-Net* (CiUNet), the *conditional invertible neural network* (CiNN), a variations of those networks using residual layers (CiUNetRes, CiNNRes). For more details on these networks, see [34, 38]. Conditional invertible networks demonstrate promising results for low-dose medical CT data, as noted in [34, 68]. Nevertheless, applying these networks to dynamic CT data represented an intriguing avenue for ongoing research. My contributions to the article [77] lie in providing the dynamic data set (see Section 5.1.1) and an implementation of the RESESOP-Kaczmarz method. The evaluation showed that the best combination of reconstruction and post-processing is RESESOP-Kaczmarz with a U-Net. The RESESOP-Kaczmarz method depends on the availability of operator inexactness η, which has previously been recognized to be a significant challenge to acquire (see Section 2.3.3). Therefore, overestimation and underestimation of η by 20% were also evaluated and still outperformed other combinations. Without post-processing, the results of RESESOP-Kaczmarz and Dremel were comparable. The U-Net has been known to perform well in many problems [77, 60, 59], however, the combination of both methods exceeded expectations.

These results inspired further exploration of the RESESOP-Kaczmarz method. An unquestionable downside of the method is its computational time requirements. The algorithm iterates through both the number of scan angles and the number of detector points per angle sequentially. Through these discretizations, the problem is separated into many sub-problems, which specifically incorporate local modeling

errors. This leads to a complex structure that makes it possible to deal with very localized errors, that is, η is available for scan angles and detector points. Section 2.4.3 goes into further detail on this method. When these localized errors are not available, splitting into such fine sub-problems is excessive. Nonetheless, the underlying SESOP structure of this method provides a promising foundation, where the search directions divide each iteration into sub-problems without limiting the overall algorithm's computational efficiency as substantially.
Taking into account Sections 3.5 and (2.27), it is obvious that the SESOP method can be unrolled into a neural network. Therefore, we believe that SESOP is a promising method for further investigation in the context of unrolling algorithms. Feinler et al. [102] introduced a learned RESESOP method, which learns the SESOP step size, that is, the parameter t in (2.27), and a regularization term in every iteration. The architectures presented in this chapter are fully learned, that is, the network is provided with input data, and all iterations are simulated by the network. The output of these networks are the finished reconstructions.
Furthermore, it is important to recognize that unrolled methods do not preserve any convergence or regularization guarantees the original iterative method might have. A comparison between iterative algorithms and unrolled networks with a fixed number of iterations is difficult. Unrolled network architectures are at most guided by the underlying iterative methods.

The following sections will demonstrate two architectures that implement a fully learned unrolling process.

4.1 Fully Learned Operator Network

Let us again consider the SESOP equation (2.27) using the forward operator $\mathcal{A}$ and a fixed index set $I_n := \{n-1, n\}$ consisting of the previous two iterations. The selection of I_n allows for a concise explanation; however, it is generally adaptable and will be explored further in this chapter. Please refer to Section 2.4.1 for a complete explanation of the SESOP method.

$$f_{n+1} = f_n - \sum_{i \in \{n, n-1\}} t_{n,i} \mathcal{A}^* w_i, \tag{4.1}$$

where g^η are the measurement data created by an operator perturbed by an unknown deformation model and $w_i = \mathcal{A} f_i - g^\eta$.

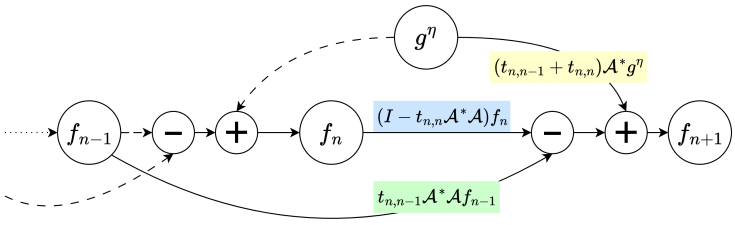
Applying w_i and writing the search directions explicitly, we arrive at

$$\begin{aligned} f_{n+1} &= f_n - t_{n,n}\mathcal{A}^*(\mathcal{A}f_n - g^\eta) - t_{n,n-1}\mathcal{A}^*(\mathcal{A}f_{n-1} - g^\eta) \\ &= f_n - t_{n,n}\mathcal{A}^*\mathcal{A}f_n + t_{n,n}\mathcal{A}^*g^\eta - t_{n,n-1}\mathcal{A}^*\mathcal{A}f_{n-1} + t_{n,n-1}\mathcal{A}^*g^\eta \\ &= (I - t_{n,n}\mathcal{A}^*\mathcal{A})f_n - t_{n,n-1}\mathcal{A}^*\mathcal{A}f_{n-1} + (t_{n,n} + t_{n,n-1})\mathcal{A}^*g^\eta \end{aligned} \tag{4.2}$$

The goal is to translate (4.2) into a neural network architecture, where the number of unrolled iterations n and the number of search directions $|I_n|$ are parameters. The network should approximate the true inverse operator $\mathcal{A}^* = \mathcal{A}_\Gamma^*$. Reconstruction algorithms in Section 2.4, such as RESESOP and RESESOP-Kaczmarz, employ the approximated operator $\mathcal{A}^\eta$, because the true operator is not available. However, the objective of the network architecture is to approximate the true operator.
This architecture has been evaluated in the context of discrete CT. Therefore, $\mathcal{A}$ has a matrix representation, and thus $\mathcal{A}^* = \mathcal{A}^T$. For a more general notation, we continue to write $\mathcal{A}^*$.

In Figure 4.2 we can see a diagram representing (4.2) for iterations f_n to f_{n+1}. The interesting parts of the iteration step are the three highlighted sections. The terms highlighted in blue and green are the back-projections f_n and f_{n-1}, respectively, and the yellow term uses noisy image data g^η. A neural network would have to learn these terms in order to reconstruct according to the structure of the SESOP algorithm. To learn these three parts, we need to build blocks of neural network layers representing each of them; this will be discussed further in Section 4.1.1.

Fig. 4.2.: Graphic representation of RESESOP method as a neural network.



The general structure of (4.2) can be simplified as shown in Figure 4.3. Each iteration step uses the previous two iteration outputs f_n and f_{n-1} as well as the perturbed measurement data g^η. The first iteration receives f_0 twice as there is no earlier reconstruction. The logic of the iteration steps is captured within the gray rectangles and is visualized in detail in Figure 4.4.

Each input of the iteration is processed in its own set of CNN layers. The structure is taken directly from (4.2). The intention of each CNN block is to transform their

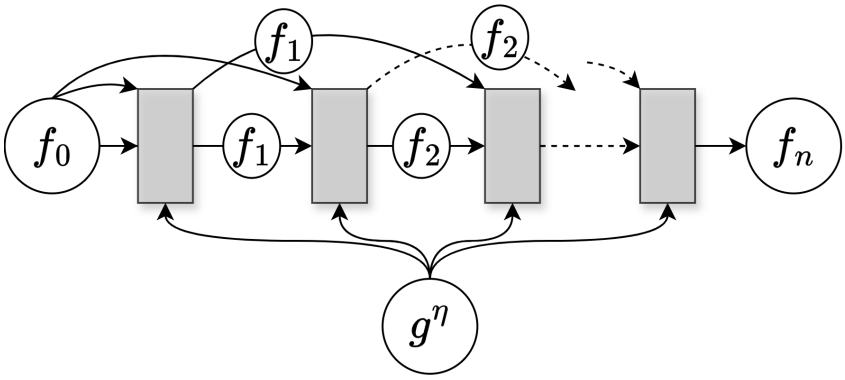


Fig. 4.3.: Overall structure of SESOP iteration for two search directions.

respective inputs. At this point, it is important to note that these transformations are not forced in the network. The layers can learn their own interpretation of the three terms. However, for now, it is assumed that the three network blocks learn their respective transformations according to (4.2) as shown in the diagrams.

For each input, the network block needs to learn the respective component of the SESOP iteration step. The weights of the CNN layers are trained to reflect these transformations. The measurement data g^η are transformed into $(t_{n,n} + t_{n,n-1})\mathcal{A}^* g^\eta$. The input f_n is transformed to $t_{n,n}\mathcal{A}^*\mathcal{A} f_n$ to create the term $(I - t_{n,n}\mathcal{A}^*\mathcal{A})f_n$, and the other search direction f_{n-1} is transformed into $t_{n,n-1}\mathcal{A}^*\mathcal{A} f_{n-1}$.

Returning to (4.2) with two search directions, the individual components have a similar structure. Here, a new notation is introduced: collections of network layers $\mathcal{N}^{\text{sd}}$ and $\mathcal{N}^{\text{d}}$ for the search directions and the data term, respectively. Each such $\mathcal{N}$ has individual weights θ_n^i, where $i = 1, 2$ indicates the search direction and $i = 0$ the data term, and n defines the iteration step. Note that for $\mathcal{N}^{\text{sd}}_{\theta_n^1}$ and $\mathcal{N}^{\text{sd}}_{\theta_n^2}$ the architecture is identical, but the weights are different. Similarly, in each iteration n the architecture of the network layers is the same, but the weights are unique for each i and n.

$$\begin{aligned} f_{n+1} &= f_n \quad -t_{n,n}\mathcal{A}^*\mathcal{A} f_n \quad -t_{n,n-1}\mathcal{A}^*\mathcal{A} f_{n-1} \quad +(t_{n,n} + t_{n,n-1})\mathcal{A}^* g^\eta \\ &\approx f_n \quad -\mathcal{N}^{\text{sd}}_{\theta_n^1}(f_n) \quad -\mathcal{N}^{\text{sd}}_{\theta_n^2}(f_{n-1}) \quad +\mathcal{N}^{\text{d}}_{\theta_n^0}(g^\eta) \end{aligned} \tag{4.3}$$

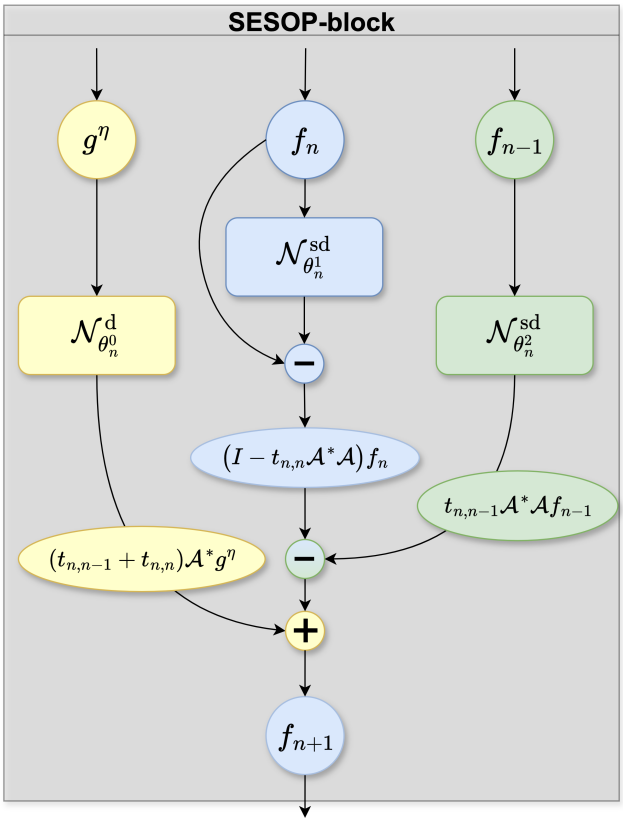


Fig. 4.4.: The SESOP block structure includes 3 CNN parts, one for each search direction and one for the measurement data.

These network blocks then approximate the search directions and data term as follows:

$$\mathcal{N}^{\mathrm{sd}}_{\theta^1_n}(f_n) \approx \omega^1_n \mathcal{A}^*\mathcal{A}(f_n) \quad , \omega^1_n \approx t_{n,n} \tag{4.4}$$

$$\mathcal{N}^{\mathrm{sd}}_{\theta^2_n}(f_{n-1}) \approx \omega^2_n \mathcal{A}^*\mathcal{A}(f_{n-1}), \ \omega^2_n \approx t_{n,n-1} \tag{4.5}$$

$$\mathcal{N}^{\mathrm{d}}_{\theta^0_n}(g^\eta) \approx \omega^0_n \mathcal{A}^*(g^\eta) \quad , \ \omega^0_n \approx t_{n,n} + t_{n,n-1} \tag{4.6}$$

The SESOP search directions always consist of a scalar value $t_{n,i}$ that depends on the iteration step n and the search direction $i \in I_n$. In the above equations, these scalars have been summarized as w^i_n with $i \in \{0, 1, 2\}$ and n being the iteration index. These scalars are not explicitly reconstructed by the network. The scalar of the data term, see (4.6), is defined slightly differently. In the original SESOP step the scalar is defined as the sum of the search direction scalars, $t_{n,n} + t_{n,n-1}$. However, since the structure of our proposed network is not designed to directly learn the values of $t_{n,i}$, the network is given the flexibility to derive the required value $\omega^0_n \approx t_{n,n} + t_{n,n-1}$. Here, we follow the lead of papers such as [3, 29, 59], where the networks have

been given certain freedoms within the structured algorithm.
The network blocks $\mathcal{N}^{\text{sd}}_{\theta^1_n}$ and $\mathcal{N}^{\text{sd}}_{\theta^2_n}$ always receive the reconstructions of previous iterations, f_n and f_{n-1}, as input. However, $\mathcal{N}^{\text{d}}_{\theta^0_n}$ transforms the input g^η, which is not dependent on the iteration step n. Therefore, it is possible to reduce the overall size of the unrolled network part by breaking this network block apart. From (4.2) it is evident that the data term depends only on the iteration step in the term $\omega^0_n \approx (t_{n,n} + t_{n,n-1})$. The result of the adjoint operator applied to the perturbed data, that is, $\mathcal{A}^* g^\eta$, can therefore be reconstructed outside the iteration block to reduce computational costs. The network layers inside the iteration block then only adjust for the ω^0_n component. The network block for the data term can, therefore, be split as follows:

$$\begin{aligned} \mathcal{N}^{\text{d}}_{\theta^0}(g^\eta) &\approx \mathcal{A}^* g^\eta =: \hat{f}^g \\ \mathcal{N}^{f}_{\theta^0_n}(\hat{f}^g) &\approx (t_{n,n} + t_{n,n-1})(\hat{f}^g) \\ \mathcal{N}^{f}_{\theta^0_n} \circ \mathcal{N}^{\text{d}}_{\theta^0}(g^\eta) &\approx (t_{n,n} + t_{n,n-1}) \mathcal{A}^* g^\eta. \end{aligned} \tag{4.7}$$

$\mathcal{N}^{\text{d}}_{\theta^0}$ is computed outside the iteration block once and $\mathcal{N}^{f}_{\theta^0_n}$ refines its output further within each iteration.

A graphic representation of this can be found in Figure 4.5. Note that the network blocks $\mathcal{N}^{\text{d}}_{\theta^0}$, $\mathcal{N}^{f}_{\theta^0_n}$, $\mathcal{N}^{\text{sd}}_{\theta^1_n}$ and $\mathcal{N}^{\text{sd}}_{\theta^2_n}$ are not trained individually, but together in one network. The complete network will be labeled $\mathcal{F}_\theta(g^\eta, f_0)$, where θ are the parameters, g^η are the noisy measurement data, and f_0 is an initial reconstruction. In the following, variations on this architecture are discussed. To simplify the notation, f_0 might be omitted.

Another core component of the SESOP iteration are the operators. For search directions the adjoint and direct operators $\mathcal{A}^*\mathcal{A}$ are needed, for the data term only the adjoint operator $\mathcal{A}^*$ is important. What this means for search directions is that $\mathcal{A}^*\mathcal{A}$ projects the current reconstruction f_n in two steps: $\mathcal{A} : X \to Y$ and $\mathcal{A}^* : Y \to X$. Intuitively speaking, the current reconstruction f_n is transformed into projection data in Y by $\mathcal{A}$ and then again backprojected into X through the adjoint $\mathcal{A}^*$. Let us define this two-step projection of $f \in X$ as

$$\hat{f} \coloneqq \mathcal{A}^*\mathcal{A} f \in X.$$

It is therefore much cheaper for the network to learn $\hat{f}$ given f, than it would be to learn $\mathcal{A}f = g$ and $\mathcal{A}^* g = \hat{f}$.
In contrast to the search direction term, the data term $\mathcal{A}^* g^\eta$ is a backprojection from Y to X. Mathematically speaking, this computation is less expensive as only one

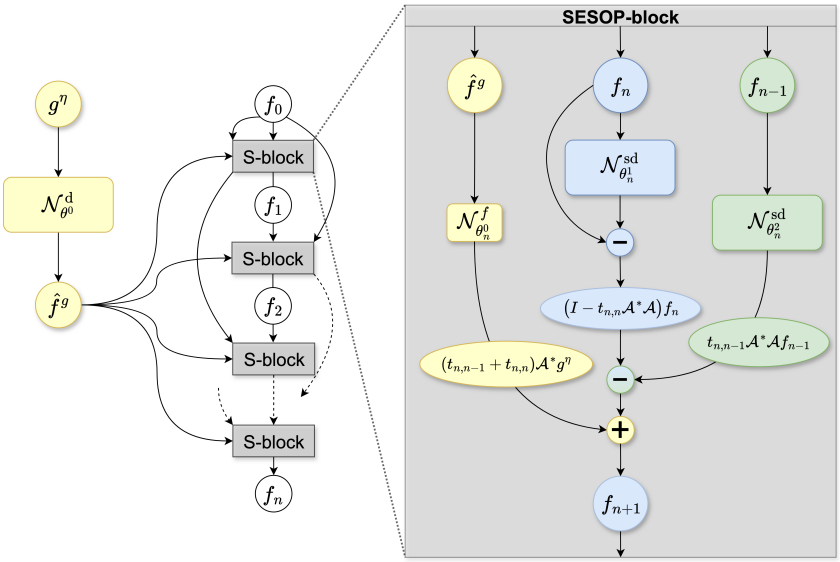


Fig. 4.5.: The SESOP-block structure includes three CNN blocks, one for each search direction and one single CNN layer for the measurement data, visualized by a smaller rectangle. Outside of the iteration block the main transformation for the measurement data is performed.

operator is applied; however, for the neural network, this means that it now has to learn the backprojection by finding a combination of weights that approximate a complete reconstruction of the data g^η. This is a much more ambitious task for the network than learning $\hat{f}$ given f.
There are different solutions to this issue.

1. Firstly, the model could be created so that the reconstruction of g^η is, in fact, fully learned by the network. In this case, the network receives input data g^η and uses a set of CNN layers to approximate $\mathcal{A}^* g^\eta$. This approach is the most complex, but gives the model the most freedom in reconstruction. In the following, this will be labeled $\mathcal{F}_\theta^{\mathcal{A}}(g^\eta)$, to emphasize that the operator $\mathcal{A}$ is fully learned by the model and is also referred to as the *fully learned operator* (*FLO*) model.

2. An additional approach involves employing an alternative reconstruction method within the neural network. This means that instead of g^η, the network receives the results of a different reconstruction method $\mathcal{T}(g^\eta)$. We chose a computationally efficient standard approach, that is FBP. Instead of learning from perturbed data g^η, the network learns from an artifact-afflicted reconstruction. In this case, the model does not have to learn the back-projection step but must compensate for the artifacts created by a reconstruction method that does not take into account the dynamic problem. Furthermore, this changes

the unrolled network from a fully learned approach to a post-processing one because it learns from a reconstruction and not the projection data. This solution will be labeled $\mathcal{F}_\theta \circ \mathrm{FBP}(g^\eta)$. Compared to the previous version, the model does not need to learn the operator.

3. An alternative approach is to utilize the adjoint Radon transform rather than employ a reconstruction algorithm. The discrete Radon transform and its adjoint can be applied to the input data g^η. This approach is similar to the previous point; however, the characteristics of the adjoint Radon in comparison to an FBP reconstruction vary, as the adjoint Radon does not apply a filter. Consequently, this could potentially affect the performance of the network. This version will be labeled $\mathcal{F}_\theta \circ \mathcal{R}^*(g^\eta)$.

Note that *fully learned operator model* refers to the architecture that tries to learn the true reconstruction $f = \mathcal{A}^* g^\eta$ only from measurement data g^η that includes a dynamic motion. Consequently, the model does not receive any information about the operator $\mathcal{A}$. The model $\mathcal{F}_\theta^{\mathcal{A}}(g^\eta)$ is therefore a fully learned operator model, while $\mathcal{F}_\theta \circ \mathcal{T}(g^\eta)$, where $\mathcal{T}$ can be the FBP or Radon transform, are models that receive some information about the operator through their inputs $\mathcal{T}(g^\eta)$.
Now that the general structure of the network has been laid out, detailed architecture choices can be discussed. In Section 3.2 several relevant concepts of neural networks have been considered, most importantly convolutional neural networks. This type of neural network is best suited for grid-like data such as image or video data. Therefore, the models studied are fully convolutional.

4.1.1 Architecture

This section will give a detailed description of the network structure, starting with the chosen design of the *residual block* described in 3.2.3. This will provide the foundation for the transformation of the input data as well as the design of the SESOP-block in Figure 4.5.

First, we will introduce notation for different combinations of layers. The most common combination is a CNN layer, followed by batch normalization (BN) and activation. Batch normalization has been discussed in Section 3.3.4 as a form of regularization. In the following, this sequence will be called *CBA*. When no batch normalization is added, the combination is called *CA*.

A residual block is composed of several residual layers, each of which executes a series of CBA layers and adds their resulting output to that of a shortcut connection.

Figure 4.6 illustrates this block with a depth of two, indicating that two residual layers are arranged in sequence. Residual blocks are incorporated at multiple stages within the network: initially, within the data block $\mathcal{N}^{\mathrm{d}}_{\theta^0}$, and subsequently as the network blocks for the search directions $\mathcal{N}^{\mathrm{sd}}_{\theta^i_n}$, $i = 1, 2$. The depths of these residual block is part of the numerical analysis in Section 6.1.1. Residual blocks were chosen at these points in the network to easily adjust the depth of different parts of the network depending on their complexity. The authors of [51] suggest that deep networks perform better when using residual layers, as was further explained in Section 3.2.3. Due to the rapidly increasing depth of the network when several iterations are unrolled, residual layers were chosen to avoid degrading the training error [51] (see Section 3.2.3).

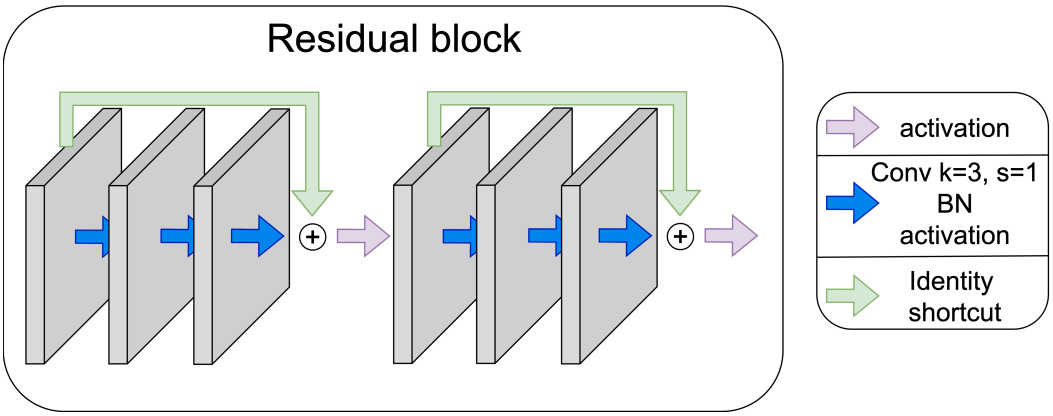


Fig. 4.6.: A residual layer consists of a sequence of three CBA layers added to the output of a shortcut layer, followed by an activation. This configuration is applied iteratively to expand the depth of the residual block. A residual block with depth of two is illustrated here. The convolutional layers maintain the data's dimensionality by using a kernel size of 3×3, strides of 1×1 and padding of 1×1. Batch normalization is performed before the activation.

The first input to the network consists of an initial estimate of the reconstructed image, labeled f_0. This input does not need many transformations because it is already in the correct image space X. Two CA layers are applied outside of the SESOP-block, as seen in Figure 4.7 to convert the image to its required dimensions. Aside from this initial transformation of f_0 outside of the SESOP-block, every subsequent f_n is converted only within this block. Thus, as the output shape is pre-defined, every input f_n and f_{n-1} to the block must have the same dimensions as the output.

Before examining the other input to the network, that is, the measurement data g^η, let us first consider the SESOP-block in more detail. In Figure 4.8 the complete iteration block is shown, including reshaping its input and output data.

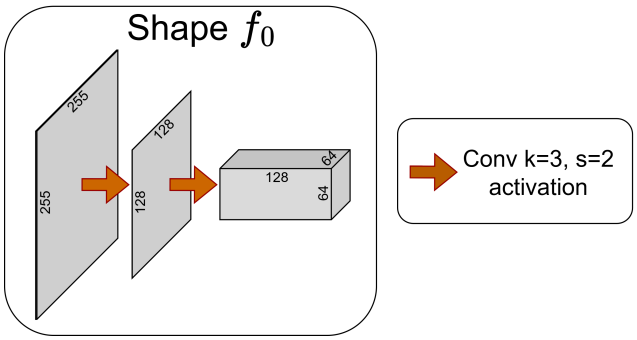


Fig. 4.7.: Transforming the initial estimate f_0 through two CA layers to achieve the necessary dimensions for the SESOP-block.

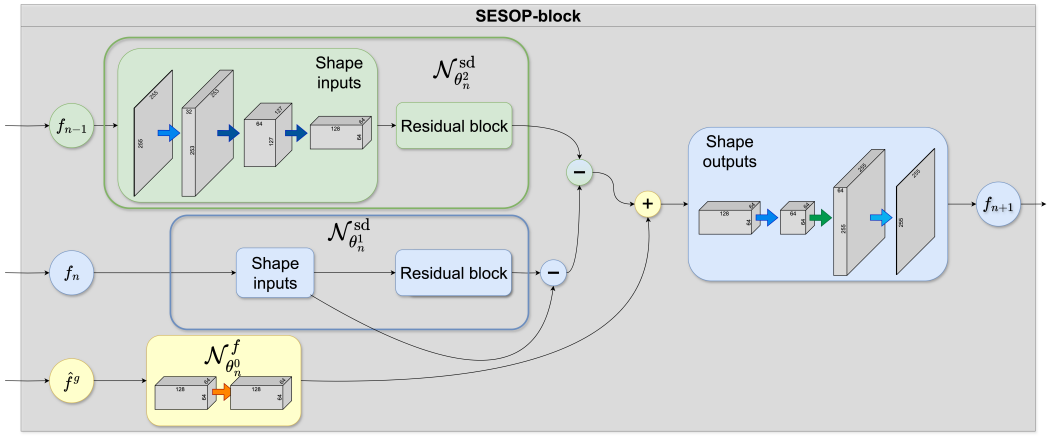


Fig. 4.8.: A more detailed overview of the internal architecture of the SESOP-block. The search direction data is first reshaped and then passed into residual blocks. The output of the data processing block $\mathcal{N}^{\mathrm{d}}_{\theta^0}$ is further transformed in $\mathcal{N}^{f}_{\theta^0_n}$. After performing the operations given by the SESOP iteration the output is transformed into its required shape.

