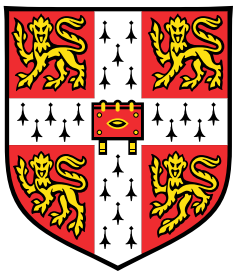
Learned Primal-Dual reconstruction for Medical Imaging



Larry Wang

Department of Physics
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Downing College

June 2024

I would like to dedicate this thesis to my family and friends, for their unwaivering support.

Declaration

I, Dai (Larry) Wang, hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text, Acknowledgements, and the Appendix.

This dissertation contains approximately 6,997 words, excluding declarations, bibliography and Generative AI acknowledgement, but including tables and figure captions and Appendices.

Larry Wang
June 2024

Acknowledgements

First and foremost, I would like to express sincere gratitude to my supervisor, Dr. Ander Biguri, for always providing prompt support throughout the completion of this dissertation. I would also like to thank our course director and course admin, James and Sri, for their help throughout the whole degree, both inside and outside of classrooms.

I would like to thank my coursemates and friends, for all the insightful conversations and great times we had throughout the year. I will always look back at our late night studies and tennis rallies with great fondness.

I would also like to thank Downing College Music Society and Downing College Chapel Choir. It had been an honour being a member of both this year, and I can't wait to see what the near future will bring.

Last but not least, I am forever grateful to my family for their continuous support throughout my education.

Abstract

In computed tomography, reconstruction of images often involves minimising objective functions which are convex and non-smooth, such as Total Variation (TV) regularisation [17]. Optimising such functions may be challenging, as traditional techniques lack convergence guarantees in face of non-smoothness. Nevertheless, the rise of proximal algorithms provides means of iteratively optimising these functions with convergence guarantees. Moreover, recent research showed that replacing vanilla updates in optimisation algorithms (such as Gradient Descent) with learned neural networks can result in impressive performances, often beating out vanilla update schemes across multiple metrics.

To this end, the Learned Primal Dual (LPD) [1] algorithm is proposed; it combines these two aforementioned ideas by replacing the fixed update schemes in proximal algorithms with learned neural networks. Its performance greatly surpassed non-machine learning methods such as Filtered Backprojection (FBP) and Total Variation (TV), and remained competitive compared to other state-of-the-art models. As extension, we also discussed and evaluated a custom variant of the LPD, TV-LPD, whose performance is marginally better than LPD at the expense of having more parameters.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

In biomedical sciences, medical imaging stands as one of its most vital fields of research. It allows practitioners to create detailed scans of the internals of a patient, which may otherwise be difficult to obtain. These images can then be used to identify abnormalities inside the patient's body, thus leading to more effective diagnosis and treatments.

Today, Computed Tomography (CT) remains as one of the most used medical imaging techniques in modern clinical settings, despite its early origin in the 1970s. During CT, X-rays are emitted in straight lines and their intensities get partially attenuated by different mediums inside the body, and the strength of attenuation is dependent on the particular medium traversed. (For example, bones are much more absorbent than fat.) Detectors on the other side of the body then pick up varying levels of remaining intensities, and these intensity values are used to infer the internal structures of the patient.

Inevitably, the patient is exposed to X-rays during the scanning process, which may be undesirable, as they are known to have cancer inducing effects. However, in order to obtain high quality reconstructions, relatively large amount of radiation doses is needed. Classical reconstruction methods such as filtered backprojection (FBP) and total variation (TV) [17] either fail to balance radiation dose with quality of reconstruction or are computationally inefficient, making finding a good balance between the two ends a challenge for researchers.

Thus, modern CT medical imaging research revolves around producing efficient high quality reconstructions whilst maintaining low clinical doses. The Learned Primal Dual (LPD) [1] algorithm proposed by Adler et al. is one of the most influential recent works that contributes towards this end. The paper proposes a novel neural network architecture consisting of

convolution and residual blocks that leverages prior information to make more accurate reconstructions.

1.1 Overview

In this thesis, the contents are split into 5 chapters; following this introduction, the remaining chapters are:

1. **Chapter 2**: We outline the underlying mathematical and physical foundations behind CT imaging, including Lambert-Beer Law, Radon Transform (forward projection), backprojection and filtered backprojection,

2. **Chapter 3**: This section evaluates the pitfalls of performances of standard CT image reconstruction methods (filtered backprojection and total variation) and introduces proximal methods, which leads to the main algorithm proposed in the paper, Learned Primal Dual. We also outline other algorithms mentioned, as well as one of our custom algorithms.

3. **Chapter 4**: In this chapter, we outline the training procedure and analyse findings by comparing the achieved results by the different models. Specifically, the results section was split into two parts; the first part presents the reproduction results, and the second part presents extension results.

4. **Chapter 5**: The final chapter begins with an overall summary of all completed work and findings, followed by a critical evaluation of LPD's reproducibility and limitations. Potential areas for future extensions are touched upon as concluding remarks.

Chapter 2

Background

This chapter begins by discussing the physics behind X-rays and CT imaging, followed by introductions of relevant mathematical transforms. Particularly, in section 2.1, the (hypothetical) physical equipment setup is discussed, as well as the relationship between the detected intensities and initial intensities. Then, in section 2.2, the mathematical transforms present in the forward process (Radon Transform) and the inverse process (backprojection and filtered backprojection) were formally introduced and derived. Lastly, an overview of inverse problems and its connections to CT reconstruction is given in section 2.3.

2.1 Initial intensity and detected intensity

In CT imaging, X-rays are used to infer the internal structures of a patient's body. At detectors, X-ray intensities are measured by counting number of protons detected. When photons pass through materials, they either get absorbed or scattered, causing the number of photons detected at the detectors to be less than the original intensity. Specifically, supposing that there is a homogeneous material of length ℓ, the Lambert-Beer law [8] states that the detected intensity is related to the original intensity I_0 as follows:

$$I = I_0 \times e^{-\mu_0 \ell} \tag{2.1}$$

where μ_0 is the **attenuation coefficient**, a measurement for strength of attenuation of materials. For heterogeneous materials, equation 2.1 can be generalised as:

$$I = I_0 \times e^{-\int_L \mu(\ell)\, d\ell} \tag{2.2}$$

where $\mu(\ell)$ is the unknown function of attenuation along the trajectory of an X-ray L at different distances ℓ. The aim is to infer the function $\mu(\ell)$ for all positions (x,y), as we can then gain detailed information on internal structures by matching measured attenuation values to reference values. Usually, multiple rays at varying angles and positions are required to fully infer values of μ at all positions.

For the baseline reproduction part of the report, to approximately align with the geometry employed in the original paper [1], it's assumed that at each angle of projection, there are 543 evenly spaced X-ray sources emitting rays parallelly with initial intensity $I_0 = 4096$ photons, and directions of projections are taken to be 1000 evenly spaced angles in the range $[0, 180)$ degrees. However, in the extension section 4.7, some of these conditions are relaxed/altered in order to compare the robustness of different models under adversarial conditions.