Inspired by the U-Net architecture [98], the data dimensions within the iteration block will be deeper, that is, many channels, while the image dimensions will be smaller. The chosen internal dimensions are $128 \times 64 \times 64$. Thus, the inputs f_n and f_{n-1} must be transformed from their original dimensions of $1 \times 255 \times 255$ to $128 \times 64 \times 64$. Similarly, the output data f_{n+1} need to be transformed into the correct shape. This has been achieved by the CNN layers visualized in Figure 4.9.

In order to reshape the input data size, three CBA layers are used. For the output reshaping, a CBA layer is employed, followed by an upsampling layer and another CBA layer. The final CBA layer incorporates a sigmoid activation function to ensure that the output pixels of the network are all between 0 and 1.

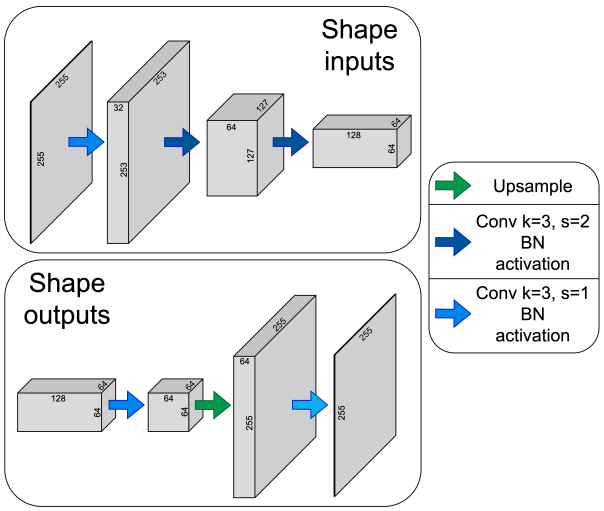


Fig. 4.9.: To reduce the size of the input data and increase their dimensions, three CBA layers are applied. For the reverse process a CBA layer, upsampling layer and CBA layer are applied. The final CBA layer has a sigmoid activation.

After reshaping the input data f_n and f_{n-1}, it is passed into two residual blocks. As seen in Figure 4.6, within the residual block, the data dimensions remain the same. The depth of these blocks is a network parameter and can be adjusted, but is always the same for all search directions and for all unrolled iterations in a network.

Let us now consider how the measurement data are processed in the network block $\mathcal{N}^{\mathrm{d}}_{\theta^0}$, which is illustrated in Figure 4.10. Due to the complex transformation required by this part of the network, several CBA transformations are performed.

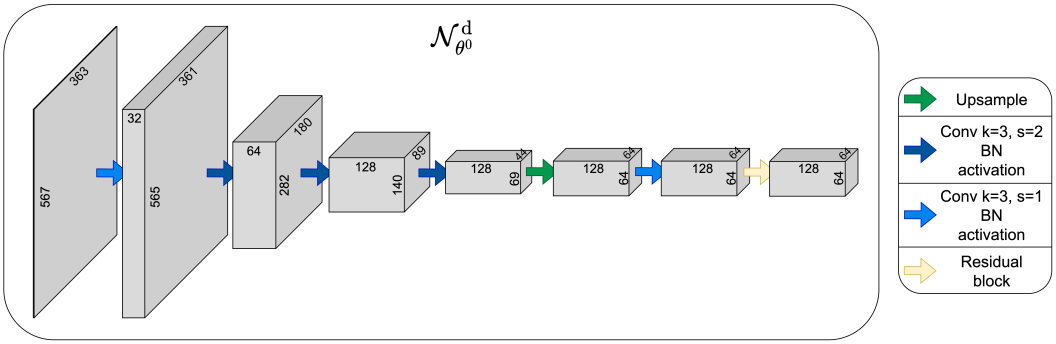


Fig. 4.10.: The measurement data undergoes several convolutional transformations. Blue arrows represent different CBA layer configurations. The green arrow symbolizes an upsampling step. The yellow arrow references a residual block.

Input data are always grayscale images, which are represented as data with one channel on the very left. The first step of transformation increases the number of channels, followed by several transformations which further increase channel depth whilst also decreasing image size. This is done with CBA layers using 3×3 kernels

and 2×2 strides, visualized as dark blue arrows. The internal dimensions of the data in the SESOP-block are defined as $128 \times 64 \times 64$. To reach this size, an upsampling layer is applied, which performs a linear interpolation between neighboring pixels, shown as a green arrow. Upsampling does not have learnable weights; therefore, it is usually followed by a convolutional layer to learn the optimal upsampling step. Finally, a residual block is applied. Its depth is a network parameter that will be analyzed in the Evaluation section 6. This main transformation of the input g^η is performed outside the iteration block. Within each iteration, only the iteration dependent factor needs to be learned. The layers representing $\mathcal{N}^f_{\theta^0_n}$ are shown in Figure 4.11. Here, a CA layer is used, which keeps the data dimensions unchanged. Its purpose is to further transform the input $\hat{f}^g$ and to learn the factor, which is dependent on the iteration step; see (4.7).

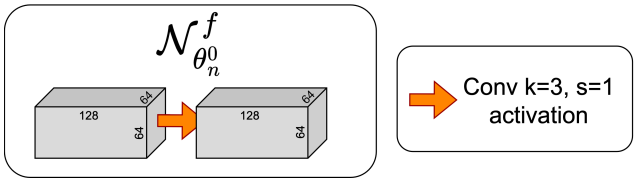


Fig. 4.11.: To adjust for the iteration dependent factor in the g^η term, a CA layer is applied within the iteration block.

4.2 Radon Network

In this section, a different architecture is presented following the concept of Adler et al. [2] where it was suggested to support the unrolled network by applying the discrete Radon transform in each iteration. For further details on the method suggested in this paper, refer to Section 3.5.
The model architecture introduced here unrolls the SESOP iteration differently to the previous approach. Instead of fully unfolding the SESOP equation as in (4.2), the original formula is used (see (4.1)).

Because our application is CT, the inexact operator in the term $(\mathcal{A}^\eta)^*(\mathcal{A}^\eta f_i - g^\eta)$ can be replaced by the discrete Radon transform $\mathcal{R}$ for each search direction index

$i \in I_n$. In the case of the adjoint operator, the transposed Radon transform is used. Applying these steps to (4.1), we arrive at

$$\begin{aligned} f_{n+1} &= f_n - \sum_{i \in I_n} t_{n,i} (\mathcal{A}^\eta)^* (\mathcal{A}^\eta f_i - g^\eta) \\ &= f_n - \sum_{i \in I_n} t_{n,i}\, \mathcal{R}^* (\mathcal{R} f_i - g^\eta) \end{aligned} \tag{4.8}$$

and then approximate each summand with a network block. The sum then reads

$$\sum_{i \in I_n} t_{n,i}\, \mathcal{R}^* (\mathcal{R} f_i - g^\eta) \approx \sum_{i \in I_n} \mathcal{N}_{\theta_n^i} (\mathcal{R}^* (\mathcal{R} f_i - g^\eta)). \tag{4.9}$$

As with the previous network, the Radon network is trained as a whole. The iteration blocks $\mathcal{N}_{\theta_n^i}$ are part of the Radon network, labeled $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta, f_0)$, or simpler $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$.

As for the FLO architecture from the previous section, the goal of the network is to learn the reconstruction according to the true operator $\mathcal{A}_\Gamma$. Thus, the network does not simply learn the values $t_{n,i}$ but tries to infer the correct reconstruction of $t_{n,i}\mathcal{A}^*(\mathcal{A} f_i - g^\eta)$ using the inexact operator $\mathcal{R}^*(\mathcal{R} f_i - g^\eta)$.

The overall structure of the unrolled method remains as in the FLO architecture from the previous section, see Figures 4.2 and 4.3. However, the construction of individual iteration blocks differs. Focusing on the search directions, the term $\mathcal{R}^*(\mathcal{R} f_i - g^\eta)$ can be directly calculated for each f_i. The result of this term and the current search direction f_i are subsequently passed to a CNN block. The foundational network proposed in [2] recommends the inclusion of a *memory* component, labeled s_k in Algorithm 16. This memory is represented by data carried over from previous iterations to serve as additional input to the current iteration, similar to the skip connections of the U-Net architecture. However, SESOP utilizes prior iteration output as search directions, which already serves as a form of memory. Therefore, this architecture does not implement memory in the sense of [2]. .

For example, if $I_n = \{n, n-1\}$ the network blocks then approximate the search directions and data term as follows:

$$\begin{aligned} \mathcal{N}_{\theta_n^i} (\mathcal{R}^* (\mathcal{R} f_n - g^\eta)) &\approx \omega_n^i \mathcal{A}^* (\mathcal{A} f_n - g^\eta) \\ \omega_n^i &\approx t_{n,i}, \end{aligned}$$

where ω_n^i for $i = n, n-1$ are learned by the network to approximate the $t_{n,i}$ factors of the SESOP equation.
Finally, the full iteration can be calculated by summing the outputs of the network

blocks $\mathcal{N}_{\theta_n^i}(\mathcal{R}^*(\mathcal{R}f_n - g^\eta))$ and subtracting them from the previous iteration results. The complete structure of this network can be found in Figure 4.12.

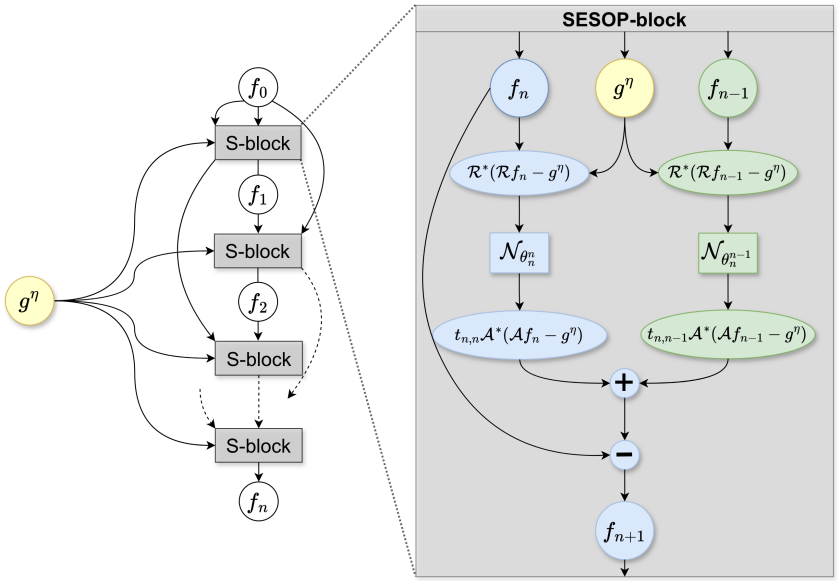


Fig. 4.12.: The structure of the SESOP-block includes one CNN block per search direction. The input of each of these block consists of the output of a previous iterations and the measurement data g^η.

This network relies heavily on the Radon transform and adjoint Radon instead of learning the true operator $\mathcal{A}_\Gamma$ as does the FLO architecture. This is reflected in training time and computational requirements; see Chapter 6.
In Equation (4.1), the true operator is used to address the dynamic nature of the problem. However, since $\mathcal{A}_\Gamma$ is unknown, this architecture uses the static Radon transform in its place. Consequently, the network must learn to compensate for the static nature of the operator as well as the factors $t_{n,i}$.

4.2.1 Architecture

Compared to the previous design, the architecture of this network is less complex. As shown in Figure 4.12, the only points at which the architecture uses neural network layers are for each search direction in each iteration block. First, the term $\mathcal{R}^*(\mathcal{R}f_i - g^\eta)$ is calculated using the Python packages *odl* [66] and *DIVal* [62]. This computation has to be wrapped into a neural network layer in order to perform the backpropagation of the network weights during training. Fortunately, researchers at the University of Bremen released this functionality as part of the *DIVal* Python package. The output of these computations is then fed into the network blocks,

labeled $\mathcal{N}_{\theta_n^i}$. The structure of these blocks is shown in Figure 4.13. Following the architecture in [2], the image created by $\mathcal{R}^*(\mathcal{R}f_i - g^\eta)$ is concatenated with f_i, before feeding it into the network block corresponding to search direction i. This step is shown in Algorithm 16 line 3.

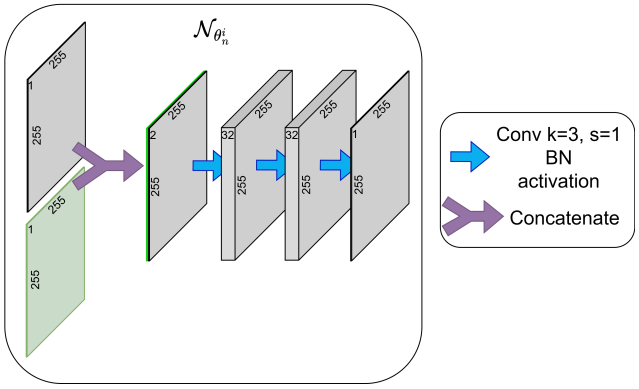


Fig. 4.13.: This block uses the Radon transform of reconstruction f_i, shown as a green image, and f_i itself, shown in gray, to feed into several CBA layers. This block has a depth of 2, as 2 intermediate layers with 32 channels are used. The output data always has a channel depth of 1.

The concatenation operation simply layers two images with one channel each into data with two channels. This is visualized with a purple arrow in Figure 4.13. Following that, several CBA layers are added that only adjust the channel dimension. The input image size is maintained via convolutions with kernels of 3×3, strides and padding of 1×1. Thus, the sequence accepts two input channels and produces an output with a single channel. The intermediate data contains 32 channels. The number of intermediate layers is parameterized, in Figure 4.13 two intermediates are shown, this is referred to as depth of two.

To conclude the chapter, we shall now review the differences of the two proposed network designs. The most obvious difference is the complexity of the FLO network. Here, the iteration steps did not receive any support through an explicit forward operator. Instead, only the input data introduce external information. Both network architectures receive measurement data g^η and an initial reconstruction estimate f_0. The Radon network additionally uses the Radon transform $\mathcal{R}$ and its adjoint $\mathcal{R}^*$. Consequently, the FLO network takes more advantage of CNN layers to transform data throughout the network. Within the iteration step, this means that data enter the block with a single channel and the required image data: $1 \times 255 \times 255$ and exit the block with the same shape. Within each iteration block, the data undergoes a transformation to its intermediate shape of $128 \times 64 \times 64$. This design choice was inspired by the U-Net architecture, which also transforms images into a shape with many channels and smaller image size. Outside of the iteration block,

the measurement data undergoes many convolutions before being used in the iteration block. This is necessary as the block $\mathcal{N}_{\theta^0}^{\mathrm{d}}$ must transform the input g^η into a reconstruction $\mathcal{A}^* g^\eta$. The size of this network block is analyzed in the following chapter. Furthermore, the depth and performance are evaluated with respect to replacing the measurement data with an FBP reconstruction or the adjoint Radon transform of the measurement data.

In contrast to this, the Radon network only applied CNN layers within the iteration block. Here, the input and output of the block are not further transformed. The image size remains the same, only the channel depth is adjusted. The measurement data are used within the Radon transform, which is why it does not have to go through the drastic transformation as in the other network. The design has been kept simple to analyze how influential the provided operator is and if the network can learn the true reconstruction even though the provided operator is the inexact version.

5 Methodology

This chapter details the methodology employed for the experimental evaluation of the core hypothesis of this dissertation. The general aim of this research is to assess whether unrolling the SESOP method into a neural network can effectively serve as a reconstruction method for dynamic CT. Specifically, the two neural network architectures presented in Chapter 4 will be rigorously evaluated across a range of parameter settings and diverse datasets.

The chapter begins in Section 5.1 with a detailed presentation of the three datasets utilized. Two of these datasets were custom-built for this research to simulate specific dynamic noise characteristics: one incorporates vibration movements imposed on the objects, while the other simulates linear drifts. All noise in these two datasets is randomly generated and unique to each sample. The third dataset is a collection of medical imaging data, which, while lacking dynamic noise, features quantum noise caused by low-dose CT scans.

Subsequent sections outline the network training design, covering aspects such as the loss function and optimization settings in Section 5.2.1. Further details of the specific network architectures are presented in Section 5.2.1. The performance of the proposed neural network models will be critically compared with two established numeric reconstruction methods, which were discussed in Section 2.4.

The performance measures used to conduct this evaluation are detailed in Section 5.4. This section introduces two key performance metrics that provide a robust foundation for a quantitative comparison of the reconstruction quality achieved by the different methods. Finally, the chapter concludes by summarizing the computational requirements of each method and architecture. This section will also provide details on the server setup used for evaluation and a summary of the required Python packages.

5.1 Datasets

In this dissertation neural network architectures are investigated that can learn a reconstruction f from data g^η, which is perturbed by an unknown deformation

model Γ. In most cases, deep learning models need a large amount of data for training. However, ground-truth data for CT reconstructions are not easily available. The low-dose parallel beam (LoDoPaB) dataset, discussed in Section 5.1.2, is a manually curated dataset of low-dose medical CT images. Another solution is to create phantoms that simulate objects of interest and generate the corresponding sinograms. This also solves the problem of the size of the dataset. Deep machine learning models, in particular, need a very large amount of samples to learn; however, in many cases, it is not feasible to generate this amount of CT data for computational and financial reasons.

In Section 5.1.1 the datasets used to investigate the effectiveness of the architectures from Chapter 4 are described. Two datasets have been generated, one simulates a vibration motion and the other simulates a linear drift of the object. The LoDoPaB dataset does not include any operator inexactness η, but a quantum noise δ. This data is used to evaluate whether the proposed architectures can also be applied to measurement data with a different type of noise.

5.1.1 Simulated dynamic CT data

Two dimensional CT data was simulated to investigate the effects of temporal perturbations on reconstruction methods for nano-CT. This gave flexibility to adjust different properties of the data, such as the shape and density of the objects as well as the temporal noise levels.
The noise has been modeled as movements between detectors and objects and is represented by shifts and rotations of the phantom. This movement was modeled on authentic nano-CT data provided by Dr.-Ing. Jonas Fell, associated with the Fraunhofer Institute for Nondestructive Testing (IZFP).
Data have been generated for two-dimensional CT imaging in both parallel beam and fan beam geometry. The sampling of scan angles and detector points can be found in Table 5.1.

Geometry	$M = N^2$	K	L
Parallel	255^2	567	363

Tab. 5.1.: Geometry settings for parallel geometry. $M = N^2$ denotes the number of reconstruction pixels, K the number of scanning angles or time steps and L the number of detector pixels.

Phantoms The first part of data generation was to create phantoms F with a field of view of 255×255 pixels, $F \in \mathbb{R}^{N \times N}$. Here, the notation was adapted to represent the two-dimensional image space. The density of the object is represented by the gray values of the pixels ranging from 0 to 1, 0 being no density. Both main and sub-shapes consist of randomly selected gray values chosen from the uniform distribution $\mathcal{U}(0.2, 0.8)$. Every pixel outside of the shape has value 0.

Every object is composed of a primary shape and three subsidiary shapes, randomly chosen to be rectangular or elliptical (see 5.1). The dimensions are randomly chosen from a uniform distribution, resulting in a coverage of 4% to 16.3% for rectangles and 3.2% to 12.6% for ellipses. For subshapes, the coverage ranges from 0.26% to 0.96% for rectangles and from 0.198% to 0.78% for ellipses.

The main shape is centered in a 26 pixel radius around the midpoint, and the sub-shapes in a 76 pixel radius. The radius for the sub-shapes is chosen larger so that not all sub-shapes are included in the object; this further increases randomness of the final object.

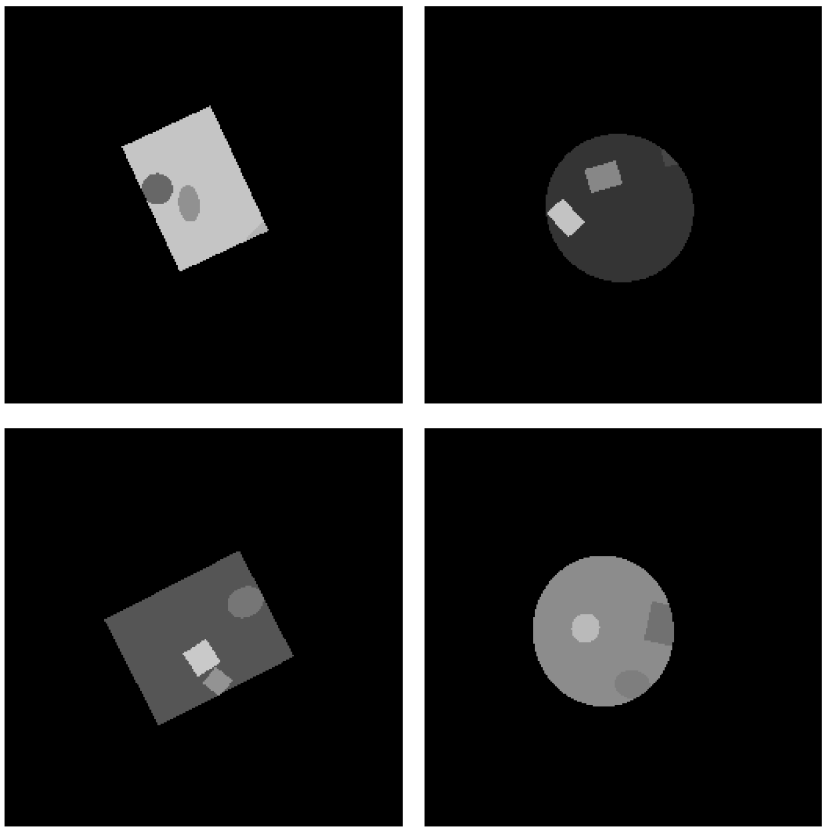


Fig. 5.1.: Four phantom samples: Each phantom consists of a primary shape with several smaller shapes included. The shapes can be either rectangular or elliptical and are generated and arranged randomly.

Noise As discussed in Section 2.3.3, the inverse problem has a dynamic component caused by the positional changes of the object relative to the CT scanner. In the following, the noise is described in the discrete context.
The main focus of this dissertation is to evaluate neural network architectures as reconstruction methods that perform well on data inflicted with dynamic noise; thus, the quantum noise defined by δ in (2.14) is assumed to be zero. We now describe the dynamic inverse problem in its discrete form. For this, the notations of (2.19) and Section 2.1.1 are combined. The disturbances were characterized with the unknown deformation model Γ and are approximated by an inexact operator, which is labeled $\mathcal{A}^\eta$, see (2.19). For an object $F \in \mathbb{R}^M$, instead of simulating $\mathcal{A}^\eta F$, the deformed object $F \circ \Gamma$ was simulated. As explained in Section 2.3.3, we refer to the perturbed measurement data as $g^\eta \in \mathbb{R}^{K \times L}$, that is, the data originating from a dynamic object with an unknown deformation model. These perturbed data were generated by

$$\mathcal{A}_j(F \circ \Gamma) = g_j^\eta, \quad \text{with } j = j(k,l) \text{ for all } k \in \mathcal{K} \text{ and } l \in \mathcal{L},$$

where $F \in \mathbb{R}^M$ and $\mathcal{A}_j = (a_{j,m})_{m=0,\dots,M-1}$ is the discretized Radon transform with

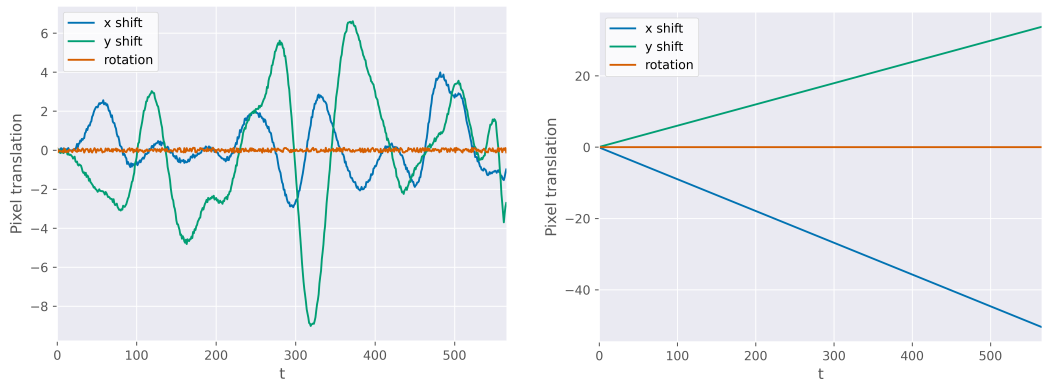
$$\mathcal{A}_{j(k,l)} = (\mathcal{R}_d f)(\omega_k, s_l),$$

as defined in (2.13).

The goal is to reconstruct a fixed state of the scanned object, i.e. the state during the first scan angle $\mathcal{A}_{j(k=0,l)}(F \circ \Gamma)$ from all available data g_j^η. For each phantom F an individual set of motions has been generated such that for each scanning angle φ_k with $0 < k \in \mathcal{K}$ the phantom is shifted with respect to its original location at $k = 0$. At this location, the object is unperturbed and, therefore, in its original state.

This research specifically focused on simulating two types of motions: **vibrations** and **linear shifts**.
To simulate both motions, the following section will introduce noise functions. The domain of these functions is the interval $[0, T]$. For implementation purposes, those functions are evaluated in the discrete set of time instances $t_k \in \mathcal{K}$, where $\mathcal{K}$ is the index set of the set Φ of angles. The time instances are therefore in a 1-to-1 relationship with the angles in Φ. **Vibration** movements are composed of two-dimensional shifts and rotations. The resulting shifts are visualized in 5.2. Two categories of random motion were combined to mimic these vibrations: dampened and overlapping sinusoidal waves combined with a minor random noise.