2.2 Mathematical formulations of transforms

2.2.1 Forward Transform (Radon Transform)

In equation 2.2, the term $\int_L \mu(\ell)d\ell$ requires calculating the integral along a parameterised line L. In 2D, any straight line can be uniquely identified by s and θ, where s is its signed perpendicular distance from the origin and θ is the angle of which the said perpendicular line makes to the x-axis, as demonstrated in figure 2.1.

Denoting this line as $L_{\theta,s}$, its parameterisation is given by [8]:

$$L_{\theta,s} = \{\ell \in [-\infty,\infty] \mid s \times \cos\theta - \ell \times \sin\theta,\ s \times \sin\theta + \ell \times \cos\theta\} \tag{2.3}$$

Thus, for any straight line $L_{\theta,s}$, the line integral in the exponent of equation 2.2 can be rewritten as:

$$g(\theta,s) = \int_{-\infty}^{\infty} \mu(s \times \cos\theta - \ell \times \sin\theta,\ s \times \sin\theta + \ell \times \cos\theta)\ d\ell \tag{2.4}$$

The Radon transform $\mathscr{R}$ is then defined as the integral transform of the function μ; for any values of θ and s, $\mathscr{R}[\mu](\theta,s)$ is denoted as $g(\theta,s) = \int_{-\infty}^{\infty} \mu(s \times \cos\theta - \ell \times \sin\theta,\ s \times \sin\theta + \ell \times \cos\theta)\ d\ell$. Usually, the parametrised attenuation function μ is assumed to be 0 outside of some radius of R, as the distance between the X-ray sources and detectors are finite. The integration limits from equation 2.4 can then be relaxed accordingly to include only the non-zero interval of μ.

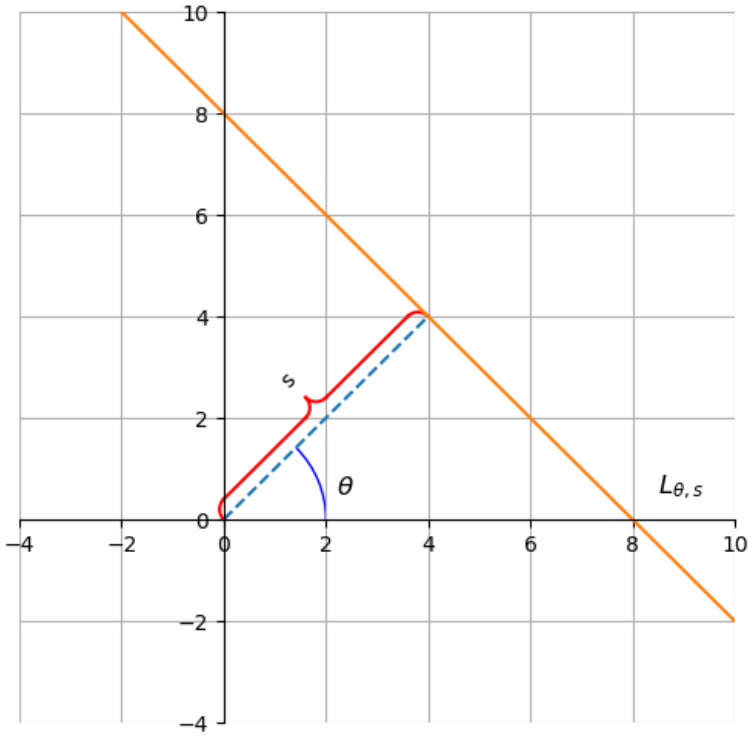


Fig. 2.1 Demonstration of the parameterised line $L_{\theta,s}$ in Radon Transform. Plot is generated with aid from `curlyBrace` repository [7].

2.2.2 Sinogram

The Sinogram is an 2D array consisting of the outputs from $\mathscr{R}[\mu]$ for different values of θ and s. When visualised, it's a 2D heatmap with θ on the horizontal axis and s on the vertical axis. The different levels of shades indicate different values of $g(\theta,s)$. Figure 2.2 illustrates an example of a Sinogram.

2.2.3 Backprojection

The adjoint operator $\mathscr{R}^*$ of Radon Transform is known as the backprojection, and it's defined as [8]:

$$\mathscr{R}^*[g](x,y) = \int_0^{2\pi} g(\theta, x \times \cos\theta, y \times \sin\theta) d\theta \tag{2.5}$$

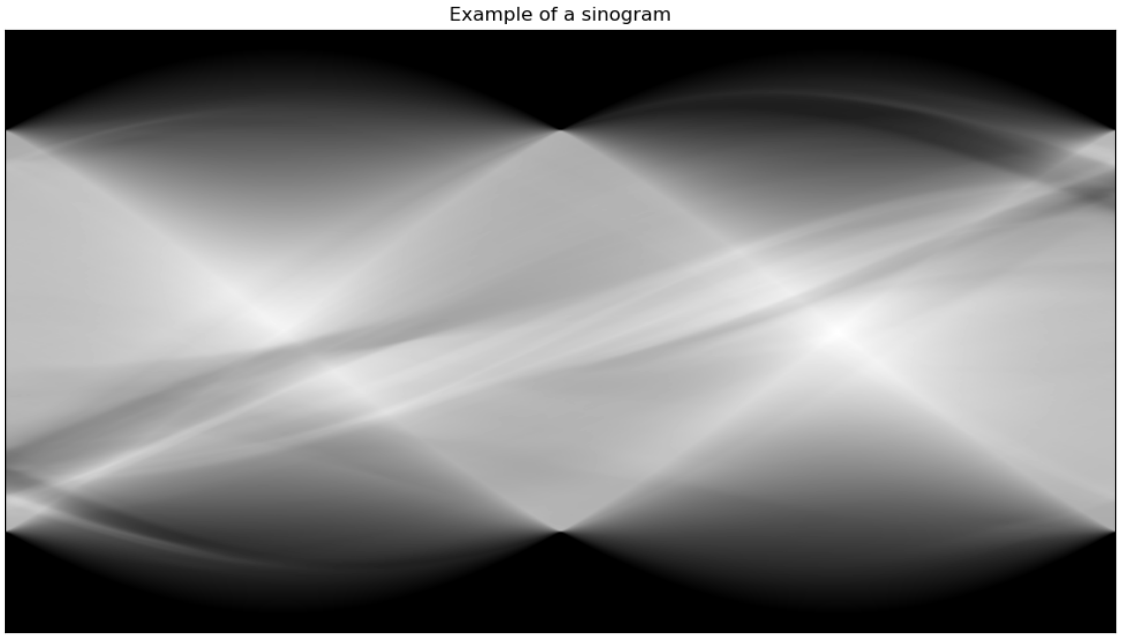


Fig. 2.2 Example of a sinogram

It can be thought of as summing over all the contributions to the forward projection $\mathscr{R}$ at different angles θ for a fixed point (x,y), and its form somewhat resembles that of the forward operation. Nevertheless, backprojection doesn't yield the analytic inverse of the Radon transform; it produces a blurred version of the original image like the one in figure 2.3.

2.2.4 Filtered Backprojection

Instead, the analytic inverse for Radon transform is given by **filtered backprojection**, whose formula is [8]:

$$\mu(x,y) = \frac{1}{4\pi}\int_0^{2\pi} \frac{1}{\sqrt{2\pi}}\left[\int_{-\infty}^{\infty} \hat{g}(\theta,\omega)|\omega| e^{2\pi i \omega \mathbf{x}\phi} d\omega\right] d\theta \tag{2.6}$$

where $\hat{g}(\theta,\omega)$ denotes the Fourier Transform of $g(\theta,s)$ in θ. Exact derivation of this can be found in Appendix A. The term *filtered* comes from the fact that a filtering operation is first done in the frequency domain (multiplication by the amplitude of frequency, $|\omega|$), before backprojecting.

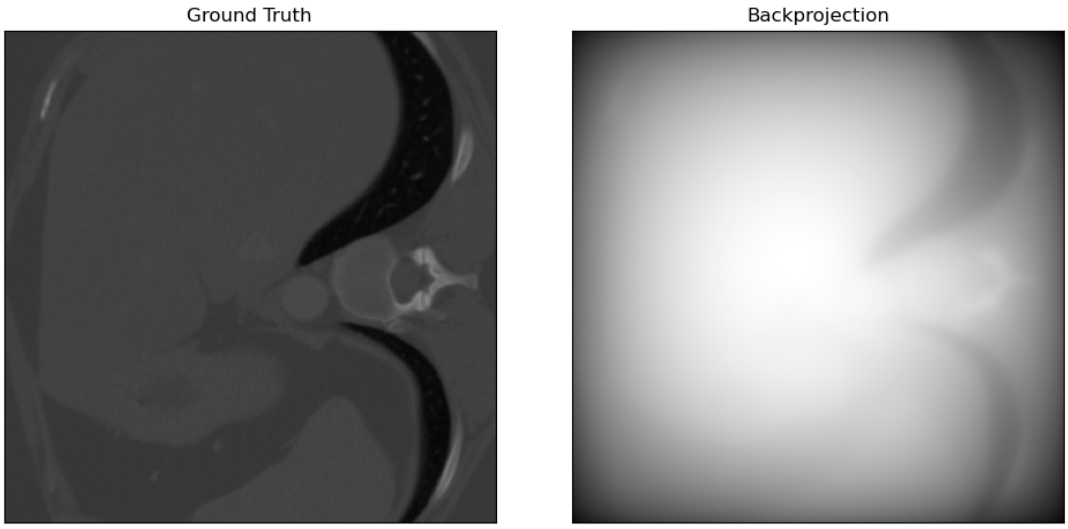


Fig. 2.3 Comparison of an image and its backprojection

2.3 Inverse problems

In general, inverse problems refer to the family of problems where the goal is to make inference on an unknown quantity $\mu \in \Re^d$, based on a series of indirect observations p; mathematically, p is linked to μ via the matrix multiplication $A\mu = p$, where A describes the physical laws that underpins the relationship between μ and p. In the context of CT imaging, 'μ' refers to the (unobserved) linear attenuation function of the internals of a patient's body, A is the (discretised) forward Radon transform and p is the Sinogram. The discretisation of the forward transform and construction of A will be further discussed in section 3.1.

Unfortunately, inverse problems are often ill-posed, meaning that they violate at least one of the well-posedness conditions:

1. **Existence**: There exists a solution.
2. **Uniqueness**: There exists only one unique solution.
3. **Stability**: The solution changes continuously with input.

The latter condition could particularly cause problems. In practice, as imperfect observations are often obtained due to measurement inaccuracies, the obtained quantity $\tilde{p}$ are perturbed away from the ground truth quantity p by a deviation δp:

$$\tilde{p} = A\mu + \delta p \tag{2.7}$$

Instability would then cause the compromised solution to deviate substantially from true solution, which is undesirable. Therefore, much of current research revolves around finding solutions to inverse problems that preserve fidelity in face of ill-posedness. With the rise of machine learning methods and abundance of readily available clinical data, large scale data-driven reconstruction methods have emerged. These methods usually fall into the two main schools of thought:

1. Post processing machine learning methods: These methods typically involve first computing an approximate solution using a 'pseudoinverse' $A^{\dagger}$ such that $A^{\dagger}\tilde{p} \approx \mu$, then learning a network f_{θ} to further denoise $A^{\dagger}\tilde{p}$. One of the models which we have covered in this thesis (FBPConvNet [11]) falls into this category.

2. Learned iterative machine learning methods: these methods learn the whole reconstruction process whilst constraining the space of solutions by incorporating the forward operators A and its adjoint A^{T}, which eliminates solutions that are infidel to the physical geometry. LPD falls into this category.

In chapters 3 and 4, we delve deep into the mathematical theory behind the setup of the algorithm (and some of its variations) and its performance; we also briefly discuss and compare the performance of post processing machine learning methods.

Chapter 3

Methods of reconstruction

In this section, we start by discussing classical, non-machine learning based reconstruction methods, their implementations and limitations. This naturally leads to the introduction of machine learning based methods, whose aims are to address the shortcomings of traditional methods. In sections 3.1 and 3.2, we discuss how the continuous transforms from chapter 2 are discretised, leading to a derivation of an estimate of uncertainty in reconstructions from discretised filtered-backprojections. In section 3.3, variational reconstructions methods, in particular, Total Variation (TV), is discussed. Proximal methods are explored extensively, as they prove crucial in optimising non-smooth convex functions, such as TV's objective function. This naturally leads into the introduction of machine learning models in sections 3.4 and 3.5, which combine proximal methods with recent advances in deep learning.

3.1 Discretisation of the transformations

Since all transforms discussed in Chapter 2 are defined over a continuous space, which is unrealistic, as we only have a finite amount of measurements, discretisation measures are therefore required to approximate these continuous transforms with discrete sums.

Firstly, the image itself is discretised into pixels, using some sampling frequency (Δi, Δj). Any pixel is determined by 4 components: $p_{left}, p_{right}, p_{top}, p_{bottom}$. They satisfy: $p_{right} - p_{left} = \Delta i$, $p_{top} - p_{bottom} = \Delta j$, and the value of the image p is assumed to be constant inside each pixel $\{[p_{left}, p_{right}], [p_{top}, p_{bottom}]\}$.