The sinus noise has been modeled using overlapping sinus waves after examples from the Fraunhofer IZFP. Randomly selected scan angles served as starting points



(a) Vibration shifts **(b)** Linear shifts

Fig. 5.2.: Shifts in x-, y-direction and rotation for vibration and linear shifts.

for 38 overlapping sinus waves. The amplitude, frequency, and damping parameters have been created randomly for each phantom. The shifts in each direction have been limited to a maximum of 15 pixels.
Let x, y be the spatial coordinates of the two-dimensional pixel data, thus $x, y \in \{1, \dots N\}$. The dampened sinus waves have been modeled as functions $[0, T] \to \mathbb{R}$ and are defined for x-y dimensions of the image:

$$\epsilon_x(t) = A_x \sin(2\pi\omega t)e^{-ct},$$
$$\epsilon_y(t) = A_y \sin(2\pi\omega t)e^{-ct},$$

where A_x and A_y are the amplitudes, ω defines the frequency and c the damping parameter. The amplitudes have been constructed by defining an overall amplitude A and distributing this value according to $A = \sqrt{A_x^2 + A_y^2}$, with $A_x = r \cdot A$, where $r \in [0, 1]$ is a scaling factor. The exact settings of the random distributions are listed in Table 5.2.
The second type of noise is a small random noise that moves the object in x, y

Parameter	Distribution
A	$\mathcal{N}(7.874, 161.29)$
r	$\mathcal{U}(0, 1)$
c	$\mathcal{U}(0.01, 0.1)$
ω	$\mathcal{U}(0.0159, 0.0286)$

Tab. 5.2.: Random distribution settings for sinusoidal noise.

directions and rotates it at an angle γ. These random variables were generated from a truncated normal distribution with $\mu = 0$ and $\sigma = \zeta$, truncated at $[-\zeta, \zeta]$, where $\zeta \in \mathcal{N}(0.001, 0.0002)$. Let ξ_x, ξ_y and ξ_γ be functions $[0, T] \to \mathbb{R}$ providing those

random shifts in both directions, x and y, as well as the random rotation with angle γ. The whole distortion can then be formalized as a function

$$d_V\left(t,(x',y')\right) = \Big(OS_1 \cdot \mathrm{R}\Big(\xi_\gamma(t),\xi_x(t),\xi_y(t),\epsilon_x(t),\epsilon_y(t)\Big) \cdot OS_2 \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}\Big)^T,$$

where R is the affine mapping based on the motion functions defined above. The function d_V maps a time instant $t \in [0,T]$ and a location (x',y') in the 2-d object to a new location. The matrix R can be written as

$$\mathrm{R}(\xi_\gamma,\xi_x,\xi_y,\epsilon_x,\epsilon_y) := \begin{pmatrix} \cos(\xi_\gamma) & -\sin(\xi_\gamma) & \epsilon_x \\ \sin(\xi_\gamma) & \cos(\xi_\gamma) & \epsilon_y \\ 0 & 0 & 1 \end{pmatrix}.$$

The translation matrices OS_1 and OS_2, defined as

$$OS_1 := \begin{pmatrix} 1 & 0 & \frac{N}{2} \\ 0 & 1 & \frac{N}{2} \\ 0 & 0 & 1 \end{pmatrix}, \quad OS_2 := \begin{pmatrix} 1 & 0 & -\frac{N}{2} \\ 0 & 1 & -\frac{N}{2} \\ 0 & 0 & 1 \end{pmatrix},$$

shift the rotations such that the image rotates around the origin.

Generating **linear shifts** is considerably more straightforward. Let us define random variables ξ_x and ξ_y from $\mathcal{N}(0,\zeta)$, truncated at $[-\zeta,\zeta]$, where $\zeta := \frac{0.4 \cdot N}{T}$. This means that the maximum distance a point in the phantom can move during the whole time frame $[0,T]$ is 40% of the FOV. Then the linear shift in x and y is defined as functions ϵ_x and $\epsilon_y : [0,T] \to \mathbb{R}$

$$\epsilon_x(t) := t \cdot \xi_x$$
$$\epsilon_y(t) := t \cdot \xi_y.$$

The distortion function d_L can then be defined as

$$d_L\left(t,(x',y')\right) = \begin{pmatrix} 1 & 0 & \epsilon_x(t) \\ 0 & 1 & \epsilon_y(t) \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}.$$

At each time $t \in [0,T]$, a point in the phantom moves in the directions x and y by $\epsilon_x(t)$ and $\epsilon_y(t)$, respectively.
When simulating the dataset, these distortions are applied to the discrete FOV with a pixel space of $N \times N$. For example, $F_{(1,1)}$ is the pixel at coordinates $x = y = 1$. To

calculate the perturbed phantom, either d_V or d_L are applied to all pixels of phantom F. Both distortion functions are dependent on the time instances $t_k \in \mathcal{K}$, which are in a 1-to-1 relationship with angles $\varphi \in \Phi$. Therefore, the perturbed phantom at time instance $t_k \in \mathcal{K}$ is defined as

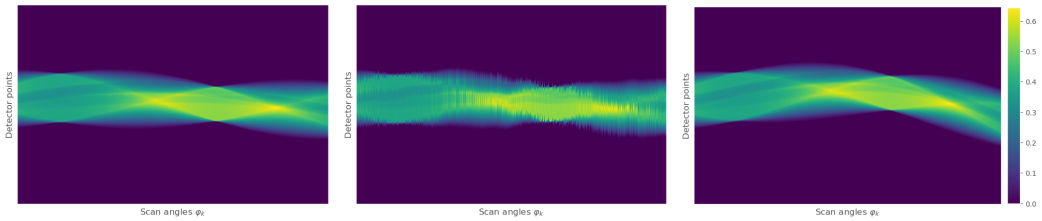
$$F^{\eta,t_k}_{(x,y)} := F_{(d_V(t_k,(x,y)))},$$

for all $(x, y),\ 1 \leq x, y \leq N$.

To calculate the sinograms $g^\eta \in \mathbb{R}^{K\times L}$ the perturbed phantom at each scan angle is required. We therefore consider a series of perturbed phantoms, one for each motion at $t_k \in \mathcal{K} = \{0, \ldots, K-1\}$.

$$F^{\eta}_{(x,y)} = \left(F_{(x,y)}, F^{\eta,t_1}_{(x,y)}, \ldots, F^{\eta,t_{K-1}}_{(x,y)}\right) \in \mathbb{R}^{K\times N\times N}.$$

Sinograms The sinograms were generated using the Operator Discretization Library (ODL) [1] and the ASTRA Toolbox [116] using the parameterization found in Table 5.1. Sinograms have been generated for the original and the perturbed cases; see Figure 5.3.



(a) Without motion. **(b)** With vibration motion. **(c)** With linear motion.

Fig. 5.3.: Sinograms for unperturbed phantom, and perturbed for each simulated motion.

To calculate a perturbed sinogram, the discrete Radon transform (see (2.13)) is applied to each phantom in the sequence of perturbed phantoms $F^\eta_{(x,y)}$ for all x, y. The discrete inexact Radon transform $\mathcal{R}^\eta_d$, can then be written as

$$(\mathcal{R}^\eta_d f)(\omega_k, s_l) := \sum_{x=1}^{N}\sum_{y=1}^{N} F^{\eta,k}_{(x,y)} a_{j,x,y}, \text{ with } j = j(k,l)$$

where $F^{\eta,k}_{(x,y)}$ is a perturbed phantom at time step t_k and ω_k is a scan angle at time step k and s_l is the discrete detector point. Compared to (2.12) $a_{j,x,y}$ has dimensions $J \times N \times N$ instead of $J \times M$. The sinograms are then of dimensions $L \times K$, where

each column corresponds to a different scan angle or time step and is calculated using the corresponding perturbed phantom.

As seen in Figure 5.3 the sample from the vibrations dataset shows clear perturbations that resemble small movements of the object. The same object, perturbed using the linear shift movement, exhibits very different distortions. Here, the object shift caused a smooth wave-like displacement of the sinogram.

Two datasets were generated using these techniques, one with vibration shifts and one with linear shifts. For machine learning, it is required to divide the datasets into training, testing, and validation subsets. The number of samples per subset can be found in Table 5.3.

object motion	Train	Val	Test	Total
Vibrations	30 489	1283	323	32 095
Linear shifts	11 631	369	323	12 323

Tab. 5.3.: Number of samples per data split for vibration and linear shift data.

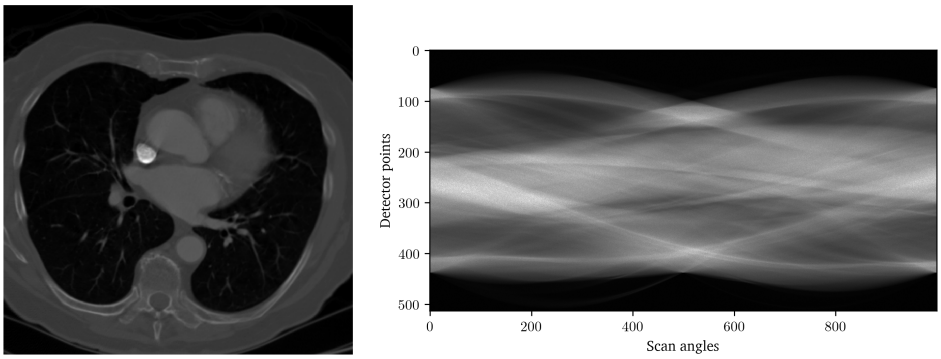
The phantoms in the linear shift data are the same as those in the vibration dataset. Of course, the simulated motion and the resulting sinograms differ. Only some of the training and validation phantoms appear in the linear shift dataset. The data split remains identical: the training phantoms in the linear shift dataset are a subset of the training samples of the vibration dataset, and the same applies for the validation samples. The dataset for the linear shift motion is intentionally smaller due to the reduced randomness in the shifts. The shift's direction and velocity remain constant throughout the motion. We expect that a neural network would need fewer data points to effectively learn this type of motion.

Due to the large impact of this dynamic noise on the quality of reconstructions, no further imaging noise (quantum mottle) has been added. These datasets are intended for foundational research. Future experiments could consider larger image size and different shifts between the scanner and the object, as well as include quantum noise $\delta > 0$.

5.1.2 Low-Dose Parallel Beam (LoDoPaB)

This dataset has been purposefully created by Leuschner et al. [69] to evaluate and compare data-based and classical reconstruction approaches. The foundation of these data is the LIDC/IDRI database: the Lung Image Database Consortium (LIDC) and the Image Database Resource Initiative (IDRI) [4, 5]. This database includes

helical thoracic CT scans of 1010 individuals from several academic centers and medical imaging companies.



(a) LoDoPaB Phantom **(b)** LoDoPaB Sinogram

Fig. 5.4.: Thoracic phantom and its simulated low-dose projection from the LoDoPaB dataset.

Therefore, the data are based on a range of technical parameters and scanner models. The reconstructions went through a rigorous selection process, in which very noisy images or images that did not offer enough valid pixel values were excluded. The final LoDoPab data include scans from 812 patients, separated into training with 632 patients and testing and validation sets with 60 patients each.
As the LIDC / IDR database provides phantom data from many different CT setups, the authors of the LoDoPaB dataset [69] cropped all phantoms to 362 px $\times$ 362 px to include only the inner circle-shaped reconstructions. These cropped phantoms were used to generate synthetic projection data.
These data do not include a model inexactness η as in the previous datasets, but include simulated quantum noise δ, see (2.14). Quantum noise can be explained by the probabilistic nature of photon generation, attenuation, and detection. In low-dose CT, the effects of quantum noise are amplified, and it is therefore crucial to include these in synthetic data. Quantum noise can generally be modeled using a Poisson distribution [25]. Using this distribution, low-dose projection data was simulated with the Operator Discretization Library (ODL) [1].
Sinograms were generated for 1000 equidistant angles between 0 and π and 513 equidistant detector points, see Table 5.4.

Geometry	$M^2 = N$	K	L
Parallel	362^2	1000	513

Tab. 5.4.: Geometry settings for LoDoPaB

The neural network introduced in Chapter 4 will be trained on this dataset to evaluate if the architectures can be effectively trained on low-dose measurement

data instead of dynamic data. The main focus of the evaluation is on different dynamic CT problems; the LoDoPaB data is therefore a good proof-of-concept to broaden the application space. For this reason, the size of the training, test, and validation sets is relatively small for such complex objects. Table 5.5 shows the size of the individual datasets available and the number of samples used in the architecture evaluation in Chapter 6.

	Train	Val	Test	Total
Original dataset	35 820	3 522	3 553	42.895
Used dataset	10 240	3 472	50	13 762

Tab. 5.5.: Number of samples per data split for LoDoPaB data.

5.2 Configuration of Methods

Chapter 3 presented a range of concepts in machine learning and neural networks. Each of these can be configured in numerous ways. Determining the configurations for training a neural network can drastically impact the performance of the model. Furthermore, the architectures presented in Chapter 4 introduce additional configuration possibilities. Sections 5.2.1 and 5.2.1 will detail the configurations used for evaluation in Chapter 6. The iterative methods used for comparison to the network results also require some parameter configurations. These will be detailed in Section 5.2.2.

5.2.1 Network Training Settings

Some of the most crucial neural network configurations are the optimization algorithm, the loss function, and the activation functions. For deep neural networks regularization is another important concept to increase the generalizability of the model. Thus, two regularization techniques are included in the learning process in addition to the batch normalization (see Section 3.3.4), which has been discussed as part of the architecture in Section 4.1.

Loss Function In Section 3.3.3 two functions were introduced: MSE and MAE. As discussed in this section, MSE is a preferable loss function unless the data include a lot of outliers. In the case of outliers, MAE improves the generalization abilities of the network. If the dataset is clean, that is, it does not include outliers, MSE

converges faster than MAE.
For CT data, the range of pixel values is limited. Both sinogram and phantom pixels are limited between 0 and 1. Within the network, the LeakyReLU activation function allows negative values, but the Sigmoid function scales all output values between 0 and 1. Therefore, outliers are unlikely to affect the loss function. For this reason, MSE was chosen as the loss function in all experiments.

Activation Function When different activation functions were tested, it was found that LeakyReLU outperformed ReLU. Models trained with ReLU struggled to learn any part of the objects and mostly returned blank images.
Although the pixel values of the reconstructions should all be between 0 and 1, the network profits from including negative values. However, to ensure that the prediction of the network is in the valid range, either ReLU or Sigmoid activation has to be used. As ReLU simply replaces all negative values with 0, Sigmoid gives a better transformation of the negative values.

Optimization Method The optimization algorithm chosen is the Adam optimizer. Pytorch [88] offers an implementation of Adam called *AdamW* which includes regularization through the L^2 norm. The parameters in Algorithm 3.21 have been set according to the default values of AdamW: the learning rate $\epsilon = 0.001$; the decay rates $\rho_1 = 0.9, \rho_2 = 0.999$; and the constant $\delta = 1\mathrm{e}{-08}$. Another parameter that is important for optimization methods based on stochastic gradient descent with mini-batches (see Section 3.3.2) is the size of the mini-batches. Generally, a larger batch size leads to a more stable optimization; however, due to memory requirements the batch size had to be set to 32.

L^2 Norm Penalties The Pytorch implementation of the Adam optimizer, called *AdamW*, includes a parameter λ that controls the impact of the L^2 norm on the objective function; see 3.33. The parameter λ was set to 0.01.

Early stopping This regularization method was applied only to experiments trained for more than 30 epochs. If the network overfits before reaching 30 epochs, other regularization techniques should be increased, such as norm penalties. When training for more epochs, early stopping patience has been set to 5 and $\Delta = 0.00015$.

Architecture Details

The architectures presented in Sections 4.1 and 4.2 require several settings, such as channel depth and data shape in CNN layers. There are no strict guidelines or recommendations on how to design CNNs. All of the settings presented are based on architectures such as the U-net [98] and trial and error.

Fully learned operator network This architecture is the most complex of the two proposed networks. Many settings are visible in the figures in Section 4.1.1. Generally, all convolutions have been designed with kernels of size 3×3. Furthermore, data enters the network with one channel. This dimension is increased to 32 in the first convolution. Following the construction of the U-net, when increasing channels the depth is doubled, whilst decreasing the image dimensions by half (by applying suitable stride and padding settings). The shape of the internal data, that is, the convolutions within the ResNet blocks, is always $128 \times 64 \times 64$. To achieve this, upsampling layers are applied. When data exists an iteration block, the original image dimension (255×255 in the case of the simulated datasets) must be reconstructed. For this, the image dimensions are upsampled first and then the channel dimensions are reduced.
The network requires initial data f_0. Moslty FBP was used as an initial estimate of the reconstruction. In some experiments a random image has been generated, using a uniform distribution on the interval $[10\mathrm{e}{-}07, 1)$. Future experiments could investigate whether an initial input of zeros is better suited than random values from this distribution.

Radon network This architecture applies the Radon transform and its adjoint throughout the iterations. This operator has been implemented as a neural network layer by the Python package DIVal [62] and Tomosipo [53]. These packages provide an implementation of the forward and backward propagation of the Radon transform and the adjoint Radon transform, which in turn has been implemented in Operator Discretization Library [66].
These packages are used to calculate $\mathcal{R}f_n$ and $\mathcal{R}^*(\mathcal{R}f_n - g^\eta)$ for iteration index n.

The dimensions of the CNN layers are straightforward. For each search direction f_n and $\mathcal{R}^*(\mathcal{R}f_n - g^\eta)$ are concatenated into data with two channels. Then the channels are increased to 32 while the image dimensions are kept consistently at 255×255. After a number of convolutions, the channel depth is reduced to 1 before

further SESOP calculations are performed to calculate f_{n+1}. The number of these convolutions is controlled by the depth parameter analyzed in Chapter 6.

5.2.2 Settings of Iterative Methods

Two iterative reconstruction methods, which were discussed in Section 2.4, will be evaluated on the same datasets as the network architectures. The RESESOP-Kaczmarz method (Section 2.4.3) and the Dremel method (Section 2.4.4) were selected as reconstruction techniques for dynamic datasets due to their specialized nature. The RESESOP-Kaczmarz approach incorporates local modeling errors into the regularization process, but depends on the model inexactness η or its estimation. Conversely, the Dremel method operates without requiring any deformation information, as it estimates and corrects operator inexactness during the iterative regularization.

RESESOP-Kaczmarz method

For this research RESESOP-Kaczmarz has been implemented in Python. The parameters crucial for this algorithm are the noise estimations η for the inexact operator (2.20) and the quantum noise in the measured data δ (2.14). As the dynamic noise of the data has been generated (see 5.1.1), we are operating in the non-realistic position that η is fully known. Therefore, in terms of operator inexactness, the results of RESESOP-Kaczmarz are as optimal as possible. Instead of using $\eta_{k,l}$ for all $k \in \mathcal{K}$ and $l \in \mathcal{L}$, we calculate the maximum norm $\|(\eta_{k,l})_{l\in\mathcal{L}}\|_\infty$ for all $k \in \mathcal{K}$. Thus, we write

$$(\eta_k)_{k\in\mathcal{K}} = (\|\eta_{k,l}\|_\infty^{l\in\mathcal{L}})_{k\in\mathcal{K}} = (\|(\eta_{0,l})_{l\in\mathcal{L}}\|_\infty, \dots, \|(\eta_{K-1,l})_{l\in\mathcal{L}}\|_\infty)\,,$$

where

$$\|(\eta_{k,l})_{l\in\mathcal{L}}\|_\infty = \max\{|\eta_{k,0}|, |\eta_{k,1}|, \dots |\eta_{k,L-1}|\}.$$

The inexactness used in the algorithm is therefore the maximum operator inexactness per scan angle. As can be seen in Pseudocode 2, the inexactness is used to regularize the method; once by limiting the size of the stripes $\mathcal{H}$ (see 2.39) and once through the discrepancy principle 2.38. When using the norm $\|\eta\|_\infty^{l\in\mathcal{L}}$ instead of $\eta_{k,l}$, the regularization is slightly relaxed.
In [77] the algorithm has also been evaluated using up- and down-scale variations of $\|\eta\|_\infty^{l\in\mathcal{L}}$ by 20%, however, for this analysis, we focus on the true η.

Parameters $\tau > 1$ and δ also influence the discrepancy principle. The purpose of τ is to guarantee that the error, that is, the value of $\|\mathcal{A}^\eta f - g^\eta\|$, can fall below the noise level $\eta\rho + \delta$. For these experiments, τ is set to 1.00001. The value of δ has been set to 0.001 even though our measurement data g^η do not include any quantum noise. However, as previously stated, slightly increasing the noise level is in our interest. Furthermore, for the experiments analyzed in Chapter 6, all RESESOP-Kaczmarz executions have been terminated after a fixed number of iterations, before the discrepancy principle was satisfied.

The ASTRA toolbox [116] was employed to implement the operators $\mathcal{A}$ and $\mathcal{A}^*$, representing the Radon transform and its adjoint, respectively, in our work with CT data.

The discrepancy principle outlined in 2.38 has been utilized as a stopping criterion along with establishing a maximum limit of N iterations for the termination of the algorithm. In the evaluation of the experimental setup described in 6, the value of N is fixed at 20.

Dremel method

The implementation of this method has been provided by researchers from the University of Bremen[1] with whom we collaborated on the publication of [77]. In [77] the Dremel method was evaluated on the vibration data detailed in 5.1.1. Consequently, the configuration of parameters, along with both pre-processing and post-processing procedures for the algorithm, were identified by assessing the performance of the method on the validation set of these data. The up-sampling factor, also referred to as the super-resolution factor, has been set to $r_{\text{SR}} = 2$ and thus $n_{\text{SR}} = 2 \cdot L = 726$. In article [35] the authors propose the use of an edge filter as a preprocessing step for the analysis of low-contrast signals; however, this was not found to be beneficial for our data. Instead, it was found that mean subtraction followed by zero padding performed better for function pre, while in function post the correlation signal is cropped to the original size of y and y^δ. Additionally, a weighting is applied that amplifies the correlation values for higher shifts, as they are biased to be smaller due to zero padding in function pre.

The learning rate, represented by ω, and the relaxation parameter, denoted as λ, are also established using validation data. In this analysis, we have assigned $\omega = 1$ and $\lambda = 1$, indicating that no relaxation is applied. In contrast to what is suggested in [35], it was found that the method performed the best when the operator correction

[1] contact Dr. Johannes Leuschner jleuschn@uni-bremen.de

is applied from the very first iteration onward. The authors of [35] suggest an initial warm-up without this correction step. Instead of an explicit discrepancy principle, the algorithm terminates after a fixed number of iterations. Evaluation of the vibration data prompted setting the maximum number of iterations N to 32.

FBP

The implementation of FBP has been provided by ASTRA toolbox [116]. For the vibration and linear shift data, the default Ram-Lak filter was applied. In the case of LoDoPaB data, [69] suggests the Hann filter with a frequency scaling of 0.641. Frequency scaling is based on the principle of conservation of mass in the Radon transform, which states that the sum of all projections of an object, that is, $\int_{L(\omega(\varphi),s)} f(x)dx$, should be proportional to the total mass (or absorption) of the object itself. In [69], it was not explained further how this frequency scaling was applied. When comparing the total mass of the validation sinograms with the attenuation of the respective objects, we received a scaling factor of 5.54 which has been used as a multiplication factor for the FBP reconstructions.

5.3 System Specifications and Software Environment

All experiments have been executed on the servers of the national high-performance cluster (Nationaler Hochleistungsrechnen – NHR).
The neural networks have been trained on a single *Nvidia A100* GPU with up to 80GB of VRAM or a single *Nvidia H100* GPU with up to 94GB of VRAM. Only the largest network settings required over 80GB of memory.

RESESOP-Kaczmarz method was executed on 10 *Intel Sapphire Rapids Xeon Platinum 8468* cores and a total of 40 GB memory for 10 samples in parallel at a time. This method took $6 - 12$ days for 20 iterations on a single sample.
The Dremel method required a *Nvidia A100* GPU with 80 GB. The method was run for 5 samples at a time which took less than 5 minutes.
The FBP was run on 4 *Intel Sapphire Rapids Xeon Platinum 8468* CPUs with 40 GB of memory. The 323 test samples were computed in one minute.

The implementation of the data generation part can be found in `https://gitlab.gwdg.de/alice.oberacker/generate-data-4-nanoct` and the network architecture and other reconstruction methods are included in `https://gitlab.gwdg.de/`

`alice.oberacker/ct_reconstructions`. The required Python packages were installed in an apptainer container. The versions of the core packages are as follows:

- Torch [88]: 2.7.0+cu126
- DIVal [62]: 0.6.2
- ODL [66]: 0.8.1
- Astra Toolbox [116]: 2.0.0
- Tomosipo [53]: 0.6.0

5.4 Performance Measures

Finally, to evaluate how well the reconstruction methods perform, it is necessary to find suitable measures. In Section 3.3.3 the mean squared error and the mean absolute error were introduced in the context of loss functions. Either of these can be used as performance metrics; however, there are other measures that are more appropriate in the context of image reconstruction, that is, the *peak signal-to-noise ratio* and the *structural similarity index measure*.

Peak signal-to-noise ratio

The *peak signal-to-noise ratio (PSNR)* is an extension of the MSE, see (3.31). It is defined in decibels (dB) by

$$\begin{aligned} \mathrm{PSNR}(y, \hat{y}) &:= 10 \cdot \log_{10}\left(\frac{m^2}{\mathrm{MSE}(y, \hat{y})}\right) \\ &= 20 \cdot \log_{10}\left(\frac{m}{\sqrt{\mathrm{MSE}(y, \hat{y})}}\right) \\ &= 20 \cdot \log_{10}(m) - 10 \cdot \log_{10}(\mathrm{MSE}(y, \hat{y})). \end{aligned} \tag{5.1}$$

We assume that the pixel values are in $[0, m]$, where m is the maximum representable density. In CT data, the maximum intensity is $m = 1$.

The benefits include retaining the simplicity of MSE on a familiar dB scale, it is differentiable for MSE > 0, it is smooth in the sense of loss functions (see 3.3.3) unless MSE is very small and it is monotenous. Normalization through PSNR scales the unprocessed MSE by the square of the maximum achievable pixel intensity.

Consequently, PSNR values are directly comparable across images or modalities possessing distinct dynamic ranges, in contrast to raw MSE values, which are influenced by the absolute scale of the data. This is useful as we can compare the performance of the architecture across different datasets. The decibel scale (dB) improves interpretability as applying $10\log_{10}(\cdot)$ in PSNR reduces the extensive variations in MSE to a more practical range. On this logarithmic scale, minor improvements at high-quality levels become more noticeable, reflecting our usual perception of incremental quality improvements.
PSNR is a widely adopted metric that is a benchmark in imaging studies. Researchers are familiar with interpreting values and their results in different applications. In contrast, raw MSE values do not offer such an intuitive understanding. Notably, PSNR shares certain limitations with MSE, such as its relatively low correlation with perceived quality and its insensitivity to structural aspects. It penalizes small and large errors quadratically, regardless of their location or perceptual relevance. For instance, a minor error in a flat background is penalized similarly to one on a significant edge. For further analysis on PSNR refer to [123].

Structural similarity index measure

The *structural similarity index measure (SSIM)* evaluates local luminance, contrast, and structural similarity over sliding windows, resulting in a score in the interval $[-1, 1]$. An SSIM of 0 indicates that there is no similarity between inputs, 1 is perfect correlation, and -1 signifies perfect anti-correlation. SSIM is defined on the image patches $x = \{x_i\}_{i=1}^{M}$, the ground truth, and $y = \{y_i\}_{i=1}^{M}$, the reconstructed image; it is defined as

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{5.2}$$

where the *pixel sample means* are given as

$$\mu_x = \frac{1}{M}\sum_{i=1}^{M} x_i, \qquad \mu_y = \frac{1}{M}\sum_{i=1}^{M} y_i.$$

The *sample variances* are defined as

$$\sigma_x^2 = \frac{1}{M-1}\sum_{i=1}^{M}(x_i - \mu_x)^2, \qquad \sigma_y^2 = \frac{1}{M-1}\sum_{i=1}^{M}(y_i - \mu_y)^2$$

and the sample covariance is defined as

$$\sigma_{xy}^2 = \frac{1}{M-1}\sum_{i=1}^{M}(x_i - \mu_x)(y_i - \mu_y).$$

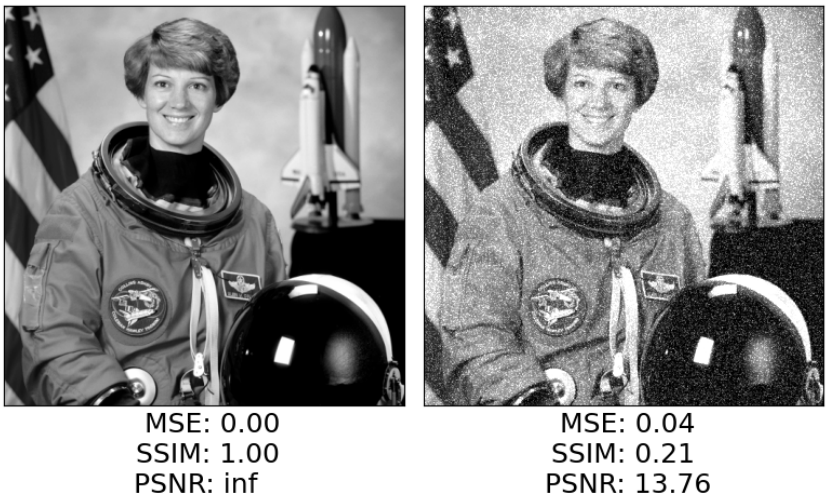
C_1 and C_2 stabilize the division when $\mu_x^2 + \mu_y^2$ or $\sigma_x^2 + \sigma_y^2$ are close to zero and are defined as

$$C_1 = (k_1 \cdot m)^2, \quad C_2 = (k_2 \cdot m)^2,$$

where m is the data range. Common defaults are $k_1 = 0.01$, $k_2 = 0.03$.