For ease of notation, let μ temporarily denote the flattened, discretised original $2D$ image μ. The Radon Transform along X-ray i is approximated as the following sum:

$$L_i \approx \sum_{k=1}^{N\times M} A_{ik}\mu_k \tag{3.1}$$

where A_{ik} corresponds to length of X-ray i present in the k-th pixel and μ_k represents the (unknown) approximated constant attenuation value at pixel k, as demonstrated in figure 3.1:

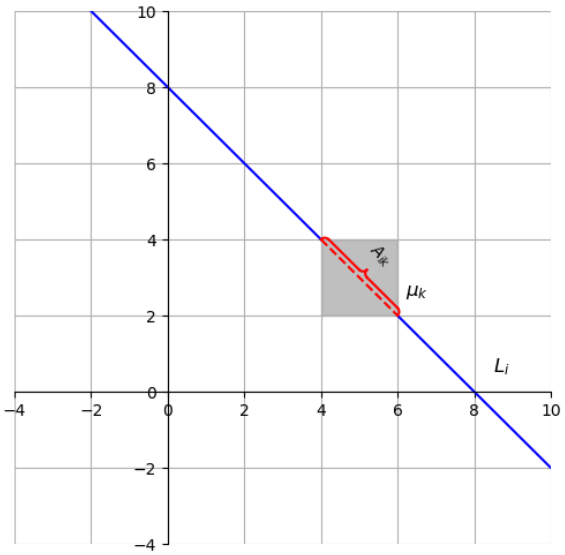


Fig. 3.1 Illustration of A_{ik}, μ_k under discretisation

Denoting A as the matrix where its i-th row is the set of values $\{A_{ik}\}_{k=1,2,\cdots,N\times M}$, the set of approximated Radon transforms values along rays of discretely sampled frequencies is given by the flattened 1D vector $A\mu = p$. Reshaping p into an 2D $N \times M$ array and visualising it would return an approximated, discretely sampled version of a continuous Sinogram. Analogously, the backprojection operation is approximated as the linear transformation A^T.

The discretisation of filtered backprojection requires more thought. Since there are two integrals involved, each requires separate discretisation:

$$\mu(x,y) = \underbrace{\frac{1}{4\pi}\int_0^{2\pi} \frac{1}{\sqrt{2\pi}} \left[\underbrace{\int_{-\infty}^{\infty} \hat{g}(\theta,\omega)|\omega| e^{2\pi i \omega \mathbf{x}\phi} d\omega}_{\text{Inner Integral}} \right] d\theta}_{\text{Outer Integral}}$$

Temporarily setting the multiplicative constants aside and denoting the inner integral as $I(x\cos\theta + y\sin\theta) = \int_{-\infty}^{\infty} \hat{g}(\theta,\omega)|\omega|e^{2\pi i\omega(x\cos\theta+y\sin\theta)\phi}d\omega$, the outer integral is approximated using Monte Carlo integration [12]:

$$\int_0^{2\pi} I(x\cos\theta + y\sin\theta)d\theta \approx \frac{2\pi}{N_{angles}} \sum_{i=1}^{N_{angles}} I(x\cos\theta_i + y\sin\theta_i) \tag{3.2}$$

where N_{angles} denotes the number of projection angles. Now, to compute the inner integral, by leveraging Convolution Theorem, the filtering step in Fourier space during filtered backprojection can be rewritten as a convolution in Sinogram space:

$$\int_{-\infty}^{\infty} \hat{g}(\theta,\omega)|\omega|e^{2\pi i\omega\mathbf{x}\phi}d\omega = \int_{-\infty}^{\infty} g(\theta,s)h(x\cos\theta + y\sin\theta - s)ds \tag{3.3}$$

where $h(\cdot)$ is the inverse Fourier Transform of the filtering function $|\omega|$ and is known. Then, the convolution integral 3.3 can also approximated as a discrete sum using Monte Carlo Integration:

$$\int_{-\infty}^{\infty} g(\theta,s)h(x\cos\theta + y\sin\theta - s)ds \approx \sum_{i=1}^{N_{proj}} g(\theta,s_i)h(x\cos\theta + y\sin\theta - s_i) \tag{3.4}$$

where N_{proj} represents the number of projections at each angle. Thus, combining equations 3.2 and 3.4 and incorporating the multiplicative constants, the eventual discrete approximation to filtered backprojection is given by:

$$\mu(x,y) = \frac{1}{4\pi}\int_0^{2\pi} \frac{1}{\sqrt{2\pi}}\left[\int_{-\infty}^{\infty} \hat{g}(\theta,\omega)|\omega|e^{2\pi i\omega\mathbf{x}\phi}d\omega\right]d\theta \tag{3.5}$$

$$\approx \frac{1}{4\pi}\frac{1}{\sqrt{2\pi}}\frac{2\pi}{N_{angles}} \sum_{i=1}^{N_{angles}}\sum_{j=1}^{N_{proj}} g(\theta_i,s_j)h(x\cos\theta_i + y\sin\theta_i - s_j) \tag{3.6}$$

$$= \frac{1}{2\sqrt{2\pi}\times N_{angles}} \sum_{i=1}^{N_{angles}}\sum_{j=1}^{N_{proj}} g(\theta_i,s_j)h(x\cos\theta_i + y\sin\theta_i - s_j) \tag{3.7}$$

3.2 Why Not Filtered Backprojection?

In practice, filtered backprojection's performance is often suboptimal. Despite it being the inverse to Radon Transform, its performance falters greatly in presence of noise; reminding

ourselves of its form in equation 2.6, the filtering term in frequency domain is equal to $|\omega|$:

$$\mu(x,y) = \frac{1}{4\pi}\int_0^{2\pi}\frac{1}{\sqrt{2\pi}}\left[\int_{-\infty}^{\infty}\hat{g}(\theta,\omega)\underbrace{|\omega|}_{\text{filtering term}}e^{2\pi i\omega\mathbf{x}\phi}d\omega\right]d\theta \tag{3.8}$$

This term has the undesirable property of magnifying components of the image with high frequency values, as seen in figure 3.2, and higher frequency components are often associated with noise. Furthermore, an analytic estimate for uncertainties in pixel values under discretised FBP reconstruction can be derived. We detail the derivation steps below (adapted from Chapter 5 of [12]).

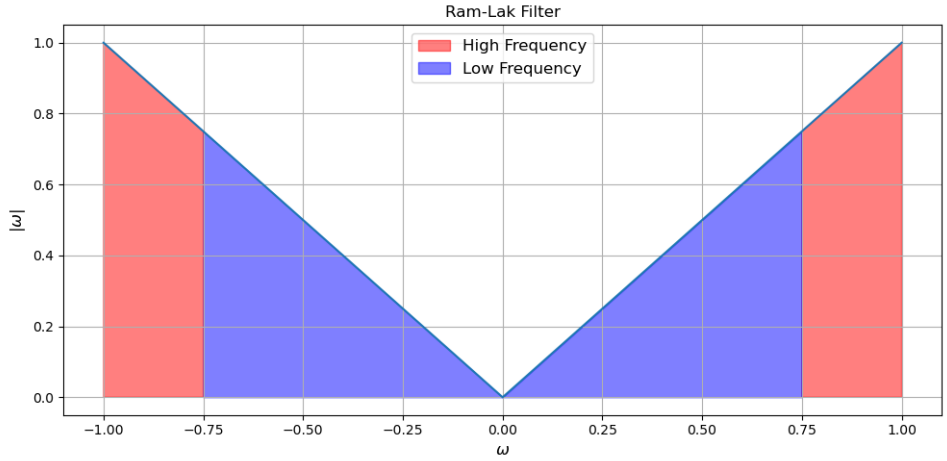


Fig. 3.2 Visualisation of Ram-Lak filter

Note that the following nomenclature are used during derivation:

- I_0: initial intensity at X-ray emitter.
- L_{θ_i,s_j}: the parameterised line corresponding discretised ray transform at angle θ_i and distance s_j.
- N_{θ_i,s_j}: the expected number of received photons for ray L_{θ_i,s_j} $\left(I_0 \times e^{-\int_{L_{\theta_i,s_j}}\mu(\ell)\,d\ell}\right)$.
- $\hat{N}_{\theta_i,s_j}$: the actual measured number of received photons for ray L_{θ_i,s_j}.
- g_{θ_i,s_j}: the expected value for $\int_{L_{\theta_i,s_j}}\mu(\ell)\,d\ell$.

- $\hat{g}_{\theta_i,s_j}$: the actual measured value for $\int_{L_{\theta_i,s_j}} \mu(\ell)\, d\ell$. It can alternatively be expressed in terms of $\hat{N}_{\theta_i,s_j}$ and I_0: $\log \frac{I_0}{N_{\theta_i,s_j}}$

At X-ray detectors, intensities are measured by counting number of detected photons. This is a **counting process**, whose uncertainties are modelled using Poisson distribution. Hence, the distribution of observed intensity $\hat{N}_{\theta_i,s_j}$ can be expressed as: $\hat{N}_{\theta_i,s_j} \sim Poisson(N_{\theta_i,s_j})$. This property can be exploited simulate different levels of noise, and the noise simulation procedure is covered more extensively in section 4.2.

Returning to problem at hand, the variance of $\hat{g}(\theta_i, s_j)$, can be derived using Delta method:

$$
\begin{aligned}
Var(\hat{g}(\theta_i,s_j)) &= Var\left(\log \frac{I_0}{\hat{N}_{\theta_i,s_j}}\right) && (3.9)\\
&= Var(\log(I_0) - \log \hat{N}_{\theta_i,s_j}) && (3.10)\\
&= Var(\log(\hat{N}_{\theta_i,s_j})) && (3.11)\\
&\approx \left[\frac{d}{dx}\log(x)\right]^2\Big|_{x=E(\hat{N}_{\theta_i,s_j})} \times Var(\hat{N}_{\theta_i,s_j}) && (3.12)\\
&= \frac{1}{N_{\theta_i,s_j}} && (3.13)
\end{aligned}
$$

The variance of the FBP reconstruction $\hat{\mu}(x,y)$ can thus be approximated as:

$$
\begin{aligned}
Var(\hat{\mu}(x,y)) &= Var\left(\frac{1}{2\sqrt{2}N_{angles}} \sum_{i=1}^{N_{angles}} \sum_{j=1}^{N_{proj}} \hat{g}(\theta_i,s_j) h(x\cos\theta_i + y\sin\theta_i - s_j)\right) && (3.14)\\
&\propto \sum_{i=1}^{N_{angles}} \sum_{j=1}^{N_{proj}} Var(\hat{g}(\theta_i,s_j)) h^2(x\cos\theta_i + y\sin\theta_i - s_j) && (3.15)\\
&\approx \sum_{i=1}^{N_{angles}} \sum_{j=1}^{N_{proj}} \frac{1}{N_{\theta_i,s_j}} h^2(x\cos\theta_i + y\sin\theta_i - s_j) && (3.16)\\
&= \sum_{i=1}^{N_{angles}} \sum_{j=1}^{N_{proj}} \frac{1}{I_0 \times e^{-g(\theta_i,s_j)}} h^2(x\cos\theta_i + y\sin\theta_i - s_j) && (3.17)\\
&\propto \mathcal{O}\left(\frac{1}{I_0}\right) && (3.18)
\end{aligned}
$$

Evidently, with lower value of I_0, the variance increases, and the rate of deterioration increases as I_0 lowers. Thus, we conclude that FBP is not stable when noise is present, consequently violating one of the well-posedness conditions stated in section 2.3.

3.3 Variational Reconstruction

Following the pitfalls of Filtered Backprojection, we seek other alternative methods for reconstruction. Going back a few steps and reminding ourselves that the forward transform was discretised into the linear system $A\mu = p$, an alternative approach of recovering μ would be solving this system. Unfortunately, standard linear algebra techniques such as Gaussian elimination would render useless here, as matrix A is often underdetermined and sparse. Instead, these systems can be solved iteratively using optimisation techniques, typically involving minimising an objective function of the following form [3]:

$$\arg\min_{\mu} \underbrace{\|A\mu - p\|_2^2}_{\text{data fidelity term}} + \underbrace{\lambda R(\mu)}_{\text{Regularising term}} \tag{3.19}$$

Of the various regularisation functions that have emerged, Total Variation (TV) is amongst one of the most successful. Its objective function is [17]:

$$D(\mu) = \|A\mu - p\|_2^2 + \lambda \|\nabla \mu\|_1 \tag{3.20}$$

Its regularisation term $\|\nabla \mu\|_1 = \sum_i |\mu_{i+1,j} - \mu_{i,j}| + \sum_j |\mu_{i,j+1} - \mu_{i,j}|$ is the 1-norm of the image gradient. This term encourages sparsity in the image gradient, thus promoting piece-wise smoothness, much alike edges in images.

However, $\lambda \|\nabla \mu\|_1$ has the significant caveat of non-smoothness, which renders traditional optimisation techniques such as regular gradient descent useless, as smoothness is often prequiste for convergence guarantees. To this end, **Chambolle Pock Algorithm** [4] is proposed. Instead of only minimising the original objective function 3.20 (the **primal** problem), an additional separate objective function is simultaneously maximised (the **dual** problem). The two optimisation problems are formulated as below:

$$\min_{\mu} \; F(A\mu) + G(\mu) \qquad \text{(Primal Problem)} \tag{3.21}$$

$$\max_{\nu} \; -F^*(\nu) - G^*(A^T \nu) \qquad \text{(Dual Problem)} \tag{3.22}$$

where F and G are convex (not necessarily smooth or differentiable), A is a linear transform, F^* and G^* denote the respective **convex conjugates** of F and G. The convex conjugate of a function F is defined as [19]:

$$F^*(\nu) = \sup_{\mu \in dom\, \mu} \{\langle \nu, \mu \rangle - F(\mu)\} \tag{3.23}$$

The procedure used for optimising 3.21 is detailed below [4]:

Algorithm 1 Chambolle-Pock Algorithm

$L \leftarrow \|A\|_2$; $\sigma\tau < \frac{1}{L^2}$; $\theta \leftarrow 1$; $n = 0$
$\mu_0 \leftarrow \mathbf{0}$, $\nu_0 \leftarrow \mathbf{0}$ ▷ Initialise the starting primal and dual values
$\bar{\mu}_0 \leftarrow \mu_0$
for $n = 1, 2, 3 \cdots$ **do**
  $\nu_{n+1} \leftarrow prox_\sigma[F^*](\nu_n + \sigma A \bar{\mu}_n)$ ▷ Update the Dual variable
  $\mu_{n+1} \leftarrow prox_\tau[G](\mu_n - \tau A^T \nu_{n+1})$ ▷ Update the Primal Variable
  $\bar{\mu}_{n+1} = \mu_{n+1} + \theta(\mu_{n+1} - \mu_n)$
end for
Return $\bar{\mu}_N$

In algorithm 1, $\|A\|_2$ is the matrix 2-norm for A, which is defined as:

$$\|A\|_2 = \max\{\|A\mu\|_2 \text{ subject to } \|\mu\| = 1\} \tag{3.24}$$

$prox_\sigma[F]$ is the proximal operator with respect to convex function F and stepsize σ, defined as: [4]:

$$prox_\sigma[F](\mu) = \arg\min_{\mu'} \left\{ F(\mu') + \frac{\|\mu - \mu'\|_2^2}{2\sigma} \right\} \tag{3.25}$$

To place the objective function of Total Variation (3.20) into the setting of the primal problem (3.21), by noticing that the div operator ∇ is also a linear transform, we define the **augmented linear operator** K, where $K = [A, \nabla]^T$. Under such convention, F can then be seen as a separable sum of two terms from equation 3.20:

$$F(K\mu) = F(A\mu, \nabla\mu) = \frac{1}{2}\|A\mu - p\|_2^2 + \lambda\|\nabla\mu\|_1 \tag{3.26}$$

and $G = 0$. The convex conjugate of F is given by:

$$F^*(\mu, v) = \frac{1}{2}\|\mu\|_2^2 + \langle \mu, p \rangle \qquad \text{subject to } |v_i| < \lambda \; \forall\, i \tag{3.27}$$

The proximals for F^* and G can be derived analytically. Full derivations can be found in the paper [19]:

$$prox_{\sigma}[F^*](\nu, z) = \left(\frac{\nu - \sigma p}{1+\sigma}, \frac{\lambda z}{\max(\lambda \mathbf{1}, |z|)} \right) \qquad \text{(proximal operator for } F^*\text{)} \tag{3.28}$$

$$prox_{\tau}[G](\mu) = \mu \qquad \text{(proximal operator for } G\text{)} \tag{3.29}$$

Thus, plugging these expressions into algorithm 1, the Chambolle-Pock algorithm for total variation regularisation is as follows:

Algorithm 2 Chambolle-Pock Algorithm for Total Variation regularisation

$L \leftarrow \|[A, \nabla]\|_2$; $\sigma\tau \leq \frac{1}{L^2}$; $\theta \leftarrow 1$; $n = 0$
$\mu_0 \leftarrow \mathbf{0}$, $\nu_0 \leftarrow \mathbf{0}$, $z_0 \leftarrow \mathbf{0}$ ▷ Initialise the starting primal and dual values
$\bar{\mu}_0 \leftarrow \mu_0$
for $n = 1, 2, 3 \cdots$ **do**
  $\nu_{n+1} \leftarrow (\nu_n + \sigma(A\bar{\mu}_n - p))/(1+\sigma)$ ▷ Update the first component of the Dual variable
  $z_{n+1} \leftarrow \lambda(z_n + \sigma\nabla\bar{\mu}_n)/(max(\lambda\mathbf{1}, |z_n + \sigma\nabla\bar{\mu}_n|))$ ▷ Update the second component of the Dual variable
  $\mu_{n+1} \leftarrow \mu_n - \tau A^T \nu_{n+1} - \tau\nabla^T z_{n+1}$ ▷ Update the component of the Primal Variable
  $\bar{\mu}_{n+1} = \mu_{n+1} + \theta(\mu_{n+1} - \mu_n)$
end for
Return $\bar{\mu}_N$

TV was used as one of the other 'traditional' benchmarks against the proposed machine learning models. `tomosipo` [9] contains numerical implementation of algorithm 2.

3.4 Inspiration for LPD

In Learned Primal Dual, the author took inspiration from algorithm 1, as they mimicked its iterative, alternating structure to optimise primal and dual variables, such as one in the form of equation 3.20. However, instead of analytically computing the proximals, they are approximated with learned neural networks instead, which may be more advantageous as some objective functions don't have analytic expressions for the proximals of F^* and G. Additionally, previous works with similar schemes [2] (replacing vanilla update schemes with learned neural networks) had already demonstrated success, thus providing further motivation for us to apply a similar approach here.

Under the learned scheme, algorithm 1 becomes:

Algorithm 3 Learned Chambolle-Pock Algorithm

$L \leftarrow \|A\|_2$; $\sigma\tau < \frac{1}{L^2}$; $\theta \leftarrow 1$; $n = 0$
$\mu_0 \leftarrow \mathbf{0}$, $\nu_0 \leftarrow \mathbf{0}$ ▷ Initialise the starting primal and dual values
$\bar{\mu}_0 \leftarrow \mu_0$
for $n = 1,2,3\cdots$ **do**
  $\nu_{n+1} \leftarrow \Gamma_{\theta^{dual}}(\nu_n + \sigma A\bar{\mu}_n, p)$ ▷ Update the Dual variable
  $\mu_{n+1} \leftarrow \Lambda_{\theta^{primal}}(\mu_n - \tau A^T \nu_{n+1})$ ▷ Update the Primal Variable
  $\bar{\mu}_{n+1} = \mu_{n+1} + \theta(\mu_{n+1} - \mu_n)$
end for
Return $\bar{\mu}_N$

where $\Gamma_{\theta^{dual}}$ and $\Lambda_{\theta^{primal}}$ are learned networks. This is one of the experimented models. Nevertheless, the author suggests that even though algorithm 3 resulted in marginal improvements compared traditional methods, the improvements aren't large enough to justify for the resources and time required to train the networks $\Gamma_{\theta^{dual}}$, $\Lambda_{\theta^{primal}}$. Instead, the idea of algorithm 3 was taken further and more relaxations were introduced in the eventual Learned Primal Dual algorithm:

- Instead of enforcing stepsizes σ and τ in the proximal steps for both primal and dual variables, $\Gamma_{\theta^{dual}}$, $\Lambda_{\theta^{primal}}$ take $(\nu_n, A(\bar{\mu}_n), p)$, $(\mu_n, A^T\nu_{n+1})$ all as inputs into the network, thus allowing the network to learn the optimal combination between current iterates and their projections.

- The dimensionalities of both primal and dual variables are expanded, i.e. $\mu \rightarrow [\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(N)}]$, $\nu \rightarrow [\nu^{(1)}, \nu^{(2)}, \ldots, \nu^{(N)}]$, allowing some 'memories' to be stored between iterations.