SSIM has benefits such as aligning well with human perception, being sensitive to luminance, contrast, and structure. These properties make it more sensitive to edge preservation and textures [122]. However, SSIM is only piecewise differentiable and non-convex, properties that make it is problematic to employ as a loss function. SSIM can have many plateaus and local minima on its loss surface, which can lead to unstable gradients during training. Depending on initialization, some parameters may not converge well due to backpropagation that leads to small gradients [26]. However, after training the network, SSIM continues to be an excellent evaluation metric. As a quality measure, it has a significantly better correlation with perceived structural and edge fidelity compared to MSE or PSNR.

Figure 5.5 highlights the weaknesses of MSE and PSNR. The first image shows the noise-free image. In Figure 5.5b, 5.5c, 5.5d the MSE loss value is 0.04 and the PSNR measure varies between $13.76 - 14.37$. However, a human would rate the quality of these images very differently, and the SSIM value reflects this human perception better.
Both PSNR and SSIM are used as performance measures in Chapter 6.



(a) Original image

(b) Image with Gauss noise

(c) Image with added constant

(d) Image with high amount of blurring

Fig. 5.5.: This set of images demonstrates how the performance of MSE, PSNR and SSIM differ. (a) shows the original image. The MSE remains consistent across all noisy images, the PSNR varies slightly, but the SSIM shows significant differences among the noisy images. Especially image (c) illustrates that SSIM reflects human perception closer than MSE or PSNR.

6 Evaluation

This chapter focuses on the core research of this dissertation. The performance of both network designs, introduced in 4, is systematically assessed in various settings. Initially, each design undergoes rigorous testing across different parameter settings to determine their optimal configurations. Once the best-performing setup for each network is identified, we will compare their results with other established reconstruction techniques. This comparative analysis provides insights into the relative strengths and weaknesses of the proposed designs in the context of dynamic CT data.
Finally, the methods will also be evaluated in terms of their computational requirements, that is, compute costs and time.

6.1 Performance

As seen in Chapter 4, there are several architecture parameters necessary for the different networks. First, the parameters required by the original iterative method, such as the number of search directions and the number of unrolled iterations; second, the parameters that regulate the depth of each iteration. Instead of evaluating all combinations, which would demand a huge amount of compute power, different parameter combinations were tested for the two types of parameters separately.

The first set of experiments evaluates how deep each of the iteration blocks should be. For this, two parameters are required: the depth of the search direction blocks and the depth of the measurement data block. We will refer to these as *depth sd* and *depth* g^η, respectively. The second set of experiments focuses on the parameters originating from the SESOP method. The two parameters are the number of unrolled iterations, labeled N, and the number of search directions, labeled *# sd*.
It is important to acknowledge that we do not expect to identify the universally optimal unrolled model in this evaluation. As illustrated in Section 3.2 and Chapter 4, the configuration of a neural network is determined by a multitude of decisions. For example, in a standard neural network, performance is significantly influenced by factors such as the number of neurons per layer, the total layers, the chosen

activation function, and the loss function. In the context of a CNN architecture, it is crucial to define the shape of the filters in each layer, including the kernel sizes and channel numbers. Regarding the particular architectures developed in Chapter 4, the data dimensions during the operations within the iteration blocks are fixed. Any of these choices might affect the performance of the evaluated networks.
Furthermore, in Section 3.1 some principles of statistical learning were discussed which build the foundation of machine learning. Training a neural network is a probabilistic process due to the stochastic gradient descent and random initialization of weights and biases, and therefore it is not expected that training the same parameter configurations multiple times produces identical outcomes.

To evaluate the proposed architectures, the performance will be tested on all three datasets. The dataset that simulates the vibration motion of the object during the scan is the largest with the most random motion. On this dataset, the largest number of parameter settings are tested. The best choice of parameters will then be evaluated on the linear shift dataset, where some modifications will be necessary. Considering that the LoDoPaB data (5.1.2) present a low-dose inverse problem, not a dynamic inverse problem, we will evaluate how well the best parameter settings work on these data. The models are individually trained on the respective datasets and evaluated using PSNR and SSIM as measures in the validation set.

Finally, the best models will be evaluated on the test data and compared to the performance of other reconstruction methods. Iterative reconstruction methods that can solve dynamic CT problems have been presented in Section 2.4. The methods used for comparison here are the RESESOP-Kaczmarz (see 2.4.3) and the Dremel method (see 2.4.4). The RESESOP-Kaczmarz method can solve linear inverse problems with inexact forward operator and noisy measurement data. A possible application for this method is dynamic nano-CT. However, the method requires an estimate of the operator inexactness η for each distinct sample. Therefore, a direct comparison with other methods is difficult, as the results of RESESOP-Kaczmarz are based on additional information. However, the Dremel method has been developed specifically for dynamic CT with rigid body movements and does not require a priori information.
The FBP (see 2.4.5), being one of the most fundamental reconstruction techniques, was also included in this analysis.

Considering that the RESESOP-Kaczmarz and Dremel methods are powerful methods for solving dynamic CT reconstructions, applying them to a static problem, such as the LoDoPaB data, is not deemed productive. This data is not perturbed by dynamic noise and thus the noise is not dependent on the scan angles and therefore does not

change between individual subproblems. Applying RESESOP-Kaczmarz to such a problem would be disproportionate. The Dremel method corrects the movement shifts of the object within the iterations; however, there are no shifts to be corrected in the LoDoPaB data. The high quantum noise might distort the cross-correlation results in the Dremel algorithm and lead to wrong reconstructions.
Instead, we refer to a quantitative study that evaluated several network architectures on the LoDoPaB data. The focus here is on how effectively the network architectures introduced in 4 reconstruct objects when the measurement data are subjected to noise caused by low-dose measurements.

6.1.1 Network experiments with Vibration Data

The most detailed analysis of architecture parameters focuses on the vibration dataset, because it is the largest and presents the most complex dynamic noise η among our simulated data sets. The initial analysis performed on the fully learned operator network (4.1) assesses the depth of the search directions and the g^η block. This analysis is performed for all three variations $\mathcal{F}_\theta^A$, $\mathcal{F}_\theta(\mathrm{FBP}(g^\eta))$, and $\mathcal{F}_\theta(\mathcal{R}^T(g^\eta))$. Building on these results, the SESOP parameters are evaluated, that are the number of unrolled iterations and the number of search directions. The Radon network architecture (4.2) is evaluated similarly. The evaluated parameters are the number of unrolled iterations, the number of search directions and the depth of the search directions. Due to the architecture of this model, training requires more resources than the FLO architecture.

Fully Learned Operator Network

The general iteration step of this architecture is defined in (4.3) as

$$f_{n+1} = f_n - \mathcal{N}^{\mathrm{sd}}_{\theta_n^1}(f_n) - \mathcal{N}^{\mathrm{sd}}_{\theta_n^2}(f_{n-1}) + \mathcal{N}^{\mathrm{d}}_{\theta_n^0}(g^\eta).$$

The evaluation of this architecture spans the three g^η block configurations that can be found in Section 4.1:

- the fully learned network: $\mathcal{F}_\theta^A(g^\eta)$
- the post-processing network supported by the reconstruction of g^η using FBP: $\mathcal{F}_\theta(\mathrm{FBP}(g^\eta))$

- the post-processing network supported by the adjoint Radon transform: $\mathcal{F}_\theta(\mathcal{R}^T(g^\eta))$.

The underlying assumption is that learning the exact adjoint operator $\mathcal{A}_\Gamma$ presents the greatest challenge, requiring a deeper network than the alternative versions. Since the FBP or adjoint Radon transforms of g^η already map the data to the target space X, the main question here is: Does the network benefit more from learning via the perturbed sinogram or the artifact-affected phantom? FBP and $\mathcal{R}^T$ introduce distinct visual artifacts, making both formats intriguing for exploration.

Exploring Depth of Iteration Blocks The first set of experiments explores the depth of the iteration blocks, which includes the depth of each search direction and the depth of the block that processes the measurement data g^η. To evaluate these settings, the other parameters need to be set to initial values. The number of unrolled iterations was set to $N = 3$ and the number of search directions was set to $\#\text{sd} = 2$. All experiments are trained for 30 epochs after which the models already showed good results. Once the final parameters have been chosen, the model will be trained for a larger number of epochs.
Tables 6.1, 6.2, and 6.3 present the results of different parameter settings for the three options. The search directions and the data block are constructed from residual blocks, as discussed in Section 4.1.1 and Figure 4.6.
The settings of interest for the depth parameter are all combinations of depth $g^\eta = \{1, 2, 3, 4\}$ and depth $\text{sd} = \{1, 2, 3, 4\}$. For all these combinations, we first evaluate the performance of the fully learned network without using $\text{FBP}(g^\eta)$ or $\mathcal{R}^T(g^\eta)$.

After running experiments for 3 iterations, 2 search directions, g^η depth of 3 and search direction depths of 1 to 4 it was clear that the model overfits. In Figure 6.1, the left side shows the MSE loss for all epochs. The plot shows a good loss improvement; however, for the validation data the loss increases for most settings after a few epochs. This is a clear sign of overfitting which can be avoided by using L^2 regularization with parameter $\lambda = 0.01$.

Using L^2 regularization reduced the overfitting for the experiments but did not solve this problem completely. Some settings still showed signs of overfitting, but instead of optimizing the regularization parameter individually for all other parameters, we will evaluate how large the overfitting tendencies are and use this insight to further influence the choice of parameter settings. Generally, the larger a network is, the more prone it is to overfitting. The first set of experiments evaluates the depth of the iteration blocks and thus directly evaluates the relationship between network

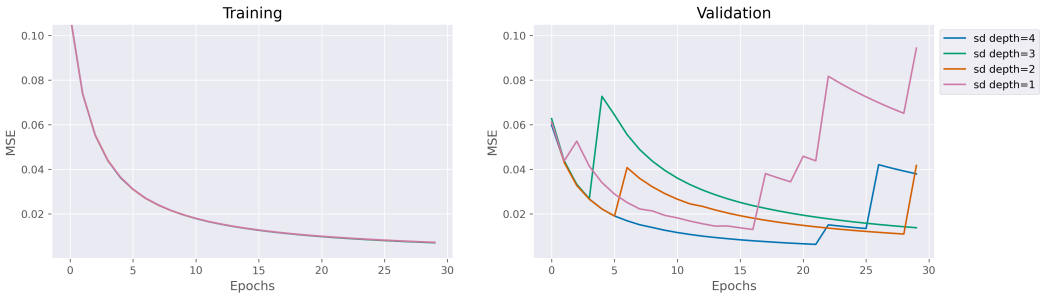


Fig. 6.1.: Experiment for 3 iterations, 2 search directions, g^η depth of 3

depth and overfitting risks. For this, the relative difference between the training loss and the validation loss in the last epoch is calculated by

$$\text{loss}_{\text{diff}} := \frac{|\text{train loss} - \text{val loss}|}{\text{train loss}}.$$

A very high value indicates that the model does not generalize well to the validation data and might therefore be overfitting. In the following experiments, we accept a value less than 0.5. For the example in Figure 6.1, the experiment with a search direction depth of 1 has the highest relative difference of 12.02 in the last epoch and the experiment with a search direction depth of 3 has the lowest relative difference of 0.95.

N = 3, # sd = 2					
depth g^η	depth sd	training	PSNR [db]	SSIM	rel. loss diff.
1	1	2h 38min	32.36 ± 2.73	0.886 ± 0.008	0.3386
	2	3h 22min	31.93 ± 2.60	0.836 ± 0.005	0.2128
	3	3h 35min	17.41 ± 3.00	0.894 ± 0.028	0.5328
	4	4h 10min	$\mathbf{32.63 \pm 2.79}$	0.812 ± 0.006	0.3188
2	1	2h 38min	31.84 ± 2.54	$\mathbf{0.921 \pm 0.010}$	0.0018
	2	2h 57min	32.43 ± 2.62	0.842 ± 0.007	0.3151
	3	3h 13min	31.73 ± 2.47	0.813 ± 0.008	1.8827
	4	3h 33min	31.27 ± 2.60	0.836 ± 0.007	0.2847
3	1	2h 42min	19.29 ± 0.50	0.148 ± 0.036	4.7365
	2	2h 59min	31.60 ± 2.53	0.857 ± 0.006	4.0593
	3	3h 15min	30.91 ± 2.49	0.858 ± 0.008	0.3160
	4	3h 36min	31.40 ± 2.54	0.806 ± 0.008	1.4375
4	1	2h 45min	31.64 ± 2.36	0.675 ± 0.048	0.3207
	2	3h 3min	32.56 ± 2.77	0.821 ± 0.007	0.3165
	3	3h 18min	$\mathbf{32.65 \pm 2.75}$	0.834 ± 0.006	3.1780
	4	3h 37min	31.54 ± 2.07	0.622 ± 0.010	2.1048

Tab. 6.1.: Fully learned operator $\mathcal{N}^{\mathcal{A}}_{\theta_n}(g^\eta)$ evaluated on validation dataset for measurements obtained from perturbed sinograms.

Considering Table 6.1, the first results with L^2 regularization and including the relative loss difference can be evaluated using the performance measures PSNR and SSIM.

Depending on which measure we consider, different experiments performed best. For PSRN we reach a score of 32.65 dB for depth g^η of 4 and depth sd of 3 and for SSIM the best score is 0.921 for depth g^η of 2 and depth sd of 1. Notice that the parameter setting with the best PSNR score also shows some signs of overfitting, since the relative loss difference is 3.18. Alternatively, we can consider the second best result of 32.63 dB for depth g^η of 1 and depth sd of 4 which has a relative loss difference of 0.32. The best result with a low relative loss difference is highlighted in bold and underlined in Table 6.1.

The difference in PSNR results among the best experiments is relatively small. However, the best SSIM result is the only one above 0.9. Furthermore, in Section 5.4, it was discussed that SSIM has a larger correlation with perceived image quality. In addition, the relative loss difference for this experiment is 0.0018, indicating that the model did not overfit.

The relative loss in Table 6.1 indicates that larger models, that is, depth g^η of 3 or 4, have a higher risk of overfitting. The g^η and search direction blocks have been constructed using residual blocks (see 3.2.3). However, the increase in the relative difference between the training loss and the validation loss indicates that a deep block with several residual layers is still prone to overfitting.

> Summarizing all these insights, the best model from this set of experiments is the model with depth g^η of 2 and depth sd of 1.
> These results confirm the intuition that the size of the network and therefore the depth parameters influence the overfitting risks of the architecture. Increasing the number of unrolled iterations and the number of search directions would greatly increase the size of the network, and thus increase the risk of overfitting. Furthermore, a smaller network size reduces the requirements on the compute units used for training. Larger networks require specialized GPUs due to the large memory requirements and increase training times. Considering that experiments with deeper g^η blocks showed clear signs of overfitting.

The experiments for $\mathcal{F}_\theta(\mathrm{FBP}(g^\eta))$ and $\mathcal{F}_\theta(\mathcal{R}^T(g^\eta))$ were performed for a depth g^η of 1 to 3. In Tables 6.2 and 6.3 the results of those experiments are shown. Let us first consider the FBP supported experiments. According to the relative loss difference, this variant of the architecture supported by FBP has a higher risk of overfitting. There is no clear connection between the depth of the iteration blocks and the risk

N = 3, # sd = 2					
depth g^η	depth sd	training	PSNR [db]	SSIM	rel. loss diff.
1	1	1h 33min	30.78 ± 2.27	$\mathbf{0.893 \pm 0.008}$	11.08
	2	1h 57min	32.41 ± 2.64	0.842 ± 0.007	0.34
	3	2h 21min	$\mathbf{32.86 \pm 2.80}$	0.824 ± 0.008	4.17
	4	2h 45min	31.32 ± 2.22	0.835 ± 0.005	4.06
2	1	1h 37min	32.23 ± 2.67	0.813 ± 0.006	0.23
	2	2h 1min	32.64 ± 2.71	0.840 ± 0.006	0.23
	3	2h 25min	32.76 ± 2.77	0.802 ± 0.009	16.96
	4	2h 49min	31.94 ± 2.44	$\underline{\mathbf{0.852 \pm 0.005}}$	0.34
3	1	1h 35min	25.75 ± 1.80	$\mathbf{0.952 \pm 0.012}$	4.82
	2	2h 0min	27.74 ± 1.51	0.582 ± 0.010	19.79
	3	2h 55min	$\underline{\mathbf{32.81 \pm 2.79}}$	0.821 ± 0.006	0.25
	4	3h 20min	22.05 ± 0.41	0.115 ± 0.026	3.25

Tab. 6.2.: Network provided with FBP of measurements, $\mathcal{F}_\theta(\mathrm{FBP}(g^\eta))$ evaluated on test dataset for measurements obtained from perturbed sinograms.

of overfitting. However, the best results have been achieved by experiments that are overfitting.
The best results according to PSRN were achieved using a depth g^η of 1 and a depth sd of 3. However, this experiment has a relative loss difference of 4.17. The second best result has been reached for a depth g^η of 3 and a depth sd of 3, where the relative loss difference is 0.25. For the SSIM we have the same problem. The best two results have a relative loss difference of 4.82 and 11.08. The third best model has an SSIM of 0.852 with a relative loss difference of 0.34 for a depth g^η of 2 and a depth sd of 4.

The experiments for the network supported by $\mathcal{R}^T$ also have a higher relative loss difference with no discernible connection to the depth of iteration blocks.

However, the best parameter settings for this variant show low indications of overfitting. For PSNR the best setting was a depth g^η of 2 and a depth sd of 3 with a PSNR of 32.71 and a relative loss difference of 0.19. For SSIM the best parameter setting is a depth g^η of 3 and a depth sd of 1 with a score of 0.866 and a relative loss difference of 0.47.
The training time for the FBP variation is faster than for the first model, which seems intuitive, as the first model has to reconstruct g^η without any support. However, the training time for the $\mathcal{R}^T$ model is longer. This is because in every forward propagation, the adjoint Radon operator is applied to the input sample. The training

N = 3, # sd = 2					
depth g^η	depth sd	training	PSNR [db]	SSIM	rel. loss diff.
1	1	2h 50min	32.01 ± 2.88	0.818 ± 0.017	0.31
	2	3h 16min	32.43 ± 2.85	0.843 ± 0.009	1.95
	3	3h 44min	31.34 ± 2.54	0.779 ± 0.023	10.28
	4	4h 15min	32.51 ± 2.77	0.818 ± 0.007	4.39
2	1	2h 52min	32.31 ± 2.71	0.787 ± 0.007	1.65
	2	3h 18min	25.72 ± 1.53	0.689 ± 0.071	1.82
	3	3h 47min	$\mathbf{32.71 \pm 2.73}$	0.795 ± 0.013	0.19
	4	4h 20min	30.98 ± 2.34	0.829 ± 0.019	2.88
3	1	2h 51min	31.75 ± 2.53	$\mathbf{0.866 \pm 0.009}$	0.47
	2	3h 20min	31.77 ± 2.45	0.842 ± 0.010	0.31
	3	3h 49min	31.57 ± 2.37	0.852 ± 0.006	3.98
	4	4h 22min	31.58 ± 2.61	0.809 ± 0.008	1.78

Tab. 6.3.: Network provided with discrete adjoint Radon transform of measurements, $\mathcal{F}_\theta(\mathcal{R}^T(g^\eta))$ evaluated on test dataset for measurements obtained from perturbed sinograms.

time could be greatly improved by preprocessing the dataset instead of performing this operation during training.

It appears that providing the network with reconstruction, either by FBP or $\mathcal{R}^T$, leads to a higher risk of overfitting. Further tests could include a larger regularization parameter.
The best results considering the SSIM have been achieved by the $\mathcal{N}^{\mathcal{A}}_{\theta_n}(g^\eta)$ model. For PSNR the model using FBP has slightly better results with 32.81 instead of 32.63 for the first model. However, as stated previously, a smaller model is preferable, and SSIM better reflects human perception. Therefore, the overall best model is considered the FLO model $\mathcal{F}^{\mathcal{A}}_{\theta}(g^\eta)$ with a depth g^η of 2 and a depth sd of 1.

Exploring SESOP Parameters Another set of important network parameters are the number of search directions and the number of iterations that are unrolled in the network. For the previous experiments, the number of search directions was set to 2 and the number of iterations to 3. Those best parameters are used to explore whether a network with more search directions or more iterations can achieve better results. The number of iterations was set to be at least one larger than the number of search directions. The results of previous iterations are used as search directions; thus if the number of search directions is the same as the number of iterations, the

last iteration would still require f_0 as an input instead of the output of an iteration. An example of this has been visualized in Figure 6.2. Here, four iteration blocks are shown with three search directions. The search directions are highlighted in different colors: f_0 in black, f_1 in red, f_2 in blue, and f_3 in magenta. The fourth iteration block is the first to not receive f_0, but only internally generated reconstructions as input.

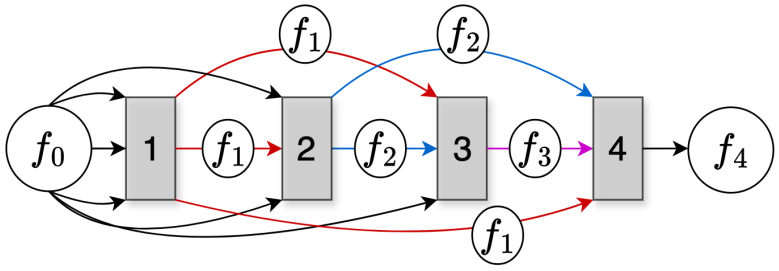


Fig. 6.2.: An unrolled iterative network with 4 iterations and 3 search directions. In order to focus on the search directions, the measurement data has been omitted. The forth iteration block is the first block that does not receive f_0 as an input.

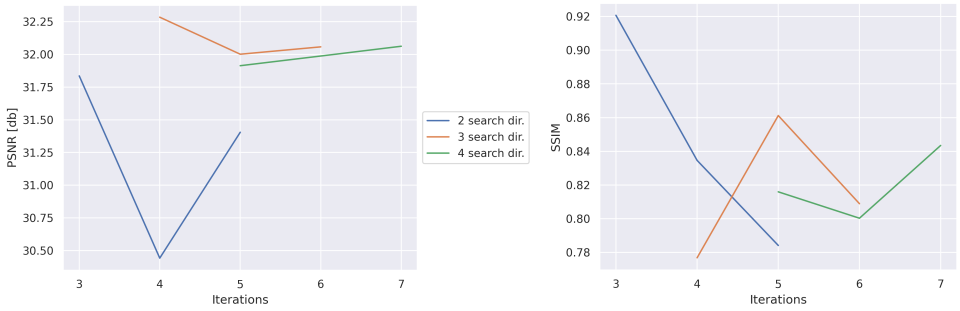
Table 6.4 shows the parameter settings that were evaluated.

depth sd $= 1$, depth $g^\eta = 2$					
# sd	N	training	PSNR [db]	SSIM	rel. loss diff.
2	3	2h 38min	31.84 ± 2.54	$\mathbf{0.921 \pm 0.010}$	0.0018
	4	3h 1min	30.44 ± 2.15	0.835 ± 0.007	0.2039
	5	3h 28min	31.40 ± 2.34	0.784 ± 0.011	0.1962
3	4	3h 25min	$\mathbf{32.28 \pm 2.65}$	0.777 ± 0.014	4.1257
	5	3h 56min	32.00 ± 2.62	0.861 ± 0.006	0.2752
	6	4h 30min	$\underline{\mathbf{32.06 \pm 2.71}}$	0.809 ± 0.007	0.2198
4	5	4h 20min	31.91 ± 2.74	0.816 ± 0.007	0.3327
	6	4h 53min	31.99 ± 2.54	0.800 ± 0.009	15.2512
	7	5h 29min	$\mathbf{32.06 \pm 2.74}$	0.843 ± 0.006	7.9521

Tab. 6.4.: Comparing PSNR and SSIM for number of iterations and number of search directions including training time.

The first setting is the best of the previous experiments. No other experiment setting was able to reach an equivalent SSIM. The best PSNR results were 32.28 for 3 search directions and 4 iterations and 32.06 for 4 search directions and 7 iterations. However, both models showed signs of overfitting, with the larger of the two models having a relative loss difference of 7.95. The best PSRN result with a low relative loss difference used 3 search directions and 6 unrolled iterations and achieved a PSNR of 32.06.

In Figure 6.3, the two plots show how the PSNR and SSIM values vary with the number of iterations, with each line color representing a different number of search directions. The PSNR values for 4 search directions show that with increasing



(a) PSNR **(b)** SSIM

Fig. 6.3.: Comparing PSNR and SSIM for number of iterations and number of search directions.

number of iterations the metric improves as well. However, for 3 search directions the model with the least amount of unrolled iterations performs best. The models with 2 search directions also show inconsistent results in relation to the number of iterations. The plot with SSIM values also shows that the size of the model does not indicate the performance.

> It is important to precisely calibrate both the number of iterations and the number of search directions with respect to each other.

Applying the same selection process as with the previous set of experiments, the best model is still the one with 2 search directions and 3 unrolled iterations. The SSIM value is by far the best, even if the PSNR value of this model is only at 31.84. SSIM better reflects human perception. Additionally, this model shows the least amount of overfitting, which indicates that a model with these settings has more capacity for improvement when training for more epochs.