- Instead of enforcing the update $\bar{\mu}_{n+1} = \mu_{n+1} + \theta(\mu_{n+1} - \mu_n)$ in the last line of algorithm 3, the update is now learned by the network $\Lambda_{\theta^{primal}}$ as well. The second channel $\mu^{(2)}$ is analogous to $\bar{\mu}$ from algorithm 3.

- Proximal operators between different iterations now have different structures, thus allowing more flexibility at the expense of introducing more parameters (i.e. instead of only having two networks $\Gamma_{\theta^{dual}}$ and $\Lambda_{\theta^{primal}}$, now have a collection of networks instead: $(\Gamma_{\theta_n^{dual}})_{n=1,2,\ldots}$ and $(\Lambda_{\theta_n^{primal}})_{n=1,2,\ldots}$, where each n correspond to a particular iteration).

Under such settings, algorithm 3 now becomes:

Algorithm 4 LPD

$\mu_0 \leftarrow \mathbf{0}$, $v_0 \leftarrow \mathbf{0}$ ▷ Initialise the starting primal and dual values
for $n = 1, 2, 3 \cdots$ **do**
  $v_{n+1} \leftarrow \Gamma_{\theta_n^{dual}}(v_n, A\mu_n^{(2)}, p)$ ▷ Update the Dual variable
  $\mu_{n+1} \leftarrow \Lambda_{\theta_n^{primal}}(\mu_n, A^T v_{n+1}^{(1)})$ ▷ Update the Primal Variable
end for
Return μ_N

3.5 Other proposed models

Lastly, we also briefly discuss some of the alternative models used for benchmarking as well as our custom model. Their performances are presented in chapter 4 and compared to LPD. The first three models also follow the idea of 'learning' the proximal operators in algorithm 1 with neural networks, but their specific structures differ slightly from LPD (algorithm 4).

1. Learned Primal: In this model, instead of learning the proximals $\Gamma_{\theta_n^{dual}}, \Lambda_{\theta_n^{primal}}$ for both primal and dual variables, only the latter is learned; the dual step is instead replaced by just computing the residual $A\mu_n^{(2)} - p$. Under such scheme, the algorithm becomes:

Algorithm 5 Learned Primal

$\mu_0 \leftarrow \mathbf{0}$, $v_0 \leftarrow \mathbf{0}$ ▷ Initialise the starting primal and dual values
for $n = 1, 2, 3 \cdots$ **do**
  $v_{n+1} \leftarrow A\mu_n^{(2)} - p$ ▷ Update the Dual variable
  $\mu_{n+1} \leftarrow \Lambda_{\theta_n^{primal}}(\mu_n, A^T v_{n+1})$ ▷ Update the Primal Variable
end for
Return μ_N

2. Learned Chambolle-Pock: as mentioned in algorithm 3.

3. Learned Total-Variation Primal Dual: In this model, we learn the proximals for the dual variables and primal variables from algorithm 2 instead of 1; the algorithm will have a slightly different form compared to LPD (4), as the linear operator K in Total Variation is augmented ($[A, \nabla]$) due to inclusion of div. Thus, to reflect this augmentation, an additional neural network is learned for updating the second dual component in algorithm 2. Under such scheme, the learned scheme becomes:

Algorithm 6 Learned Total Variation Primal Dual

$\mu_0 \leftarrow \mathbf{0}$, $v_0 \leftarrow \mathbf{0}$, $z_0 \leftarrow \mathbf{0}$ ▷ Initialise the starting primal and dual values
$\bar{\mu}_0 \leftarrow \mu_0$
for $n = 1, 2, 3 \cdots$ **do**
  $v_{n+1} \leftarrow \Gamma_{\theta_n^{dual}}(v_n, A\mu_n^{(2)}, p)$ ▷ Update the first component of the Dual variable
  $z_{n+1} \leftarrow \Phi_{\theta_n^{dual}}(z_n, \nabla\mu_n^{(2)})$ ▷ Update the second component of the Dual variable
  $\mu_{n+1} \leftarrow \Lambda_{\theta_n^{primal}}(\mu_n, A^T v_{n+1}^{(1)}, \nabla^T z_{n+1})$ ▷ Update the Primal Variable
end for
Return $\bar{\mu}_N$

Note that this algorithm isn't included in the original paper; rather, it's an extension inspired by the origianl LPD. Therefore, in Chapter 4, we include a separate results section to analyse its performance compared to standard LPD in details.

4. FBPConvNet: This is a post-processing method, which takes in a 'pseudoinverse' - in this case - the reconstruction obtained from filtered backprojection, and denoises it using a trained neural network. The network architecture is almost identical to UNet [16], barring the inclusion of additional channels during downsampling and additional residual connection between the original input and the final output. Some additional adjustments were also made to the padding and strides of the `MaxPool` blocks to ensure the channel sizes align properly during upsampling and downsampling, as our images are 362×362, different to the conventional settings where the resolutions are powers of 2.

Chapter 4

Implementation of the network and data analysis pipeline

In this chapter, we cover the pipeline used during training and analyse the results. Particularly, in sections 4.1, 4.2, 4.3, we discuss the architecture of the network used, as well as the generation procedure for noisy data and hyperparameters used during training. Later, in section 4.6, we compare the performance of different models using standard image metrics, and present detailed figures showcasing the strengths and shortcomings of LPD. Lastly, in section 4.7, we outline findings/results which weren't present in the original paper and provide a detailed analysis for them.

4.1 Architecture of the network

In algorithm 4, each iteration involves passing through current iterates through the networks $\Gamma_{\theta_n^{dual}}, \Lambda_{\theta_n^{Primal}}$. All these 'sub' networks follow the same architecture, each consisting of 3 convolution blocks with 32 channels (barring the original input channels and the final output channels) with 3×3 convolution kernels, as well as residual connection between the original input and the final output. CNN architecture is particularly attractive for imaging problems due to its translation invariant feature and emphasis on learning localised details.

Regarding the hyperparameters from algorithm 4 itself, both primal and dual iterates are extended to have 5 channels each, i.e. $\mu = [\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(5)}]$, $\nu = [\nu^{(1)}, \nu^{(2)}, \ldots, \nu^{(5)}]$; number of iterations is taken to be 10, as this strikes a good balance between quality of reconstruction and computational costs of repeatedly taking forward/back projections. The

weights in the network are Xavier initialised, biases are zero initialised, and activation function between convolution blocks are chosen to be `PReLU` functions.

4.2 Data Generation process

The LoDoPab-CT dataset [14] was used for benchmarking performances. It contains over $40,000$ slices of lung images from around 800 patients, and are split into training, test and validation sets in ratios of approximately $80:10:10$.