The combination of parameters of the best performing model was trained again for up to 100 epochs. Early stopping caused the training process to stop after 69 epochs. As discussed above, training the same network multiple times can lead to different results. Unfortunately, the network trained for 69 epochs could not improve the performance of the best model. This model reached a PSNR on the validation data of 32.40 ± 2.73 dB and an SSIM of 0.90 ± 0.0060. According to the PSNR measure, the model improved by 0.56 dB, but the SSIM measure decreased by 0.021. The relative difference at the end of the last epoch between the training loss and the

f_0	training	PSNR [db]	SSIM	rel. loss diff.
FBP	2h 38min	$\mathbf{31.84 \pm 2.54}$	$\mathbf{0.921 \pm 0.010}$	0.0018
random	2h 18min	26.74 ± 2.27	0.789 ± 0.016	0.0553

Tab. 6.5.: Comparison of fully learned operator network $\mathcal{F}_\theta^A(g^\eta)$ with 3 unrolled iterations, 2 search directions, search direction depth of 1 and g^η depth of 2 with FBP initialization and with a random initialization. Evaluated on the validation dataset for measurements obtained from perturbed sinograms.

data	PSNR [db]	SSIM
val	31.84 ± 2.54	0.921 ± 0.010
test	31.61 ± 2.54	0.920 ± 0.010

Tab. 6.6.: Quantitative evaluation of fully learned operator network $\mathcal{F}_\theta^A(g^\eta)$ with 3 unrolled iterations, 2 search directions, search direction depth of 1 and g^η depth of 2. The model was evaluated on validation and test data for measurements obtained from perturbed sinograms.

validation loss was 0.2557, which is sufficiently small. Similarly to the iterative reconstruction methods discussed in 2.4, the network architecture use FBP as an initial f_0. To demonstrate the impact of this initial reconstruction on the network the best parameter settings were also tested with a random f_0 initialization. The results can be found in Table 6.5.

Both measures show that the model with FBP initialization can learn the reconstruction better. However, we can also see that the model still performs well without this initialization, instead using only uniformly distributed values between 0 and 1 and the noise measurement data g^η.

Finally, in Table 6.6 the best model is evaluated on the vibration data reserved for testing and compared to the validation data.

The PSNR and SSIM measures perform similarly well on both data splits, showing that the model can generalize well to previously unseen data.

In Section 6.1.1 the performance of this model will be compared to other reconstruction methods.

Radon Network

The iteration of the Radon network $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$, see (4.8), is written as

$$f_{n+1} = f_n - \sum_{i \in I_n} \mathcal{N}_{\theta_n^i}(\mathcal{R}^T(\mathcal{R}f_i - g^\eta)).$$

The Radon network has been evaluated for different settings for the number of unrolled iterations, the number of search directions, and the depth of the search directions since there is no separate g^η block. The FLO architecture has been evaluated for different initialization values to evaluate if the artifacts introduced by the FBP reconstructions are detrimental to the model or useful. The Radon model applies the Radon transform in each iteration and thus reintroduces these artifacts within the network. For this reason, the Radon model has only been evaluated with FBP reconstructions as initializations. As can be seen in Table 6.7, the training time for this architecture is much longer than for the previous architecture.

sd depth	# sd	N	training	PSNR [db]	SSIM	rel. loss diff.
1	2	3	9h 50min	30.41 ± 2.77	0.866 ± 0.038	0.0150
		4	12h 38min	30.53 ± 2.91	0.869 ± 0.037	0.1019
	3	4	18h 6min	30.57 ± 2.85	0.913 ± 0.033	0.1221
		5	21h 27min	30.13 ± 2.79	0.864 ± 0.025	0.0474
2	2	3	10h 21min	30.92 ± 2.85	0.949 ± 0.015	0.0758
		4	13h 9min	31.33 ± 2.77	0.955 ± 0.013	0.0085
	3	4	17h 52min	$\mathbf{31.35 \pm 2.69}$	$\underline{\mathbf{0.959 \pm 0.011}}$	0.0158
		5	22h 15min	30.93 ± 2.43	$\mathbf{0.961 \pm 0.011}$	0.1838

Tab. 6.7.: Quantitative evaluation of Radon network $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$ evaluated on validation dataset for measurements obtained from perturbed sinograms with search direction depth of 1 and 2.

Throughout the training process, the term $\mathcal{R}^T(\mathcal{R}f_i - g^\eta)$ must be evaluated for every sample, corresponding to each search direction i and each unrolled iteration, and at each epoch. The chosen implementation was provided by the DIVal [62] package, which is further explained in 5.3. Although this network was trained for only 30 epochs, the training of the different parameter settings took between 9 hours 50 minutes and 22 hours 15 minutes. This is a dramatic increase compared to the fully learned operator network, where the longest training time out of all parameter settings was 5 hours 29 minutes. Depending on the use case, this increase in training time might be justifiable.

However, the performance of the Radon network seems to surpass those of the previously discussed models. All experiments with a search direction depth of 2 have an SSIM value of more than 0.9. The best model according to the SSIM measure is the largest of the selected parameter settings with an SSIM of 0.961. The next best model according to the SSIM is the model with 4 iterations, 3 search directions, each with a depth of 2. This model also has the best PSNR value with 31.35 dB. Moreover, this model has a lower relative loss difference and took 4 hours and 32 minutes less to train. We were unable to improve these results by increasing the number of epochs.

data	PSNR [db]	SSIM
val	31.35 ± 2.69	0.959 ± 0.011
test	31.00 ± 2.61	0.958 ± 0.011

Tab. 6.8.: Quantitative evaluation of Radon network $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$ with 4 unrolled iterations, 3 search directions and a search direction depth of 1. The model was evaluated on validation and test dataset for measurements obtained from perturbed sinograms.

The results in Table 6.8 show that the model can generalize well to unknown data, as PSNR and SSIM on the test data are similar to the validation results. The PSNR and SSIM of the fully learned operator model and the Radon model are quite close together, with a 0.61 dB improvement in PSNR for $\mathcal{F}_\theta^{\mathcal{A}}(g^\eta)$ and a 0.038 improvement in SSIM for $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$. However, how does this difference in measures affect the reconstructions visually? In the next section, we compare the reconstructions of some samples visually.

Comparison of Models

We now examine the reconstruction samples in more detail to gain insight into the type of noise and artifacts that degrade the model predictions. In Figure 6.4 the ground truth of four samples is shown together with their reconstructions using the FLO model $\mathcal{F}_\theta^{\mathcal{A}}(g^\eta)$ and the Radon model $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$.

A first look at the reconstructions shows that both models successfully reconstructed the objects and that none of the reconstructions have streak artifacts outside of the objects. When comparing the individual reconstructions between the two models, we can see that the FLO model did a better job at reproducing the edges of the main object as well as the internal shapes. However, elliptical shapes are better reconstructed than rectangular shapes, as some of the corners of the internal shapes

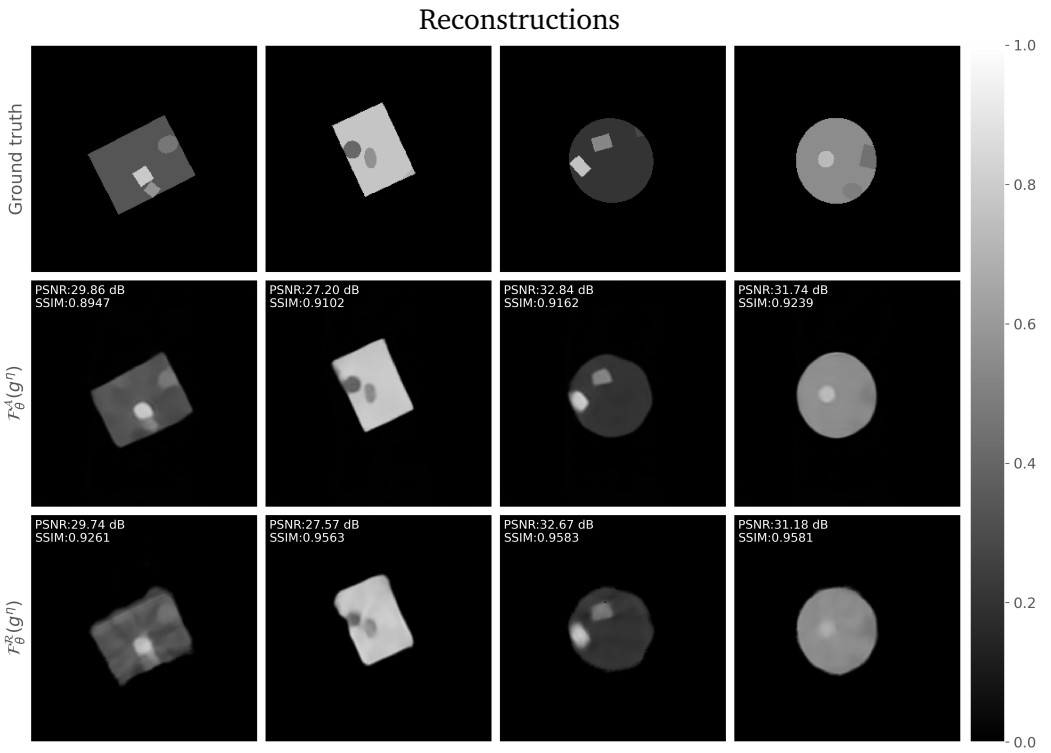


Fig. 6.4.: Four reconstructions of measurement data perturbed with vibrations for the fully learned operator model $\mathcal{F}_\theta^{\mathcal{A}}(g^\eta)$ and the Radon model $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$.

are not as sharp as in the ground truth. This is especially visible in the leftmost sample. The difference in density between shapes in the rightmost sample is smaller than in the other samples. Both models struggled to reproduce these shapes. The Radon model did not reconstruct the edges well. The corners of the rectangular shapes have not been well reproduced and the elliptical shapes show a wave-like boundary. The third sample has a wave-like edge in both reconstructions; however, the FLO model produced a slightly better edge. The first reconstruction of the Radon model has many artifacts inside the main object, which creates the appearance of multiple shapes included in this object. We can understand where these artifacts come from, considering the FBP reconstructions in Figure 6.5. The artifacts within this object are much more apparent than in the other reconstructions, causing problems for the Radon model. Additionally, by repeatedly applying the Radon transform within this network, the model cannot compensate for the artifacts.

Overall, the reconstructions of the FLO model appear less blurry and are closer to the ground truth.

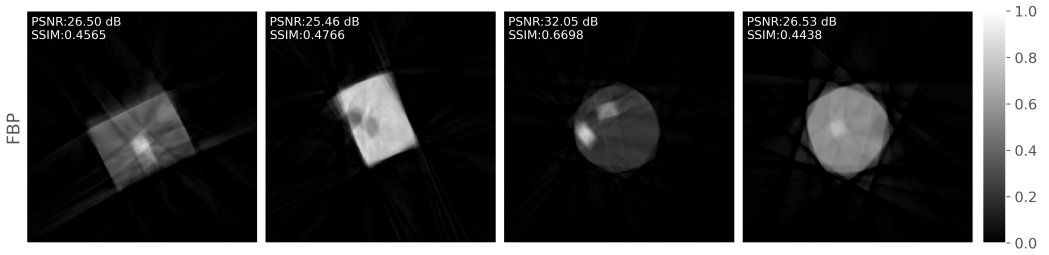


Fig. 6.5.: The FBP reconstructions of measurement data perturbed with vibrations.

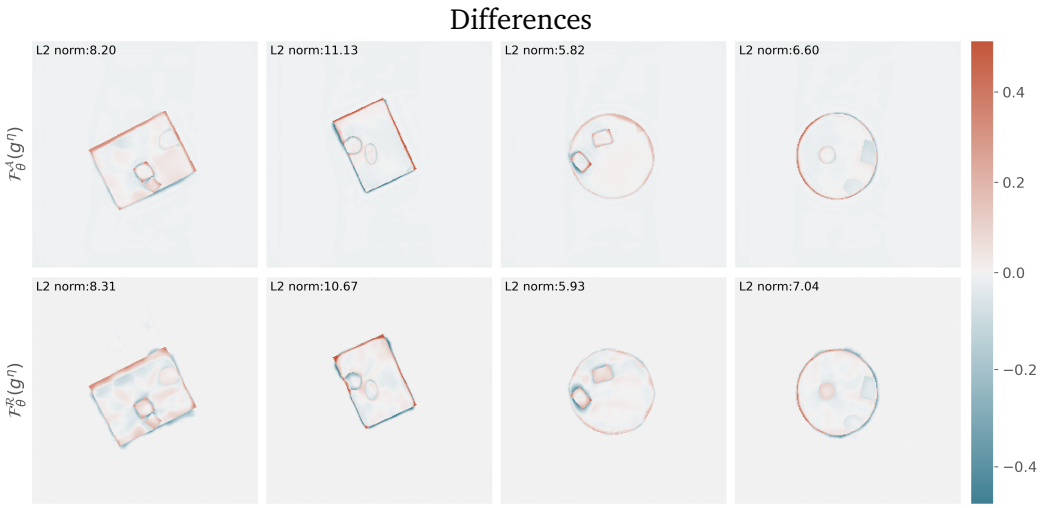


Fig. 6.6.: Difference of reconstructions with ground truth of measurement data perturbed with vibrations by the fully learned operator model $\mathcal{F}_\theta^{\mathcal{A}}(g^\eta)$ and the Radon model $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$.

To get a clearer impression of the noise and artifacts included in the reconstructions, Figure 6.6 displays the differences between the ground truths and the predictions. The regions with overestimated densities appear in red, and those with underestimated densities are depicted in blue. We see that the objects reconstructed by the FLO model have blue edges on one side and red edges on the other, indicating that the object is incorrectly positioned. Most of the Radon model reconstructions have edges that include both blue and red indicating that the object is not shifted but is distorted. The Radon model has therefore more edge pixels correctly reconstructed, but the edges appear worse, as they are not straight or curved as in the ground truth. Furthermore, this explains the PSNR and SSIM measures. In all reconstructions, the Radon model achieved higher SSIM values. In all but one reconstruction, the PSNR is higher for the FLO model. Previously, in Section 5.4, it was discussed that SSIM reflects human perception better than PSNR. However, these results contradict this hypothesis.

Comparison with Other Reconstruction Methods

We now include the non-learned reconstruction methods into our analysis. In Table 6.9 the quantitative results of all reconstruction methods are listed. The best reconstruction results according to PSRN and SSIM are the fully learned operator network and the Radon network, respectively. However, we established previously that the FLO model and the Radon model reconstructed the edges of the objects differently. The FLO model produced straighter edges, though slightly shifted, whereas the edges in the Radon model reconstructions were less straight but had more correctly aligned edge pixels. To gain further insight into how these results compare with the other reconstruction methods, we review the results in Figure 6.7.

	PSNR [dB]	**SSIM**
FBP	27.93 ± 2.82	0.511 ± 0.104
Dremel	30.99 ± 3.01	0.829 ± 0.076
RESESOP-Kacz. with η	30.67 ± 2.74	0.865 ± 0.102
Fully learned op. net	$\mathbf{31.61 \pm 2.54}$	0.920 ± 0.010
Radon net	31.00 ± 2.61	$\mathbf{0.958 \pm 0.011}$

Tab. 6.9.: Quantitative evaluation on vibration test datasets for measurements obtained from perturbed sinograms. The best results for both measures are highlighted.

Between the three non-learned methods, the Dremel method created reconstructions with the least amount of streak artifacts. Both FBP and RESESOP-Kaczmarz have a considerable amount of streaks inside and outside of the object, whereas the reconstructions of the Dremel method have fewer artifacts which are mostly limited to the outside of the object. These artifacts are more visible in Figure 6.8. The blurred edges of the reconstructions are also more visible in this figure. Here we notice that the edges of the first two samples have not been reconstructed as well by the Dremel method as they have been by the FLO model. Overall, the reconstructions by the FLO model produce the best results combining well-defined edges and accurate densities whilst not creating any streaking artifacts.

6.1.2 Network experiments with Linear Shift Data

In this section, the architectures are trained on the data with linear drifts. Having found good parameter combinations for the dataset with vibrating objects in the previous experiments, these settings will be applied to the architecture trained on the linear dataset. This can provide insights on how transferable the network architecture is for different datasets.

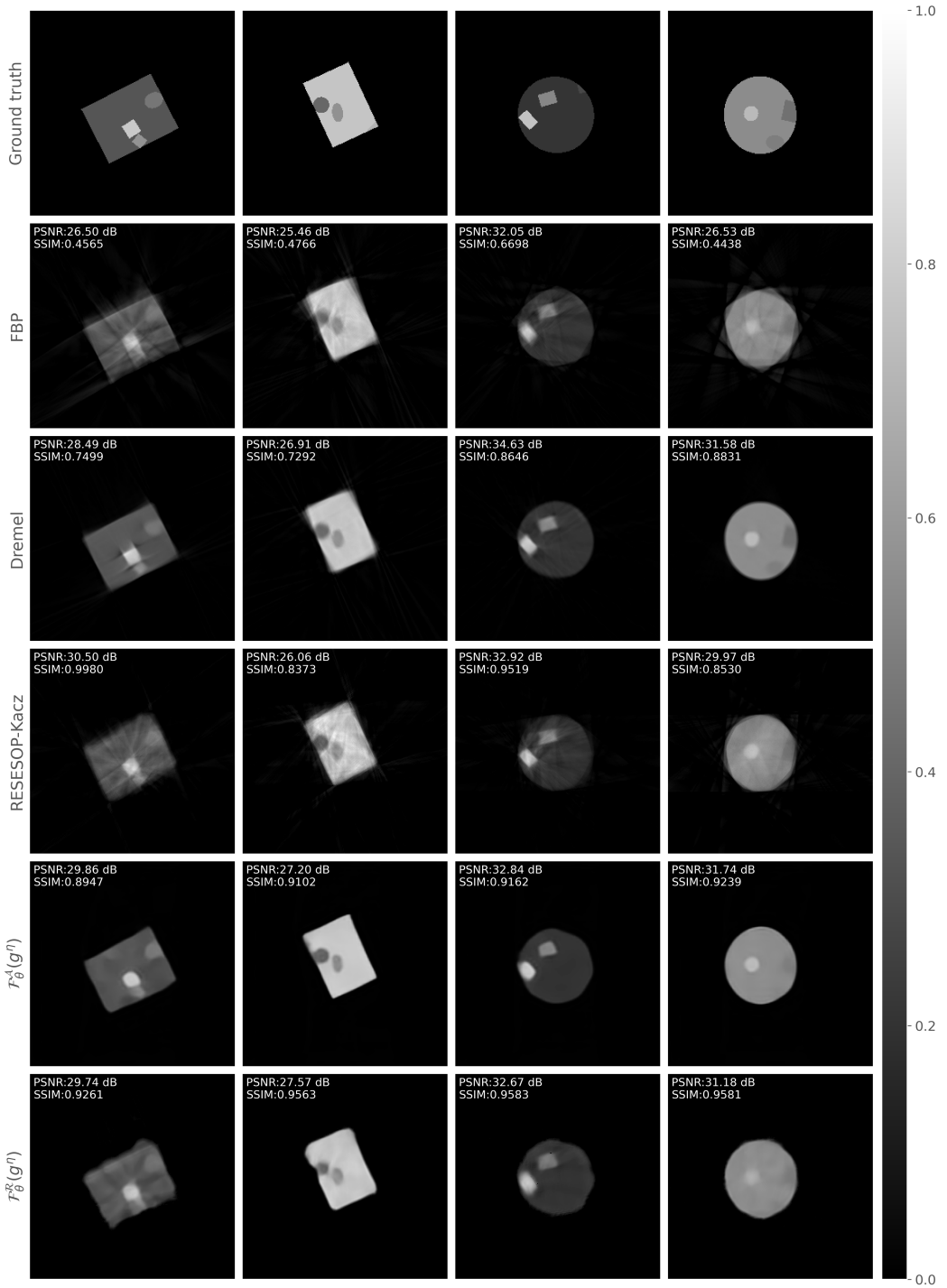


Fig. 6.7.: Reconstructions of measurement data perturbed with vibrations by all considered reconstruction methods.

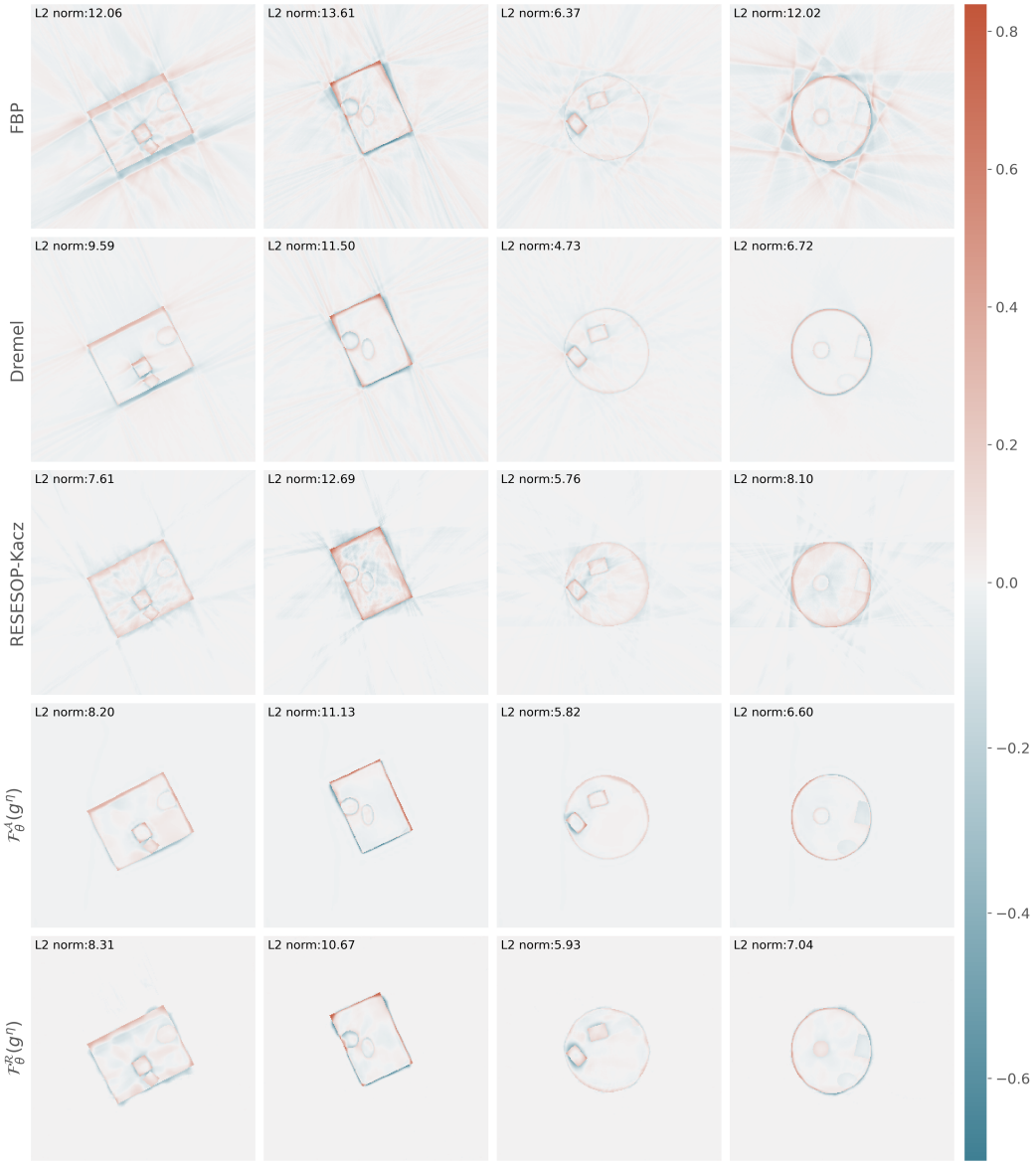


Fig. 6.8.: Differences between ground truths and reconstructions of measurement data perturbed with vibrations by all considered reconstruction methods.

Fully Learned Operator Network

The general iteration step of this architecture is defined in (4.3) as

$$f_{n+1} = f_n - \mathcal{N}^{\text{sd}}_{\theta^1_n}(f_n) - \mathcal{N}^{\text{sd}}_{\theta^2_n}(f_{n-1}) + \mathcal{N}^{\text{d}}_{\theta^0_n}(g^\eta).$$

The FLO model was trained on the linear shift data with 3 unrolled iterations, 2 search directions, a search direction depth of 1 and a g^η depth of 2. In experiments on the vibration data, FBP served as a good initialization to the model. However, the total movement in the linear shift data is more significant, causing the FBP reconstructions to be of much lower quality. In turn, these reconstructions might no longer be useful to the model. In Table 6.10 shows the quantitative results of two models, one trained with FBP initialization and one with random initialization, both trained for 100 epochs.

f_0	data	training	PSNR [db]	SSIM
FBP	val	3h 20min	27.44 ± 2.56	0.302 ± 0.014
	test	-	$\mathbf{27.07 \pm 2.56}$	0.303 ± 0.014
random	val	2h 48min	25.32 ± 2.52	0.653 ± 0.017
	test	-	24.95 ± 2.41	$\mathbf{0.653 \pm 0.016}$

Tab. 6.10.: Quantitative evaluation of fully learned operator network with 3 unrolled iterations, 2 search directions, search direction depth of 1 and g^η depth of 2. The model was evaluated once with FBP initialization and once with a random initialization on the validation and test datasets for measurements obtained from perturbed sinograms. The best test results for both measures have been highlighted.

Evaluating the model with random initialization on the test data split shows that the model can generalize well. SSIM and PSNR for validation and test data are comparable. However, it is notable that the performance of the architecture on this dataset is worse than on the vibration dataset. On the previous dataset, this architecture achieved a PSNR of 31.61 and an SSIM of 0.92. For the linear shift data, this has dropped to 24.95 dB and an SSIM of 0.653. We have previously shown that the architectures can reconstruct unperturbed measurement data and measurement data perturbed by vibration motion well. However, the model inexactness introduced by the linear shift motion seems to be too great to reconstruct the objects well.

Radon Network

The iteration of the Radon network, $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$, is written as

$$f_{n+1} = f_n - \sum_{i \in I_n} \mathcal{N}_{\theta_n^i}(\mathcal{R}^T(\mathcal{R}f_i - g^\eta)).$$

We will now consider the Radon network architecture, which has also been able to reconstruct from the vibration data, although not as well as the FLO model. As for the vibration data, this architecture has been trained on the linear shift data with a network with 4 unrolled iterations, 3 search directions, and a search direction depth of 1. Random initialization has not been tested because this architecture applies the Radon transform in each search direction and iteration, and thus reintroduces the motion artifacts, even if f_0 does not. Table 6.11 shows the quantitative results of this model on the validation and test data.

data	training time	PSNR [db]	SSIM	rel. loss diff.
val	10h 58min	23.16 ± 2.87	0.820 ± 0.017	0.2892
test	-	23.03 ± 2.69	0.819 ± 0.017	0.2892

Tab. 6.11.: Quantitative evaluation of Radon network with 4 unrolled iterations, 3 search directions and a search direction depth of 1. The model was evaluated on validation and test data for measurements obtained from perturbed sinograms using FBP as an initial reconstruction.

For the vibration data, this model achieved a PSNR of 31 dB and an SSIM of 0.958. On the linear shift data, the PSNR of this model is 23.03 dB and the SSIM is 0.819. Compared with the FLO model, the PSNR has dropped slightly; however, the SSIM is still quite high. Here, the Radon model outperforms the FLO model by 0.166. This is surprising as the PSNR is quite low. More detailed evaluation of the reconstructions is necessary to understand why the SSIM value is that high.

Comparison of models

Let us first review the FBP reconstructions for these samples in Figure 6.9 before comparing the two FLO models. These reconstructions also provide information about the extent of the object's movement. The reconstructions of the second and third samples show that the movements were smaller than those of the first and last samples, where the objects are not recognizable.

In Figure 6.10 the reconstructions of the FLO models with FBP and random initialization are shown together with the reconstructions of the Radon model. It is

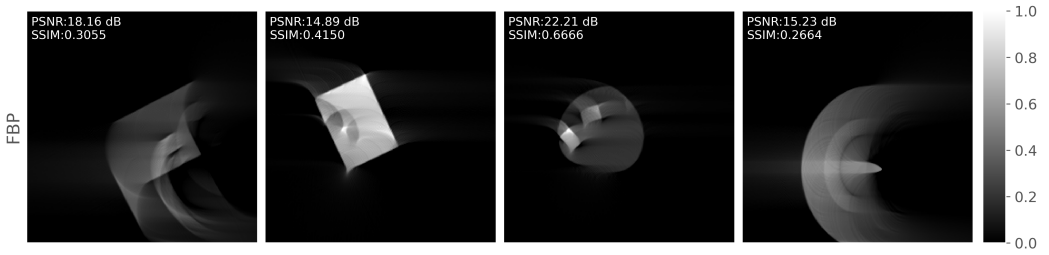


Fig. 6.9.: The FBP reconstructions of measurement data perturbed with linear shifts.

immediately clear that these reconstructions are not satisfactory. We can see that in certain cases initialization with FBP improved the FLO model in a meaningful way, as this is the only model that could somewhat reconstruct the shapes within the objects of the first and third samples. Given that the FBP reconstruction of the third sample was already relatively clear, it makes sense that using it to initialize the FLO model resulted in the best reconstruction. This improved PSNR by 4.24 dB over the original FBP reconstruction, but SSIM was significantly reduced by 0.3584. Even in the first sample, where the FBP reconstruction appears unusable, initializing the FLO model with it results in an increase in the PSNR of 2.63 dB over random initialization. Although the reconstructions of the FLO model initialized by the FBP perform visually better than random initialization, the SSIM score is consistently worse. Looking at the difference in density for each of the reconstruction methods in Figure 6.11, it can be seen that initialization with the FBP results in a more accurate density than the randomly initialized FLO model or the Radon model. Looking at the first and fourth samples, it can be seen that the features inside the sample are more effectively reproduced with FBP initialization than with a random initialization, but in sample two it resulted in an internal artifact. Although the reconstruction of the FLO network is certainly not great, it is possible to recognize the shape of the object. The darker areas inside reveal that the internal shapes are not properly represented in this reconstruction. In the second sample, a major artifact can be seen. This main object was reconstructed relatively well; however, the model places a very large elliptical shape with pixel values close to 1 in the top part of the object. Despite this large artifact, the PSNR has improved compared to the FBP reconstruction by 9.32 while the SSIM has decreased by 0.1061. This is likely due to the high intensity of the artifact.

Evaluating the reconstructions of the FLO model that were initialized using a random image, it is apparent that the object shapes in all samples have been reconstructed only at a basic level, with the internal shapes not being visible. The Radon model reconstructions exhibit a somewhat circular form, with portions of the objects' edges visible within these circular patterns. This model achieved the highest SSIM values;

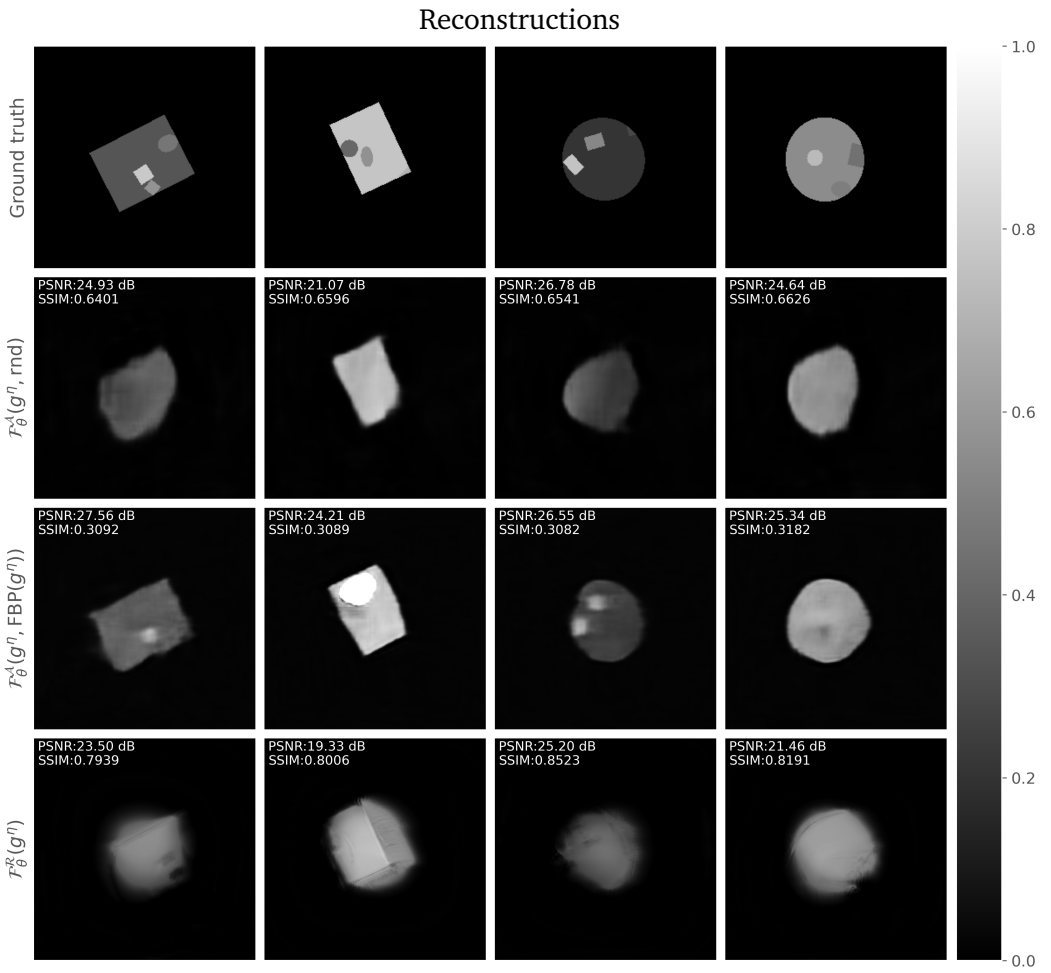


Fig. 6.10.: Four reconstructions of measurement data perturbed with linear shifts for the fully learned operator model $\mathcal{F}_\theta^{\mathcal{A}}(g^\eta)$ with FBP and random initialization and the Radon model $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$.

however, it is not clear why this is the case. Even when considering the images in Figure 6.11, it is obvious that the reconstructions of the Radon model are much worse than the reconstructions of the FLO model initialized with FBP reconstructions.

Comparison with Other Reconstruction Methods

We now extend our analysis to include the non-learned reconstruction methods. In previous results, it was clear that reconstructions of these perturbations suffer from considerable distortions. This is also reflected in the results of the non-learned methods. As can be seen in Table 6.12, the PSNR and SSIM values of the FBP and Dremel reconstructions are the lowest. For these methods, we will see that the objects in the reconstructions are shifted due to the severe linear shift in the

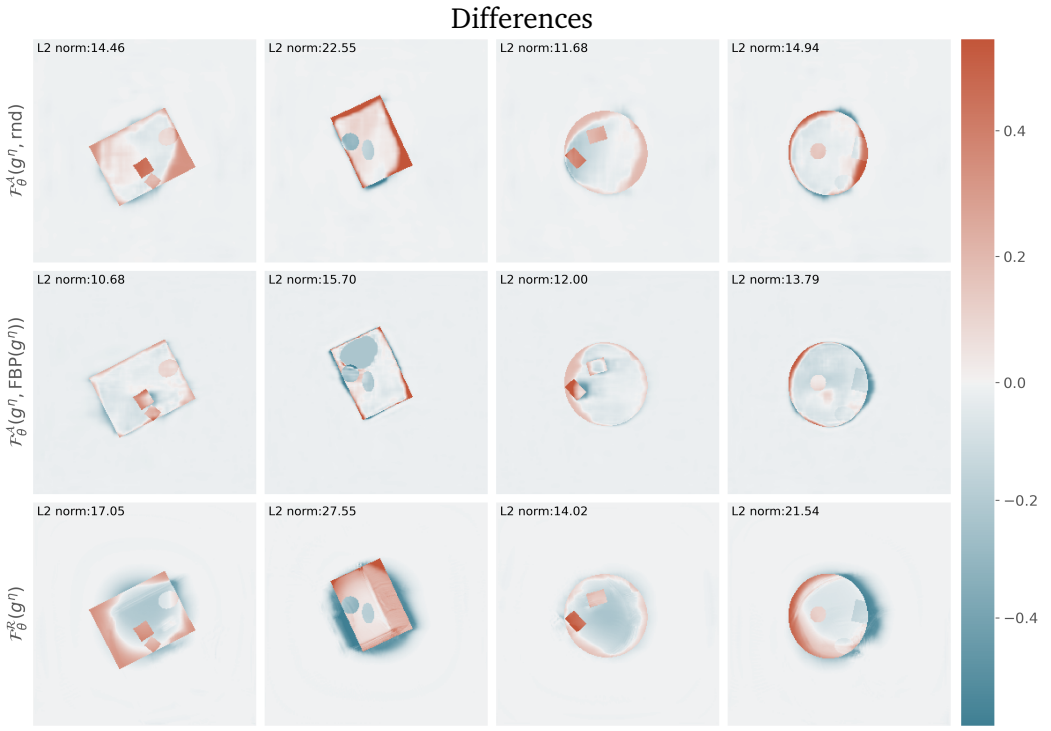


Fig. 6.11.: Differences between ground truths and reconstructions of measurement data perturbed with linear shifts for the fully learned operator model $\mathcal{F}_\theta^{\mathcal{A}}(g^\eta)$ initialized with FBP reconstructions and random values and the Radon model $\mathcal{F}_\theta^{\mathcal{R}}(g^\eta)$.

measurement data. To be able to better evaluate the quality of the objects instead of their location, an automated image translation has been performed by applying a pixel cross-correlation between the ground truths and the reconstructions. This algorithm is available in the scikit Python package [120]. The shifted results of both methods perform better compared to the original reconstructions. However, some of the objects are so distorted that the image translation did not work effectively.

The FLO model and the Radon model have the highest PSNR and SSIM values, respectively. However, in the previous section, we noticed that the SSIM does not accurately reflect the performance of the Radon model on this dataset. The results of the RESESOP-Kaczmarz method are averaged over 100 samples of the test data; all other methods have been evaluated on the 323 samples in the test data. This choice was made because of the extensive computation time of this method.

We can now visually compare the reconstructions in Figure 6.12. Similarly to the vibration data, the Dremel method delivers the visually best results, where linear translation is low, but when the impact of the motion is higher, the results appear smeared or distorted. The first sample has such a large shift that none of the non-learned methods reconstructed this well. The second sample has been better

	PSNR [dB]	SSIM
FBP	17.71 ± 3.63	0.395 ± 0.144
FBP shifted	20.46 ± 4.22	0.480 ± 0.118
Dremel	17.41 ± 3.79	0.482 ± 0.150
Dremel shifted	21.50 ± 4.84	0.581 ± 0.132
RESESOP-Kacz. with η	22.88 ± 3.37	0.777 ± 0.055
Fully learned op. net	$\mathbf{24.95 \pm 2.41}$	0.653 ± 0.016
Radon net	23.03 ± 2.69	$\mathbf{0.819 \pm 0.017}$

Tab. 6.12.: Quantitative evaluation on linear test datasets for measurements obtained from perturbed sinograms. RESESOP-Kaczmarz is evaluated on 100 samples.

reconstructed by Dremel and FBP, where the latter reconstruction has more artifacts. The reconstructions of the third sample are quite distorted by both the FBP and Dremel method, and the last reconstruction has been very smeared by the FBP and is very distorted by the Dremel method. Before moving on to the figures of the differences between ground truths and reconstructions, we consider the RESESOP-Kaczmarz reconstructions. Despite the PSNR and SSIM results, none of the shapes have been reconstructed. It is clear that this method was not able to compensate for the linear shifts.

Let us review the differences of the phantoms and reconstructions in Figure 6.13. The difference figures clearly show that all of the objects that are relatively good reconstructions have been slightly shifted. This could indicate that FBP and Dremel have reconstructed different time steps of the scan process. In comparison with the FLO model, there are more artifacts in the non-learned reconstructions in addition to the incorrect locations of the reconstructions.

Overall, we can summarize that the FLO model clearly outperforms any of the other methods even when the measurement data were perturbed by large shifts. This model consistently reconstructed the main shape and was able to reconstruct some of the inner shapes in cases with large perturbations where other methods did not deliver usable reconstructions.

6.1.3 Experiments with Unperturbed Data

To put the results for the vibration and linear shift data in perspective, we consider the performance of all methods on unperturbed measurements g, without any motion or noise, as a baseline.

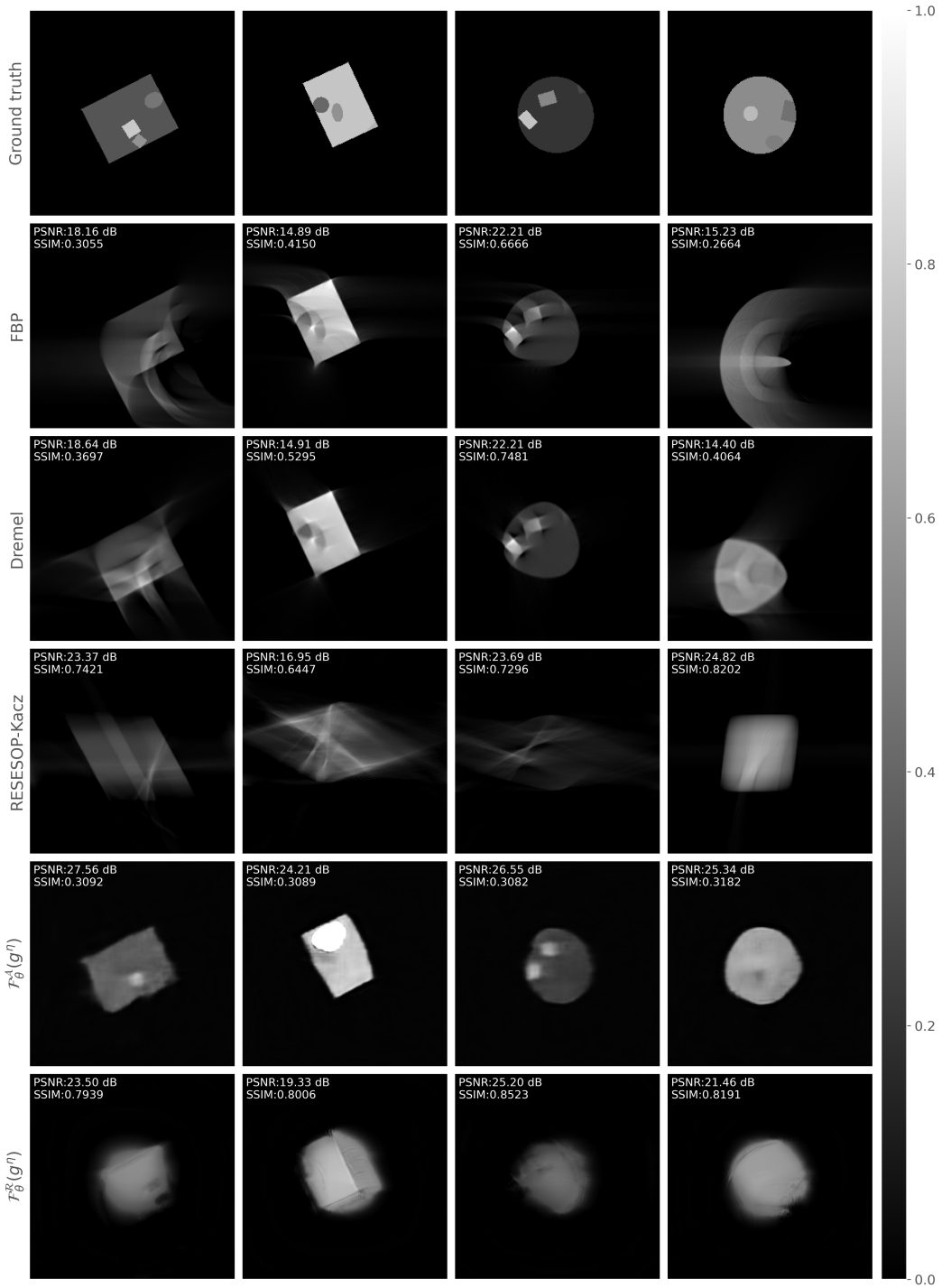


Fig. 6.12.: Reconstructions of measurement data perturbed with linear shifts by all considered reconstruction methods.

See Table 6.13 for a quantitative representation of the results. Unsurprisingly, the performance of both models is very good on the unpertrubed data. The FLO model

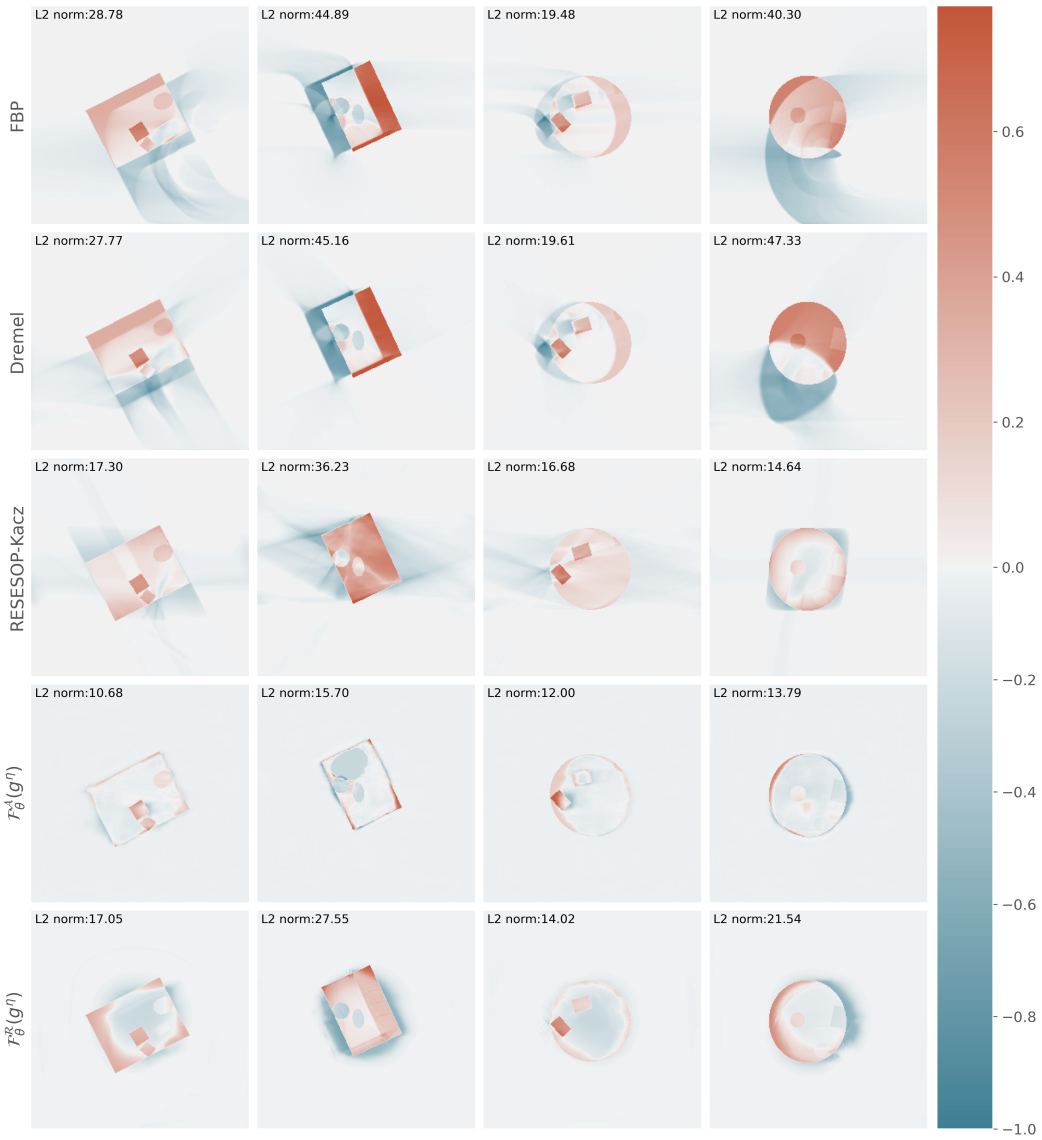


Fig. 6.13.: Difference of reconstructions to ground truth of measurement data perturbed with linear shifts by all considered reconstruction methods.

could achieve a PSNR of 41.47 dB and an SSIM of 0.949 on the test data while the Radon model performed slightly worse according to the PSNR with 41.13 dB. However, the SSIM of the Radon model is only 0.855, which is worse than the results of the perturbed model. This model was trained only for 6 epochs, as the validation error quickly stagnated. Thus, the training time was very short. The FLO model, on the other hand, trained for 100 epochs.

The non-learned reconstruction methods also performed very well on the unperturbed measurement data. The RESESOP-Kaczmarz method performed the best in both measures, followed by both learned methods and FBP. The Dremel method per-

	training time	PSNR [dB]	SSIM
FBP	-	40.10 ± 2.60	0.967 ± 0.022
Dremel	-	35.29 ± 2.81	0.945 ± 0.028
RESESOP-Kacz. ($\eta = 0$)	-	$\mathbf{43.96 \pm 2.34}$	$\mathbf{0.980 \pm 0.078}$
Fully learned op. net	11h 20min	41.47 ± 1.85	0.949 ± 0.001
Radon net	3h 35min	41.13 ± 2.19	0.855 ± 0.009

Tab. 6.13.: Quantitative evaluation on vibration test datasets for measurements obtained from unperturbed sinograms.

formed the worst on this dataset. In each iteration of this method, a cross-correlation is performed between the projection of the current operator and the current reconstruction $\mathcal{A}_k f$ and the measurement data g_k^δ to calculate the necessary shift of the operator. However, the measurement data in this dataset do not include any motion. The cross-correlation seems to over-correct here, and thus the reconstructions of this method are not as good as the ones by other methods. Figures 6.14 and 6.15 show the reconstructions and their differences to the ground truth. Due to the high quality of the results, it is very difficult to see artifacts in the reconstructions by only reviewing images in Figure 6.14. However, Figure 6.15 clearly shows that the edges of the Dremel reconstructions are not as clear as for other methods. In addition, these reconstructions include artifacts outside of the objects.

In these images, it is easy to see that the FLO architecture achieved higher quality reconstructions than the Radon architecture. The difference plots for the Radon model show that the density of several shapes has been incorrectly predicted as well as some of the edges. This difference in density might be the reason for the poor SSIM score, due to its sensitivity towards contrast. Furthermore, the edges of the reconstruction made by the FLO model appear grainy. When considering these areas in Figure 6.14, there appears to be a grid-like coloration in some of the shapes. This could be related to the upsampling layers used in this architecture. These layers apply interpolation followed by convolutions. A possible reason for these grid-like artifacts could be that the convolutions did not sufficiently correct for the interpolation step.
Overall, FBP, RESESOP-Kaczmarz, and the FLO model deliver very good results, while the Radon model did not reconstruct the intensities as well.

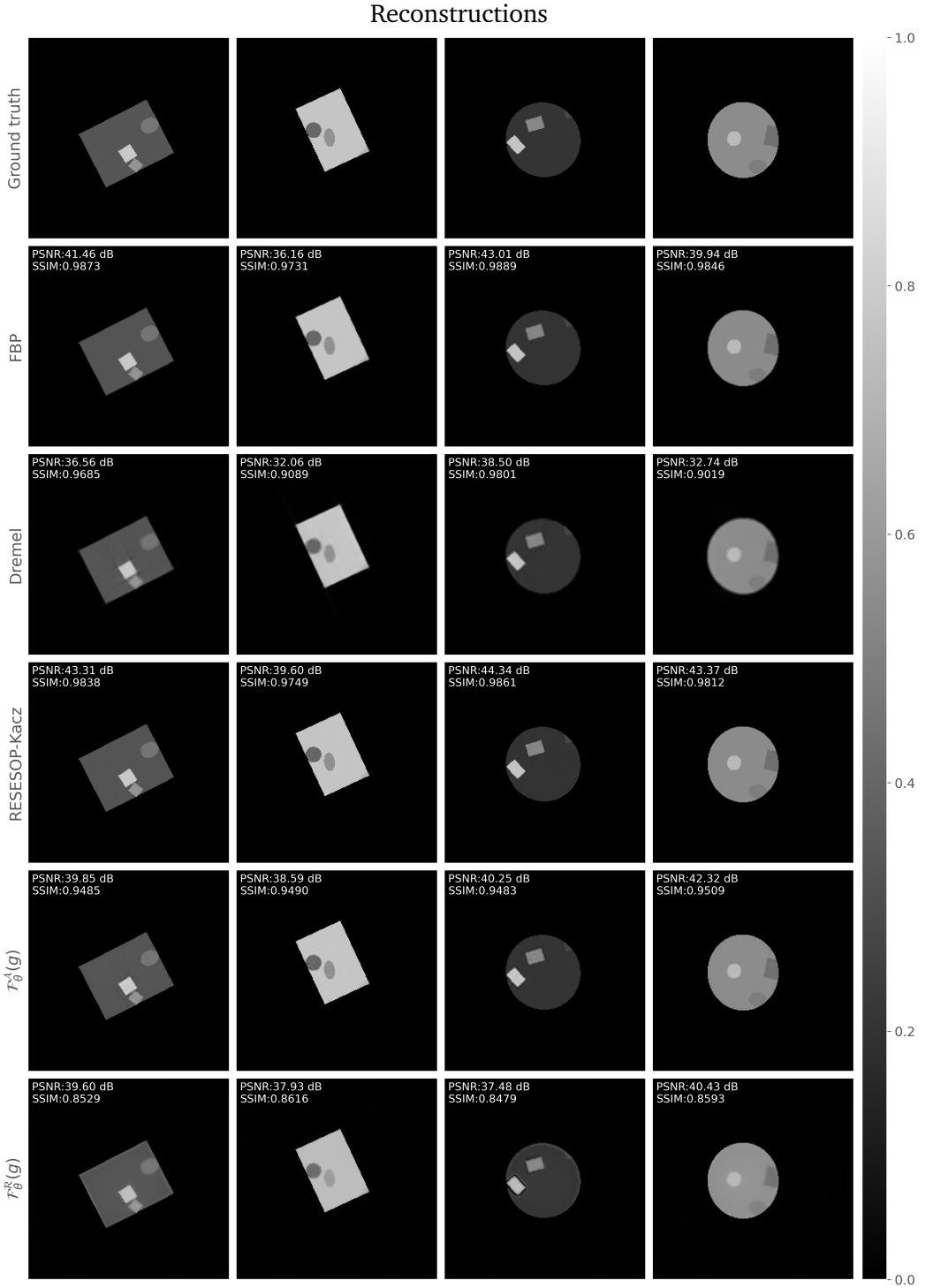


Fig. 6.14.: Reconstructions of unperturbed measurement data by all considered reconstruction methods.

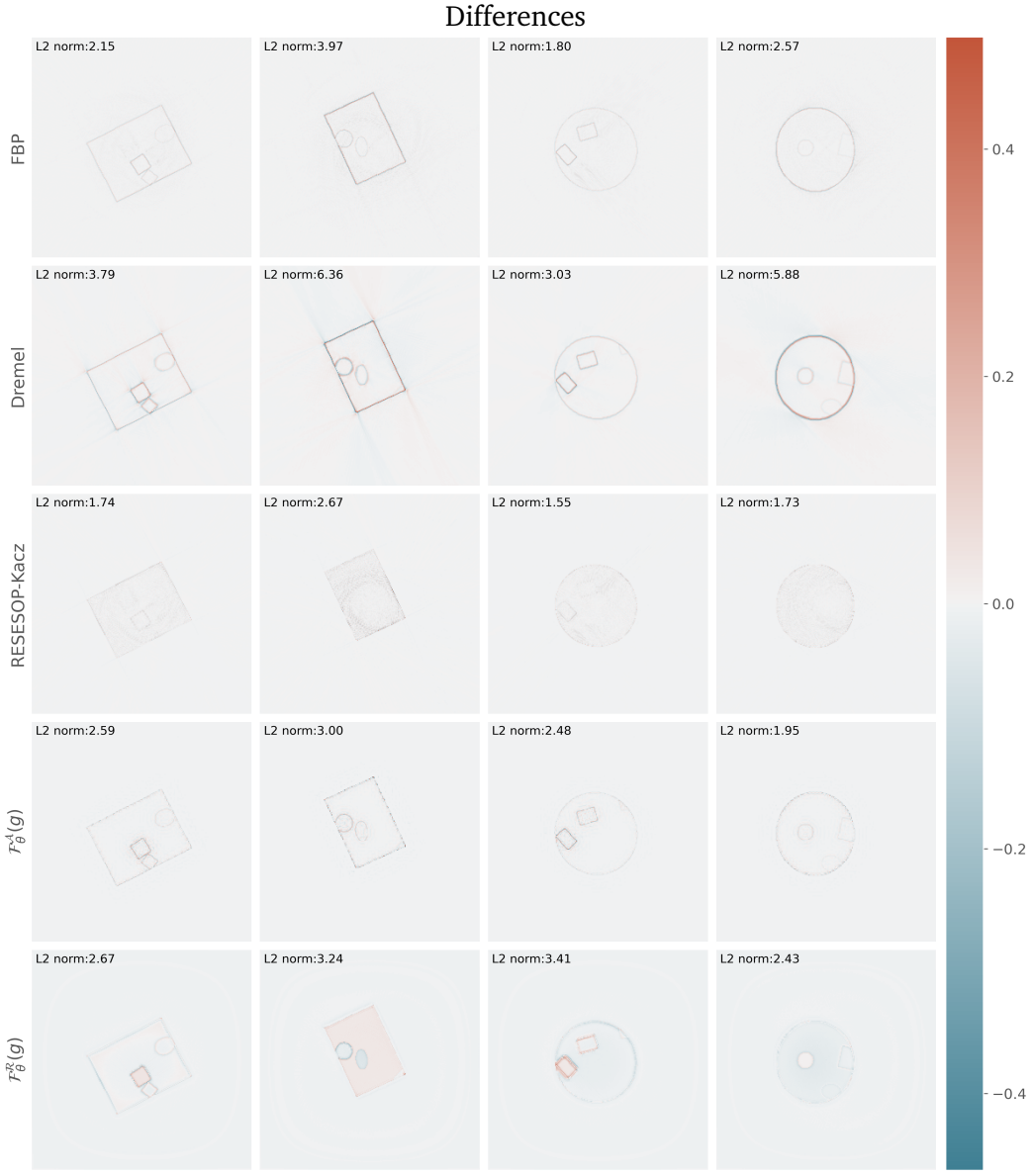


Fig. 6.15.: Difference of reconstructions to ground truth of unperturbed measurement data by all considered reconstruction methods.

6.1.4 Network experiments with LoDoPaB Data

Finally, we apply the architectures to non-dynamic measurement data. The LoDoPaB dataset includes low-dose medical CT data; therefore, the shapes and intensities are very different from the simulated datasets. As stated in the beginning of this chapter, the RESESOP-Kaczmarz and Dremel methods are not suitable reconstruction techniques for non-dynamic measurement data. The results of the Dremel method applied to the unperturbed data showed that the reconstructions had blurry edges and some artifacts. Although the RESESOP-Kaczmarz algorithm performed well on motion-free measurement data, its extensive computation time renders it impractical as a viable option. The performance of the learned models will therefore be compared with the FBP reconstructions.

The quantitative results in Table 6.14 show that both learned methods perform better than the FBP.

	training time	PSNR [dB]	SSIM
FBP	-	24.70 ± 1.23	0.787 ± 0.101
Fully learned op. net	10h 36min	$\mathbf{37.55 \pm 2.61}$	0.895 ± 0.074
Radon net	19h 19min	37.44 ± 2.39	$\mathbf{0.900 \pm 0.072}$

Tab. 6.14.: Quantitative evaluation of FLO network $\mathcal{F}_\theta^A(g)$ with 3 unrolled iterations, 2 search directions, search direction depth of 1 and g depth of 2; and of Radon network with 4 unrolled iterations, 3 search directions and a search direction depth of 1; and FBP. The models were trained and evaluated on validation and test data in the LoDoPaB dataset.

The FLO model achieved a PSNR of 37.55 dB, closely followed by the Radon model with 37.22 dB. The difference in performance between the Radon model and the FLO model according to the SSIM is marginal. The Radon model reached an SSIM of 0.9 while the FLO model achieved 0.895. According to both measures, the FBP performed worse with a PSNR of 24.7 dB and an SSIM of 0.787.

Now, we will consider the visual performance of the reconstruction techniques on randomly selected samples, see Figure 6.16. The ground truth images indicate that the measurement data were obtained from scans of the human thorax. A patch of 100×100 pixels was chosen from each image and enlarged to facilitate a detailed comparison of the reconstructions. The FBP-reconstructed image patches show a grainy texture resulting from the quantum noise. The Radon reconstructions exhibit a less grainy texture and higher contrast. The best reconstructions have been produced by the FLO model, where no grainy texture is present, which improves

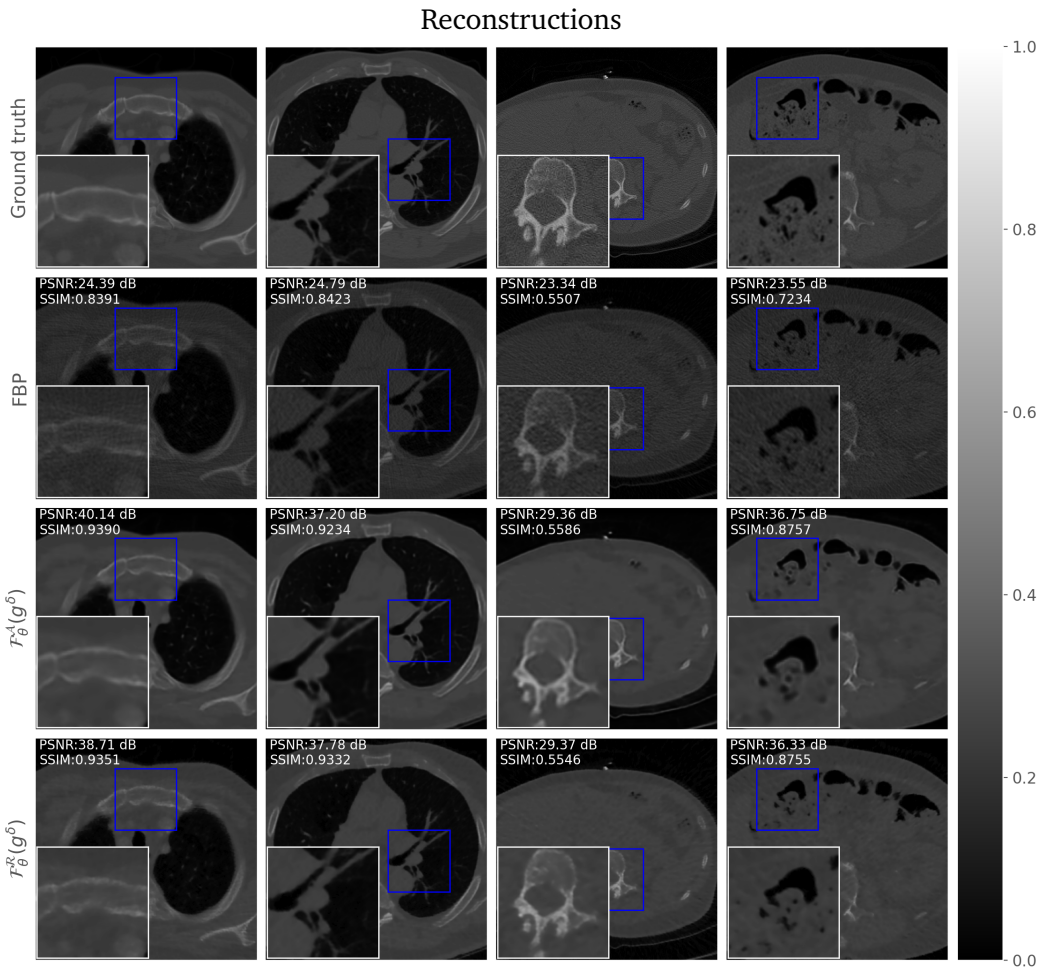


Fig. 6.16.: Reconstructions of LoDoPaB data by all considered reconstruction methods.

the clarity of the reconstructions. The only sample that could not be improved to a PSNR of more than 30 dB is the third sample. The PSNR values of all samples improved by at least 6.02 dB and up to 15.75 dB, while the SSIM values improved by at least 0.0079 and up to 0.15.

Some areas appear slightly blurred, as can best be seen in the last sample. There are very fine details in the sponge-like structure of the ground truth that have not been reconstructed clearly. Some of the small dark spots are blurred and gray. These details are easily discernible in Figure 6.17. Note that the color scales for blue and red are not symmetric as positive values reach 0.6, while negative values reach -0.17. This means that positive and negative values cannot be compared by shade of color. The most prominent detail of the differences between ground truths and reconstructions is that the pixel values of the FBP reconstructions are all smaller than in the ground truth. In Section 5.2.2 the frequency scaling of these reconstructions

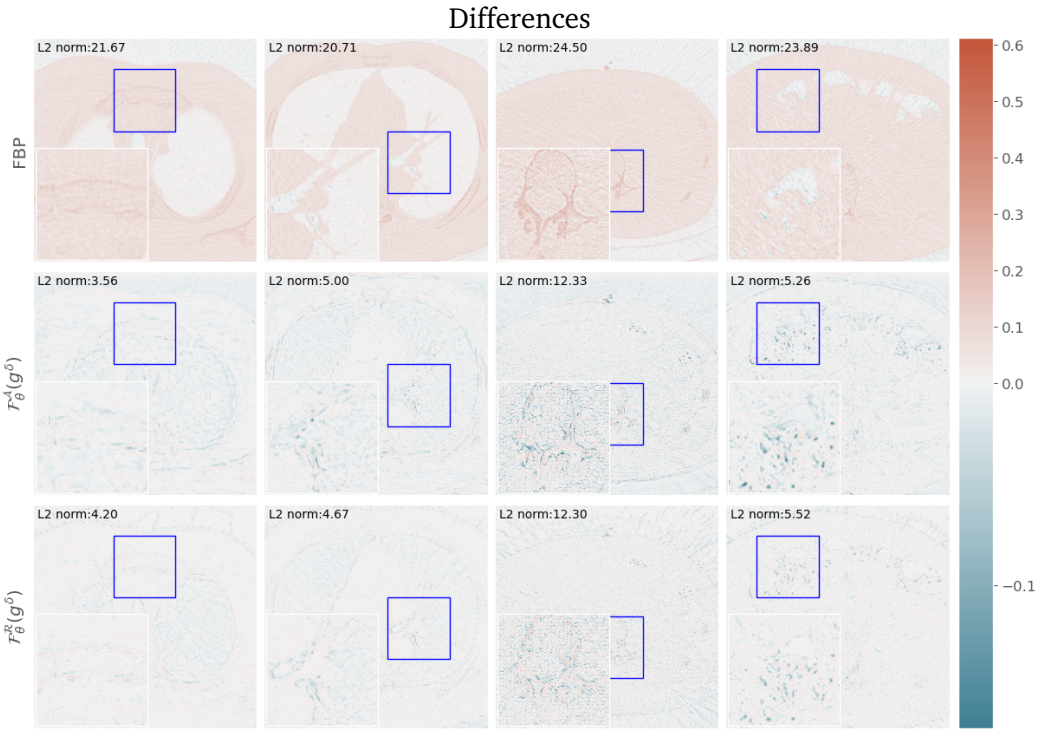


Fig. 6.17.: Difference of reconstructions to ground truth of unperturbed measurement data by all considered reconstruction methods.

is discussed. Due to the low-dose measurement data, the intensity values of the reconstructions are also low. The reconstructions were scaled by a factor of 5.54 to restore the total attenuation of the object. The difference figures now show that the intensities are still under-scaled compared to the ground truth. In the figures of the learned reconstructions, we see that most intensity values have been overestimated. Especially in the last two samples the differences between the blurry reconstructions and the ground truth are highlighted. In general, the model reconstructions are of high quality.

To put the results of these architectures on the LoDoPaB dataset into perspective, we consider the performance of other learned reconstruction methods as reported in [68]. Leuschner et al. [68] conducted a quantitative analysis of the results of several deep learning-based reconstruction methods on the LoDoPaB dataset. The best three of the models presented are the learned primal-dual network [3], which has been discussed in Algorithm 15; the U-Net [98] also discussed in Section 3.2.3 and the ISTA U-Net [75], an unrolled network based on the iterative ISTA algorithm (see Algorithm 7) and the U-Net. The results of these methods are summarized in Table 6.15.

Model	PSNR [dB]	SSIM	Learned parameters
Learned P.-D.	40.52 ± 3.64	0.926 ± 0.076	874 980
ISTA U-Net	40.36 ± 3.65	0.924 ± 0.080	83 396 865
U-Net	40.28 ± 3.59	0.923 ± 0.079	613 322

Tab. 6.15.: The best three learned models as reported in [68].

These results show that there are models that are much more suitable for reconstructing the LoDoPaB measurement data. However, a direct comparison is difficult, as we trained the FLO and Radon architectures on a fraction of the available LoDoPaB data. The results in Table 6.15 are based on models that have been trained on the entire training dataset and are tested on the *challenge dataset*, which is a separate test dataset. Furthermore, we cannot draw any conclusions about the performance of these architectures on dynamic CT data.

6.2 Computation cost

In this section, the resource requirements of the methods are compared. Table 6.16 lists the different features of all reconstruction techniques. Starting with the non-learned methods, we can see that the RESESOP-Kaczmarz method has a much higher inference time than any of the other methods. All non-learned methods require less than 30 GB or memory. The inference times of all methods have been evaluated on the test data of the linear shift dataset and estimated for a single sample. The inference time range, that is, the time it takes to calculate a single reconstruction, for the RESESOP-Kaczmarz method is due to fluctuations in the compute units used. The samples were reconstructed over a period of several weeks, in which the availability of the compute units fluctuated. The training times of the learned models are also based on the linear shift dataset. Training and inference times are dependent on the size of the measurement data and reconstructions, as well as the size of the dataset. We can see that the FLO model has many more learnable parameters than the Radon model, that is $4.8 \cdot 10^6$ and $11.5 \cdot 10^3$, respectively. This is due to the complex architecture. The total size of the FLO model is 17.52 GB and the total Radon model has a size of 13.22 GB. The latter model also includes the Radon transform.

Method	Training time	Inference time	Learned parameters	Model size in GB	GPU or CPU	Memory in GB
FBP	-	< 1 sec.	-	-	CPU	16
Dremel	-	< 1 min.	-	-	GPU	24
RESESOP-Kacz.	-	$6 - 12$ days	-	-	CPU	12
FLO model	< 4h	< 1 sec.	4,882,512	17.52	GPU	< 40
Radon model	11h	< 1 sec.	11,556	13.22	GPU	< 40

Tab. 6.16.: Summary of method requirements, training and inference times of the discussed reconstruction methods. Training and inference times are based on evaluation of a single sample of the test data in the linear shift dataset.

7 Discussion

The network architectures presented in Section 3.5 have been tested with many different parameter settings on three datasets. These datasets included simulated objects with different types of movements as well as low-dose medical CT images without movement. While there was no single winner for all situations, **we can confidently say that the learned models outperformed the other reconstruction techniques in almost all settings**. This chapter will draw further conclusions and highlights important considerations.

7.1 Performance Measures

Let us begin with the performance measures that are frequently used to evaluate CT reconstructions. In Section 5.4, the PSNR and SSIM were introduced and it was hypothesized that SSIM better reflects human perception. However, the results in Chapter 6 show that in many cases the Radon model performed the best according to the SSIM, but a visual comparison of its reconstructions revealed many shortcomings compared to other methods. Reconstructions of the vibration data showed that while more edge pixels were correctly aligned, edges were not as uniform as for the FLO model; see Figure 6.7. However, SSIM suggested that the Radon model performed better than the FLO model.

The linear shift dataset caused an even larger discrepancy between the SSIM performance and a visual evaluation. The SSIM of the Radon model is higher by 0.17 compared to the FLO model, but the reconstructions (see Figure 6.12) revealed that the Radon reconstructions barely resemble the original objects. Some of the PSNR values also contradicted the visual analysis of the reconstruction. In Figure 6.13 the RESESOP-Kaczmarz reconstructions of the linear shift data are shown. This method performed better than FBP and Dremel according to the PSNR score. However, the reconstructions of this method do not resemble the objects shown in the ground truth images. In Section 5.4, it was discussed that PSNR and MSE do not always reflect the quality of a reconstruction appropriately. Figure 5.5 demonstrates such a case.

It is crucial to identify the features that must be accurately reconstructed. Due to the movement of the objects, blurring of edges or a spatial shift in the reconstructed objects relative to the actual reference data were most likely. Thus, for the objects in the simulated datasets, the edges and spatial positioning of the shapes were the most important. Specific intensity values were deemed less critical, provided that the contrast between distinct shapes was adequately captured. It is essential to consider the most important features of the objects, as this should influence the choice of performance measure used. Furthermore, the contradiction between SSIM scores and visual assessment raises the question if SSIM is a suitable loss function considering the type of objects included in the data.

7.2 Impact of Data

The results across different dynamic perturbations showed that the extent of the movement should inform the decision on which reconstruction method should be applied. For example, the reconstructions of the vibration data applying the learned methods were only slightly improved compared to the Dremel and RESESOP-Kaczmarz methods. Taking into account that RESESOP-Kaczmarz requires a priori information about the model inexactness η the practicality of this method is reduced. However, the Dremel method does not require any knowledge of the inexactness of the model. This method performed very well on the vibration data (see Table 6.9); however, when the movements of the object had a greater impact on the measurement data, the results were not as good (see Table 6.12) and when there was no movement, this method performed worse than FBP (see Table 6.13). Thus, we can form the hypothesis that the Dremel technique is a highly suitable option for reconstructions involving movements that are definitely present but of limited magnitude.

The learned methods showed promising results for all datasets. However, the greatest impact was recorded for the linear shift dataset. More work is needed to improve the reconstructions for this type of data; however, the FLO model showed promising results. The reconstructions in Figure 6.12 showed that despite the large translations of the object during the scan, the general structures and some of the internal shapes could be restored. One problem of this model is that it creates artifacts inside of the objects. They are easily recognized because their intensity values are much higher than those of other features in the object.

Additionally, it was demonstrated that the model architectures could generate reconstructions surpassing those of FBP. The findings on the LoDoPaB dataset

demonstrate that this model is not solely restricted to handling dynamic objects; it is also applicable to measurement data experiencing a different type of noise, specifically low-dose data with elevated quantum noise. Performance measures were not as high as for other learned reconstruction methods as reported in [68] and listed in Table 6.15. However, considering that the architectures have not been tailored for this application, the results show that there is potential to apply these architectures to a wider range of problems.

An important consideration when choosing a reconstruction method for given measurement data is the availability of a dataset. The learned methods require a large amount of measurement data paired with ground truths. Collecting this amount of data is time consuming and costly. The LoDoPaB data are a great collection of medical CT images that include curated ground truths where the low-dose measurement data were created by manually applying noise. In the case of dynamic CT, knowledge about the movement is necessary to further simulate the perturbations on the measurement data. This noise may be unique to an appliance and location, as different CT scanners might cause varying drifts of the object. This dependence of the neural network on the training data can be a disadvantage to other techniques, such as the Dremel method, that are not dependent on such a dataset.

7.3 Computation Requirements

In Section 6.2 the computation costs were summarized for all methods. Although the learned methods require a training phase, the inference time is very fast. This is particularly useful when many reconstructions are required.
The RESESOP-Kaczmarz method has a very long reconstruction time because it incorporates the local errors, depending on the scan angles and detector points. All non-learned methods are relatively memory efficient, requiring less than 30 GB of memory, which makes them suitable for a broader range of computational environments without specialized hardware. Between the learned methods, the FLO model has a substantially higher number of learnable parameters compared to the Radon model. This larger parameter count leads to a larger total model size (17.52 GB), making it more resource-intensive to train. However, this model was also the most promising across all datasets and object movements. Even though the Radon model requires fewer parameters, it demands many applications of the Radon transform. We believe that this is the cause of the increased memory consumption, which is comparable to the FLO model.

7.4 General Characteristics of Unrolled Networks

Unlike many "black-box" neural networks, unrolled networks inherit the interpretable structure of the original algorithm. These models are generally easier to understand and require fewer parameters than conventional fully-connected network models [79]. The inference time of neural networks is very fast, whereas iterative algorithms often take a long time to converge to a solution.

Despite the strong connection, comparing a traditional iterative algorithm to an unrolled neural network can be challenging. Many traditional iterative algorithms have been proven to converge to a minimum norm solution. A stopping criterion is used to terminate the iteration when the residual error $\|\mathcal{A}^\eta f_{n^*} - g^\delta\|$ for the iteration index n^* is smaller than or equal to the noise level $\tau(\eta\rho + \delta)$. However, an unrolled network is a truncated version of the algorithm, where the number of iterations is fixed and cannot be determined by a stopping criterion. Furthermore, its learnable parameters are optimized via backpropagation on a dataset. This fundamental difference in how they are optimized and used makes direct comparison difficult.

The RESESOP method has been proven to be a regularization technique for $\delta \to 0$ with a stop criterion that provides a finite stop index n^* such that f_{n^*} is a regularized solution of $\mathcal{A}^\eta f_{n^*} = g^\delta$ [106, 105, 107]. This method can be applied to a variety of inverse problems. A neural network, on the other hand, is a function that has learned to approximate the solution for a specific data distribution and may not generalize well to out-of-distribution data. Thus, the architectures have been trained on each of the datasets individually. A model trained on vibration data alone is unlikely to perform well when used for linear shift data.

8 Conclusion and Outlook

The application of unrolled neural network models to dynamic computed tomography (CT) reconstruction presents a promising alternative to traditional, purely iterative methods. Although the RESESOP-Kaczmarz algorithm provides a foundation rooted in theoretical guarantees, it is computationally prohibitive for real-time applications due to its high inference time and sensitivity to the estimation of the model inexactness η. Although convergence or regularization guarantees for the Dremel method have not yet been established, its performance has demonstrated a high degree of effectiveness on certain types of perturbations. Additional analysis is required to ascertain the specific motions that this method is capable of handling. In contrast, data-driven unrolled networks such as the proposed FLO model is a powerful alternative, achieving very fast inference times and producing reconstructions that outperform the iterative methods in both tested motion types. This architecture does not rely on any information from the operator or the motion, instead learning solely from the measurement data. This makes it a highly flexible method suitable for real-time reconstructions.

However, the transition to a data-driven framework introduces a fundamental trade-off. Unrolled neural network architectures, while empirically effective, do not inherently preserve the convergence or regularization guarantees of their underlying iterative methods. A direct comparison between these two paradigms with a fixed number of iterations is therefore challenging, as the network's behavior is at most guided by, but not strictly bound to, the original algorithm. The proof of the regularization properties of unrolled neural networks is not obvious and is subject of future research. While experiments with the LoDoPaB data indicate that the model exhibits a robust handling of quantum noise, this remains an empirical observation rather than a formally provable characteristic related to regularization as the data error $\delta \to 0$.

This investigation lays the groundwork for several crucial avenues of future research. A deeper analysis into the impact of different performance measures and loss functions is warranted to determine their influence on the effectiveness of the model and the resulting image properties. Similarly, further investigation is required to systematically characterize the performance of the architecture under a wider range

of motion types, including a more comprehensive analysis of linear shifts and the amount of training data necessary for robust generalization. The specific artifacts observed in the FLO model with linear shifts, in particular, demand a detailed investigation to understand their origin and develop mitigation strategies.

Ultimately, while numerous innovative methods are being explored, a continued investigation into this data-driven approach appears to be a highly promising path forward for dynamic CT reconstruction, as it seeks to bridge the gap between empirical success and the development of a new, data-centric theoretical framework. Future research is necessary to analyze convergence, stability, and regularization properties of unrolled neural networks.

Bibliography

[1] Jonas Adler, Holger Kohr, Axel Ringh, et al. *odlgroup/odl: ODL 0.7.0*. Version v0.7.0. Sept. 2018 (cit. on pp. 119, 121).

[2] Jonas Adler and Ozan Öktem. "Solving ill-posed inverse problems using iterative deep neural networks". In: *Inverse Problems* 33.12 (Nov. 2017), p. 124007 (cit. on pp. 89, 90, 95, 107, 108, 110).

[3] Jonas Adler and Ozan Öktem. "Learned Primal-Dual Reconstruction". In: *IEEE Transactions on Medical Imaging* 37.6 (2018), pp. 1322–1332 (cit. on pp. 85, 87–89, 100, 164).

[4] Samuel G. Armato III, Geoffrey McLennan, Luc Bidaut, et al. "The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A Completed Reference Database of Lung Nodules on CT Scans". In: *Medical Physics* 38.2 (2011), pp. 915–931. eprint: `https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1118/1.3528204` (cit. on p. 120).

[5] Samuel G. Armato III, Geoffrey McLennan, Luc Bidaut, et al. "Data from LIDC-IDRI (Version 4) [Data set]". In: *The Cancer Imaging Archive* (2015) (cit. on p. 120).

[6] Simon Arridge, Pascal Fernsel, and Andreas Hauptmann. "Joint reconstruction and low-rank decomposition for dynamic inverse problems". In: *Inverse Problems and Imaging* 16.3 (2022), pp. 483–523 (cit. on pp. 24, 43).

[7] Pouria Aryan, Santhakumar Sampath, and Hoon Sohn. "An Overview of Non-Destructive Testing Methods for Integrated Circuit Packaging Inspection". In: *Sensors* 18.7 (2018) (cit. on p. 16).

[8] Daniel Otero Baguer, Johannes Leuschner, and Maximilian Schmidt. "Computed tomography reconstruction using deep image prior and learned reconstruction methods". In: *Inverse Problems* 36 (2020) (cit. on p. 82).

[9] Randall Balestriero, Mark Ibrahim, Vlad Sobal, et al. *A Cookbook of Self-Supervised Learning*. 2023. arXiv: `2304.12210 [cs.LG]` (cit. on p. 49).

[10] Andreas Balles, Dominik Müller, Karl-Heinz Hiller, and Astrid Hölzing. "Using high-resolution nano-CT methods to detect and analyze counterfeit semiconductor structures". In: Feb. 2024 (cit. on pp. 16, 17).

[11] Ronny Bergmann, Roland Herzog, Maurício Silva Louzeiro, Daniel Tenbrinck, and José Vidal-Núñez. "Fenchel Duality Theory and a Primal-Dual Algorithm on Riemannian Manifolds". In: *Foundations of Computational Mathematics* 21.6 (Jan. 2021), pp. 1465–1504 (cit. on p. 88).

[12]David Berthelot, Nicholas Carlini, Ian Goodfellow, et al. “MixMatch: A Holistic Approach to Semi-Supervised Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, et al. Vol. 32. Curran Associates, Inc., 2019 (cit. on p. 49).

[13]P. V. S. S. S. Bhaskar, C. A. J. van Leeuwen, J. A. A. Van Gils, and P. P. J. M. van Zeist H. W. H. J. R. W. van der. “Compensation of some time dependent deformations in tomography”. In: *IEEE Transactions on Medical Imaging* 26.2 (2007), pp. 261–269 (cit. on p. 24).

[14]Stephanie E Blanke, Bernadette N Hahn, and Anne Wald. “Inverse problems with inexact forward operator: iterative regularization and application in dynamic imaging”. In: *Inverse Problems* 36.12 (Oct. 2020), p. 124001 (cit. on pp. 23–25, 27, 31, 32, 34, 36–38, 44).

[15]Léon Bottou, Frank E Curtis, and Jorge Nocedal. “Optimization methods for large-scale machine learning”. In: *SIAM review* 60.2 (2018), pp. 223–311 (cit. on pp. 72, 73).

[16]Rostyslav Boutchko, Vitaliy L. Rayz, Nicholas T. Vandehey, et al. “Imaging and modeling of flow in porous media using clinical nuclear emission tomography systems and computational fluid dynamics”. In: *Journal of Applied Geophysics* 76 (2012), pp. 74–81 (cit. on p. 22).

[17]Robert S. Bradley, Ian K. Robinson, and Mohammed Yusuf. “3D X-Ray Nanotomography of Cells Grown on Electrospun Scaffolds”. In: *Macromolecular Bioscience* 17.2 (2017), p. 1600236. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/mabi.201600236` (cit. on p. 16).

[18]T. A. Bubba, M. März, Z. Purisha, M. Lassas, and S. Siltanen. “Shearlet-based regularization in sparse dynamic tomography”. In: *Wavelets and Sparsity XVII*. Ed. by Yue M. Lu, Dimitri Van De Ville, and Manos Papadakis. Vol. 10394. International Society for Optics and Photonics. SPIE, 2017, 103940Y (cit. on p. 24).

[19]Tatiana A Bubba, Gitta Kutyniok, Matti Lassas, et al. “Learning the invisible: a hybrid deep learning-shearlet framework for limited angle computed tomography”. In: *Inverse Problems* 35.6 (May 2019), p. 064002 (cit. on p. 83).

[20]Tatiana A. Bubba, Mathilde Galinier, Matti Lassas, et al. “Deep Neural Networks for Inverse Problems with Pseudodifferential Operators: An Application to Limited-Angle Tomography”. In: *SIAM Journal on Imaging Sciences* 14.2 (2021), pp. 470–505. eprint: `https://doi.org/10.1137/20M1343075` (cit. on p. 84).

[21]Tatiana A. Bubba, Luca Ratti, and Andrea Sebastiani. *Revisiting ΨDONet: microlocally inspired filters for incomplete-data tomographic reconstructions*. 2025. arXiv: `2501.18219 [math.OC]` (cit. on p. 84).

[22]Martin Burger, Hendrik Dirks, Lena Frerking, et al. “A variational reconstruction method for undersampled dynamic x-ray tomography based on physical motion models”. In: *Inverse Problems* 33.12 (Nov. 2017), p. 124008 (cit. on pp. 23, 24).

[23] Martin Burger, Thomas Schuster, and Anne Wald. “Ill-posedness of time-dependent inverse problems in Lebesgue-Bochner spaces”. In: *Inverse Problems* 40.8 (July 2024), p. 85008 (cit. on p. 24).

[24] Jerrold T Bushberg, Anthony J Seibert, Edwin M Leidholdt, and John M Boone. *The essential physics of medical imaging; 3rd ed.* Philadelphia, PA: Lippincott Williams & Wilkins, 2012 (cit. on pp. 19, 20).

[25] Thorsten Buzug. *Computed tomography: From photon statistics to modern cone-beam CT*. English. Springer Berlin Heidelberg, Jan. 2014 (cit. on pp. 7, 121).

[26] Yijun Cao, Fuya Luo, and Yongjie Li. “Toward Better SSIM Loss for Unsupervised Monocular Depth Estimation”. In: *Image and Graphics*. Springer Nature Switzerland, 2023, pp. 81–92 (cit. on p. 130).

[27] Antonin Chambolle and Thomas Pock. “A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging”. In: *J. Math. Imaging Vis.* 40.1 (May 2011), pp. 120–145 (cit. on p. 87).

[28] Chong Chen, Barbara Gris, and Ozan Öktem. “A New Variational Model for Joint Image Reconstruction and Motion Estimation in Spatiotemporal Imaging”. In: *SIAM Journal on Imaging Sciences* 12.4 (2019), pp. 1686–1719 (cit. on p. 23).

[29] Hu Chen, Yi Zhang, Yunjin Chen, et al. “LEARN: Learned Experts’ Assessment-Based Reconstruction Network for Sparse-Data CT”. In: *IEEE Transactions on Medical Imaging* 37.6 (2018), pp. 1333–1347 (cit. on pp. 85, 86, 100).

[30] Hu Chen, Yi Zhang, Mannudeep K. Kalra, et al. “Low-Dose CT With a Residual Encoder-Decoder Convolutional Neural Network”. In: *IEEE Transactions on Medical Imaging* 36 (2017), pp. 2524–2535 (cit. on p. 67).

[31] A. M. Cormack. “Representation of a Function by Its Line Integrals, with Some Radiological Applications”. In: *Journal of Applied Physics* 34.9 (Sept. 1963), pp. 2722–2727. eprint: `https://pubs.aip.org/aip/jap/article-pdf/34/9/2722/18330000/2722\_1\_online.pdf` (cit. on p. 7).

[32] T. Cover and P. Hart. “Nearest neighbor pattern classification”. In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27 (cit. on p. 54).

[33] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142 (cit. on p. 63).

[34] Alexander Denker, Maximilian Schmidt, Johannes Leuschner, and Peter Maass. “Conditional Invertible Neural Networks for Medical Imaging”. In: *Journal of Imaging* 7.11 (2021) (cit. on p. 96).

[35] Kilian Dremel. “Modellbildung des Messprozesses und Umsetzung eines modellbasierten iterativen Lösungsverfahrens der Schnittbild-Rekonstruktion für die Röntgen-Computertomographie”. doctoralthesis. Universität Würzburg, 2018 (cit. on pp. 39, 40, 95, 126, 127).

[36]Anton du Plessis, Stephan Gerhard le Roux, and Anina Guelpa. “Comparison of medical and industrial X-ray computed tomography for non-destructive testing”. In: *Case Studies in Nondestructive Testing and Evaluation* 6 (2016), pp. 17–25 (cit. on p. 7).

[37]John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *J. Mach. Learn. Res.* 12.null (July 2011), pp. 2121–2159 (cit. on p. 74).

[38]Christian Etmann, Rihuan Ke, and C. Schönlieb. “iUNets: Learnable Invertible Up- and Downsampling for Large-Scale Inverse Problems”. In: *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)* (2020), pp. 1–6 (cit. on p. 96).

[39]Muhammad Faizan, Megat F. Zuhairi, Shahrinaz Ismail, and Sara Sultan. “Applications of Clustering Techniques in Data Mining: A Comparative Study”. In: *International Journal of Advanced Computer Science and Applications* 11.12 (2020) (cit. on p. 50).

[40]Jonas Fell, Christoph Pauly, Michael Maisl, et al. “Three-dimensional imaging of microstructural evolution in SEM-based nano-CT”. In: *Tomography of Materials and Structures* 2 (2023), p. 100009 (cit. on p. 17).

[41]Zoubin Ghahramani. “Unsupervised Learning”. In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*. Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 72–112 (cit. on p. 50).

[42]Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 315–323 (cit. on p. 58).

[43]Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on pp. 45, 46, 49, 51, 53, 54, 59, 61, 63, 64, 69–76, 78, 81).

[44]J. Graetz, D. Müller, A. Balles, and C. Fella. “Lenseless X-ray nano-tomography down to 150 nm resolution: on the quantification of modulation transfer and focal spot of the lab-based ntCT system”. In: *Journal of Instrumentation* 16.01 (Jan. 2021), P01034 (cit. on p. 17).

[45]Karol Gregor and Yann LeCun. “Learning Fast Approximations of Sparse Coding.” In: *ICML*. Ed. by Johannes Fürnkranz and Thorsten Joachims. Omnipress, 2010, pp. 399–406 (cit. on pp. 83, 84).

[46]J. Hadamard. “Sur les problèmes aux dérivés partielles et leur signification physique”. In: *Princeton University Bulletin* 13 (1902), pp. 49–52 (cit. on p. 19).

[47]B. N. Hahn. “Motion Estimation and Compensation Strategies in Dynamic Computerized Tomography”. In: *Sensing and Imaging* 18.10 (2017), pp. 1–20 (cit. on pp. 23, 24).

[48]Ji He and Jianhua Ma. “Radon inversion via deep learning”. In: *Medical Imaging*. 2018 (cit. on p. 82).

[49]Ji He, Yongbo Wang, and Jianhua Ma. “Radon Inversion via Deep Learning”. In: *IEEE Transactions on Medical Imaging* 39.6 (2020), pp. 2076–2087 (cit. on pp. 82, 83).

[50]Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034 (cit. on p. 65).

[51]Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778 (cit. on pp. 65, 66, 104).

[52]Tommi Heikkila. “STEMPO – dynamic X-ray tomography phantom”. In: 2022 (cit. on p. 82).

[53]Allard Hendriksen, Dirk Schut, Willem Jan Palenstijn, et al. “Tomosipo: Fast, Flexible, and Convenient 3D Tomography for Complex Scanning Geometries in Python”. In: *Optics Express* (Oct. 2021) (cit. on pp. 124, 128).

[54]Geoffrey Hinton. *Coursera Neural Networks for Machine Learning lecture 6*. Accessed: 2025-09-24. 2018 (cit. on p. 75).

[55]Heidi Hoffman, William E. Torres, and Randy D. Ernst. “Paleoradiology: advanced CT in the evaluation of nine Egyptian mummies.” In: *Radiographics: a review publication of the Radiological Society of North America, Inc* 22 2 (2002), pp. 377–85 (cit. on p. 8).

[56]G. N. Hounsfield. “Computerized transverse axial scanning (tomography): Part 1. Description of system”. In: *British Journal of Radiology* 46.552 (Jan. 2014), pp. 1016–1022. eprint: `https://academic.oup.com/bjr/article-pdf/46/552/1016/54245397/0007-1285-46-552-1016.pdf` (cit. on p. 7).

[57]Xiang Huang, Stefan M. Wild, and Zichao Wendy Di. “Calibrating Sensing Drift in Tomographic Inversion”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 1267–1271 (cit. on p. 23).

[58]Sergey Ioffe and Christian Szegedy. “Batch normalization: accelerating deep network training by reducing internal covariate shift”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML’15. Lille, France: JMLR.org, 2015, pp. 448–456 (cit. on pp. 65, 79, 80).

[59]Yiran Jia, Noah McMichael, Pedro Mokarzel, et al. “Superiorization-inspired unrolled SART algorithm with U-Net generated perturbations for sparse-view and limited-angle CT reconstruction”. In: *Physics in Medicine & Biology* 67 (Dec. 2022) (cit. on pp. 67, 85–87, 96, 100).

[60]Kyong Hwan Jin, Michael T. McCann, Emmanuel Froustey, and Michael Unser. “Deep Convolutional Neural Network for Inverse Problems in Imaging”. In: *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4509–4522 (cit. on pp. 67, 82, 96).

[61]Christoph Heinzl Johann Kastner Bernhard Plank. “Advanced X-ray computed tomography methods: High resolution CT, phase contrast CT, quantitative CT and 4DCT”. In: *Digital Industrial Radiology and Computed Tomography (DIR 2015)* (2015) (cit. on p. 22).

[62]Leuschner Johannes, Schmidt Maximilian, Otero Baguer Daniel, Erzmann David, and Baltazar Mateus. “DIVal Library”. In: Feb. 2025 (cit. on pp. 109, 124, 128, 144).

[63]A. Katsevich, M. Silver, and A. Zamyatin. “Local Tomography and the Motion Estimation Problem”. In: *SIAM Journal on Imaging Sciences* 4.1 (2011), pp. 200–219. eprint: `https://doi.org/10.1137/100796728` (cit. on p. 23).

[64]Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]` (cit. on p. 76).

[65]Achim Klenke. *Wahrscheinlichkeitstheorie*. Jan. 2013 (cit. on pp. 47–49).

[66]Holger Kohr and Jonas Adler. “ODL (Operator Discretization Library)”. In: (Apr. 2017) (cit. on pp. 109, 124, 128).

[67]Louis Landweber. “An iteration formula for Fredholm integral equations of the first kind”. In: *American Journal of Mathematics* 73.3 (1951), pp. 615–624 (cit. on p. 26).

[68]Johannes Leuschner, Maximilian Schmidt, Poulami Somanya Ganguly, et al. “Quantitative Comparison of Deep Learning-Based Image Reconstruction Methods for Low-Dose and Sparse-Angle CT Applications”. In: *Journal of Imaging* 7.3 (2021) (cit. on pp. 96, 164, 165, 169).

[69]Johannes Leuschner, Maximilian Schmidt, Daniel Otero Baguer, and Peter Maass. “LoDoPaB-CT, a benchmark dataset for low-dose computed tomography reconstruction”. In: *Scientific Data* 8 (Apr. 2021), p. 109 (cit. on pp. 82, 120, 121, 127).

[70]Wenye Li. “Early-Stopping Regularized Least-Squares Classification”. In: Nov. 2014, pp. 278–285 (cit. on p. 81).

[71]Yuelong Li, Mohammad Tofighi, Vishal Monga, and Yonina Eldar. “An Algorithm Unrolling Approach to Deep Image Deblurring”. In: May 2019, pp. 7675–7679 (cit. on p. 85).

[72]Eugene Lin and Adam Alessio. “What are the basic concepts of temporal, contrast, and spatial resolution in cardiac CT?” In: *Journal of cardiovascular computed tomography* 3 (Nov. 2009), pp. 403–408 (cit. on p. 16).

[73]Benoit Liquet, Sarat Moka, and Yoni Nazarathy. *Mathematical Engineering of Deep Learning*. CRC Press, 2024 (cit. on p. 60).

[74]Mengnan Liu, Yu Han, Xiaoqi Xi, et al. “Thermal Drift Correction for Laboratory Nano Computed Tomography via Outlier Elimination and Feature Point Adjustment”. In: *Sensors* 21.24 (2021) (cit. on p. 23).

[75]Tianlin Liu, Anadi Chaman, David Belius, and Ivan Dokmanic. “Interpreting U-nets via task-driven multiscale dictionary learning”. In: *arXiv preprint arXiv:2011.12815* (2020) (cit. on pp. 67, 164).

[76]Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. *Towards Understanding Regularization in Batch Normalization*. 2019. arXiv: 1809.00846 [cs.LG] (cit. on p. 80).

[77]Tom Lütjen, Fabian Schönfeld, Alice Oberacker, et al. “Learning-Based Approaches for Reconstructions With Inexact Operators in nanoCT Applications”. In: *IEEE Transactions on Computational Imaging* 10 (2024), pp. 522–534 (cit. on pp. 1, 27, 43, 44, 67, 82, 83, 95, 96, 125, 126).

[78]Fabian Lutter, Philipp Stahlhut, Kilian Dremel, et al. “Combining X-ray Nano Tomography with focused ion beam serial section imaging — Application of correlative tomography to integrated circuits”. In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 500-501 (2021), pp. 10–17 (cit. on p. 16).

[79]Vishal Monga, Yuelong Li, and Yonina C. Eldar. “Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing”. In: *IEEE Signal Process. Mag.* 38.2 (2021), pp. 18–44 (cit. on p. 170).

[80]V.A. Morozov. “The error principle in the solution of operational equations by the regularization method”. In: *USSR Computational Mathematics and Mathematical Physics* 8.2 (1968), pp. 63–87 (cit. on p. 22).

[81]Dominik Müller, Jonas Graetz, Andreas Balles, et al. “Laboratory-Based Nano-Computed Tomography and Examples of Its Application in the Field of Materials Research”. In: *Crystals* 11.6 (2021) (cit. on pp. 16, 17).

[82]Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012 (cit. on p. 51).

[83]Guy Narkiss and Michael Zibulevsky. “Sequential Subspace Optimization Method for Large-Scale Unconstrained Problems”. In: (Jan. 2005) (cit. on pp. 27, 28).

[84]F. Natterer. *The Mathematics of Computerized Tomography*. Society for Industrial and Applied Mathematics, 2001. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898719284 (cit. on pp. 9, 13, 20, 21, 23, 32–34, 40, 41).

[85]Frank Natterer and Frank Wübbeling. *Mathematical Methods in Image Reconstruction*. Society for Industrial and Applied Mathematics, 2001. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718324 (cit. on pp. 8, 43).

[86]Maëlle Nodet. “Inverse problems for the environment: tools, methods and applications”. Mémoire d’habilitation à diriger des recherches. 2013 (cit. on p. 17).

[87]Fabrício Borges de Oliveira, Maurício de Campos Porath, Vitor Camargo Nardelli, Francisco Augusto Arenhart, and Gustavo Daniel Donatelli. “Characterization and Correction of Geometric Errors Induced by Thermal Drift in CT Measurements”. In: *Measurement Technology and Intelligent Instruments XI*. Vol. 613. Key Engineering Materials. Trans Tech Publications Ltd, July 2014, pp. 327–334 (cit. on p. 23).

[88]Adam Paszke, Sam Gross, Francisco Massa, et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035 (cit. on pp. 58, 123, 128).

[89]Anton du Plessis, Chris Broeckhoven, Anina Guelpa, and Stephan Gerhard le Roux. “Laboratory x-ray micro-computed tomography: a user guideline for biological samples”. In: *GigaScience* 6.6 (Apr. 2017), gix027. eprint: `https://academic.oup.com/gigascience/article-pdf/6/6/gix027/25514729/gix027.pdf` (cit. on p. 16).

[90]Boris Polyak. “Some methods of speeding up the convergence of iteration methods”. In: *Ussr Computational Mathematics and Mathematical Physics* 4 (Dec. 1964), pp. 1–17 (cit. on p. 73).

[91]Ivens Portugal, Paulo Alencar, and Donald Cowan. “The use of machine learning algorithms in recommender systems: A systematic review”. In: *Expert Systems with Applications* 97 (2018), pp. 205–227 (cit. on p. 50).

[92]J. R. Quinlan. “Induction of Decision Trees”. In: *Mach. Learn.* 1.1 (Mar. 1986), pp. 81–106 (cit. on p. 54).

[93]Johann Radon. “On the determination of functions from their integral values along certain manifolds”. In: *IEEE Transactions on Medical Imaging* 5.4 (1986), pp. 170–176 (cit. on p. 7).

[94]T. Ramos, J. S. Jørgensen, and J. W. Andreasen. “Automated angular and translational tomographic alignment and application to phase-contrast imaging”. In: *J. Opt. Soc. Am. A* 34.10 (Oct. 2017), pp. 1830–1843 (cit. on p. 23).

[95]R.Z.B.B. Ramsundar. *TensorFlow for Deep Learning*. O’Reilly Media, Incorporated, 2018 (cit. on p. 59).

[96]Andreas Rieder. *Keine Probleme mit Inversen Problemen : eine Einführung in ihre stabile Lösung*. German. Vieweg Verlag, 2003. 300 pp. (cit. on pp. 8, 10, 11).

[97]Ángela Rodríguez-Sánchez, Adam Thompson, Lars Körner, Nick Brierley, and Richard Leach. “Review of the influence of noise in X-ray computed tomography measurement uncertainty”. In: *Precision Engineering* 66 (2020), pp. 382–391 (cit. on p. 20).

[98]Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi. Cham: Springer International Publishing, 2015, pp. 234–241 (cit. on pp. 66, 83, 105, 124, 164).

[99] W. C. Röntgen. “Ueber eine neue Art von Strahlen”. In: *Annalen der Physik* 300.1 (1898), pp. 12–17. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.18983000103 (cit. on p. 7).

[100] S. Roth and M.J. Black. “Fields of Experts: a framework for learning image priors”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. 2005, 860–867 vol. 2 (cit. on p. 85).

[101] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall, 2010 (cit. on pp. 45, 49, 50, 56, 59, 60, 65).

[102] Mathias S Feinler and Bernadette N Hahn. “Learned RESESOP for solving inverse problems with inexact forward operator”. In: *Inverse Problems* 41.8 (July 2025), p. 085002 (cit. on pp. 44, 91, 97).

[103] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. “How does batch normalization help optimization?” In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 80).

[104] Gesa Sarnighausen, Anne Wald, and Alexander Meaney. “Dynamic computerized tomography using inexact models and motion estimation”. In: 2024 (cit. on pp. 43, 44).

[105] F Schöpfer and T Schuster. “Fast regularizing sequential subspace optimization in Banach spaces”. In: *Inverse Problems* 25.1 (Dec. 2008), p. 015013 (cit. on pp. 30–32, 170).

[106] F Schöpfer, Thomas Schuster, and Alfred Louis. “Metric and Bregman projections onto affine subspaces and their computation via sequential subspace optimization methods”. In: *J. Inv. Ill-Posed Problems* 15 (Jan. 2007), pp. 1–29 (cit. on pp. 28, 32, 170).

[107] Thomas Schuster, Barbara Kaltenbacher, Bernd Hofmann, and Kamil Kazimierski. “Regularization Methods in Banach Spaces”. In: *Regularization Methods in Banach Spaces* (July 2012) (cit. on pp. 30, 32, 170).

[108] Johannes Schwenck, Dominik Sonanini, Jonathan M Cotton, et al. “Advances in PET imaging of cancer”. In: *Nature Reviews Cancer* 23.7 (2023), pp. 474–490 (cit. on p. 7).

[109] Zakir Shaikh, Alcy Torres, and Takeoka. “Neuroimaging in Pediatric Epilepsy”. In: *Brain Sciences* 9 (Aug. 2019), p. 190 (cit. on p. 7).

[110] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press, 2014 (cit. on p. 52).

[111] Sigray, Inc. *TriLambdaXRM-30 Specifications*. https://www.sigray.com/trilambda-30/. [Online; accessed 5-April-2024]. 2024 (cit. on p. 17).

[112] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV] (cit. on p. 65).

[113] Christian Szegedy, Wei Liu, Yangqing Jia, et al. “Going Deeper With Convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015 (cit. on p. 65).

[114] Juan Terven, Diana-Margarita Cordova-Esparza, Julio-Alejandro Romero-González, Alfonso Ramírez-Pedraza, and Edgar Chávez Urbiola. “A comprehensive survey of loss functions and metrics in deep learning”. In: *Artificial Intelligence Review* 58 (Apr. 2025) (cit. on pp. 76, 77).

[115] Mehmet Ozan Unal, Metin Ertas, and Isa Yildirim. “An unsupervised reconstruction method for low-dose CT using deep generative regularization prior”. In: *Biomedical Signal Processing and Control* 75 (2022), p. 103598 (cit. on p. 82).

[116] Wim van Aarle, Willem Jan Palenstijn, Jan De Beenhouwer, et al. “The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography”. In: *Ultramicroscopy* 157 (2015), pp. 35–47 (cit. on pp. 40, 119, 126–128).

[117] Vladimir N Vapnik and Alexey Ya Chervonenkis. “On a class of pattern-recognition learning algorithms”. In: *Automation and Remote Control* 25.6 (1964), pp. 838–845 (cit. on p. 54).

[118] L. Vásárhelyi, Z. Kónya, Á. Kukovecz, and R. Vajtai. “Microcomputed tomography–based characterization of advanced materials: a review”. In: *Materials Today Advances* 8 (2020), p. 100084 (cit. on p. 16).

[119] Anne Wald. “Sequential subspace optimization for nonlinear inverse problems with an application in terahertz tomography”. PhD thesis. Universität des Saarlandes., 2017 (cit. on pp. 27, 30–32).

[120] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, et al. “scikit-image: image processing in Python”. In: *PeerJ* 2 (June 2014), e453 (cit. on pp. 16, 155).

[121] Wei Wang, Xiang-Gen Xia, Chuanjiang He, et al. *A deep network for sinogram and CT image reconstruction*. 2020. arXiv: `2001.07150 [eess.IV]` (cit. on p. 83).

[122] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612 (cit. on p. 130).

[123] Zhou Wang and Alan C. Bovik. “Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures”. In: *IEEE Signal Processing Magazine* 26.1 (2009), pp. 98–117 (cit. on pp. 77, 129).

[124] Rong Xu, Yang Yang, Fei Yin, et al. “Heterogeneous damage in Li-ion batteries: Experimental analysis and theoretical modeling”. In: *Journal of the Mechanics and Physics of Solids* 129 (2019), pp. 160–183 (cit. on p. 16).

[125] Shuo Xu, Jintao Fu, Yuewen Sun, Peng Cong, and Xincheng Xiang. “Deep Radon Prior: A fully unsupervised framework for sparse-view CT reconstruction”. In: *Computers in Biology and Medicine* 189 (2025), p. 109853 (cit. on p. 82).

[126] Qingsong Yang, Pingkun Yan, Yanbo Zhang, et al. “Low-Dose CT Image Denoising Using a Generative Adversarial Network With Wasserstein Distance and Perceptual Loss”. In: *IEEE Transactions on Medical Imaging* 37 (2017), pp. 1348–1357 (cit. on p. 82).

[127] Yang Yang, Rong Xu, Kai Zhang, et al. “Quantification of Heterogeneous Degradation in Li-Ion Batteries”. In: *Advanced Energy Materials* 9.25 (2019), p. 1900674. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/aenm.201900674` (cit. on p. 16).

[128] Bo Zhu, Jeremiah Zhe Liu, Bruce R. Rosen, and Matthew S. Rosen. “Image reconstruction by domain-transform manifold learning”. In: *Nature* 555 (2017), pp. 487–492 (cit. on p. 82).

List of Figures

List of Tables

List of Algorithms

Colophon

This thesis was typeset with LaTeX 2$_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.