To simulate noisy observations, we first note that the expected number of photons arriving at the detector corresponding to ray $L_{\theta,s}$ is equal to $I_0 \times e^{-\int_{L_{\theta,s}} \mu(\ell) d\ell} = I_0 \times e^{-g(\theta,s)}$, and the observations follow a Poisson distribution (section 3.2). Thus, noisy observations for a particular pair of (θ, s) can be simulated by drawing from the distribution $Poisson(I_0 \times e^{-g(\theta,s)})$. The Signal to Noise Ratio (SNR), is one of the metrics used to assess the noise level in an image. It's defined as:

$$SNR(I) = \frac{E(I)}{\sqrt{Var(I)}} \tag{4.1}$$

and higher SNR values correspond to better image qualities. Under Poisson assumption, SNR is equal to:

$$SNR(I(\theta,s)) = \frac{I_0 \times e^{-g(\theta,s)}}{\sqrt{I_0 \times e^{-g(\theta,s)}}} = \sqrt{I_0 \times e^{-g(\theta,s)}} \tag{4.2}$$

Evidently, lower initial intensity (I_0) indicates lower SNR and lower image quality. For reference, the standard clinical dose is 10^5; in our investigation, $I_0 = 4096$ was taken as the initial intensity as per LoDoPab-CT's assumption.

Below, we outline the noise simulation procedure:

Algorithm 7 Noise Simulation procedure from ground truth images

Initialise μ by fetching an image from the dataset
$p \leftarrow \mathscr{R}(\mu)$ ▷ Obtain the **Absorption** Sinogram
$I_{max} \leftarrow maximum(p)$ ▷ Get the maximum pixel value from the Absorption Sinogram
$p_I \leftarrow exp(-p/I_{max}) * 4096$ ▷ Get the **normalised Intensity Sinogram**
$p_I \leftarrow clamp(p_I, min = 0.001)$ ▷ Ensure there is no 0 intensity
$p_{noisy} = -\log(Poisson(p_I)/4096) \times I_{max}$ ▷ Simulate Poisson noise and convert back to Absorption Sinogram
Return p_{noisy}

Note that an additional step was introduced to ensure that the Intensity Sinogram has no 0 values, as Poisson distribution is undefined for non-positive values.

4.3 Training details

During training, the Mean Squared Error (MSE) is used as the loss function:

$$\mathscr{L}_f(\theta) = \frac{1}{N}\sum_{i=1}^{N} \| f_\theta(p^*) - \mu \|_2^2 \tag{4.3}$$

Stochastic Gradient Descent (SGD) is chosen as the optimisation algorithm with batch sizes set to 1. The default `pytorch` Adam [13] optimiser was used, with β values equal to $0.99, 0.999$. Due to the large size of LoDoPab-CT, we trained the network twice. First, we only trained on a subset of the whole training dataset for 50 epochs, which consists of approximately 2600 images, then again on the full dataset, also for 50 epochs. Both models resulted in almost identical performance. Hence, all subsequent variants of LPD models (such as Learned Primal, Learned Chambolle-Pock) are also only trained on the subset dataset for resource considerations (as training **one** model using all 40000 images would require more than 200 GPU hours on HPC).

Similar to the original paper, Cosine annealing learning rate schedule [15] was used. Under such schedule, the learning rate α_t at the t-th iteration is:

$$\alpha_t = \frac{\alpha_0}{2}\left(1 + \cos\left(\pi \frac{t}{t_{max}}\right)\right) \tag{4.4}$$

where α_0 is the starting learning rate, which we take to be 0.001. This learning rate schedule is particularly good, as it employs larger updates at the beginning of training while the model

is still unrefined and smaller updates towards the end, allowing for fine-tuning of more intricate details. Also, for the prototype model, there are approximately 2600×50 iterations in the whole training process. Therefore, we take t_{max} to be 130001 in accordance. Gradient clipping was also employed during the update at each iteration to ensure stability.

Additionally, the original paper didn't propose suitable stopping criteria for training; the model was merely trained for 100000 iterations without considering convergence metrics. Thus, to improve on this shortcoming, we incorporate validation during our training scheme. At the end of each epoch, image metrics are calculated on validation images. These metrics are used to determine the optimal number of epochs. Checkpoints are saved at each epoch during training and the ones corresponding to the best epochs from validation are used for inference.

Algorithm 4 involves taking forward projection $A\mu_n^{(2)}$ and $A^T \nu_{n+1}^{(1)}$ during forward pass. Using the physical geometry proposed in section 2.1 (with distance scaling to ensure that the operator norm of the forward operator A is approximately 1 for stable forward pass), the projections are computed using the packages `tomosipo` and `ts_algorithms`, leveraging GPU's for faster computation times.

Also, `ts_algorithms.tv_min2d` contains implementation of Chambolle-Pock algorithm for Total Variation (algorithm 2). This is used to compute the total variation regularised solution and benchmark its performance against our trained neural network.

4.4 Image Metrics For Comparison

The metrics used for performance benchmarking are MSE (equation 4.3), Structural Similarity Index (SSIM) and Peak Signal to Noise Ratio (PSNR). SSIM is a measure with value between $[0, 1]$, with 0 indicating substantial difference between two images and 1 indicating perfect similarity; PSNR has no limits in its range, but higher PSNR values are associated with better quality images.

4.5 Repository Organisation

Implementations of the models are stored under `src/models`, training scripts under `src/training_scripts`, plotting scripts under `src/plotting_scripts` and evaluation scripts under

`src/evaluation _scripts`.

In the file `dataloader.py`, raw image data from `HDF5` files are read and converted into `torch.tensors` to be fed into neural networks using `pytorch`'s `Dataset` and `h5py` [6]. The file `utils.py` contains several helper functions, including implementation of algorithm 7, custom plotting functions and checkpoint saving function. `README.md` and documentation contains further details for usages of each specific script and function. The following tree diagram illustrates the skeletal repository structure.

```
root
├── docs
├── src
│   ├── utils.py
│   ├── dataloader.py
│   ├── plotting_scripts
│   │   ├── __init__.py
│   │   ├── plotting_script_1
│   │   └── plotting_script_2
│   ├── evaluation_scripts
│   │   ├── __init__.py
│   │   ├── evaluation_script_1
│   │   └── evaluation_script_2
│   ├── training_scripts
│   │   ├── __init__.py
│   │   ├── training_script_1
│   │   └── training_script_2
│   └── models
│       ├── __init__.py
│       ├── model_1.py
│       └── model_2.py
└── Other files for best practice
```

4.6 Baseline Reproduction Results

Below, we present the obtained image metrics of test set images, using filtered backprojection, total variation, LPD as well as other baseline models mentioned in section 3.5:

Method	Image Metric		
	MSE	PSNR	SSIM
FBP	$1.957 \times 10^{-3} \pm 5.560 \times 10^{-4}$	23.36 ± 2.05	0.347 ± 0.084
TV	$3.575 \times 10^{-4} \pm 3.734 \times 10^{-4}$	31.81 ± 3.14	0.747 ± 0.129
Learned PDHG	$2.735 \times 10^{-4} \pm 3.821 \times 10^{-4}$	33.93 ± 3.97	0.799 ± 0.154
Learned Primal	$2.591 \times 10^{-4} \pm 3.702 \times 10^{-4}$	34.30 ± 4.11	0.808 ± 0.153
LPD	$1.939 \times 10^{-4} \pm 3.409 \times 10^{-4}$	$\mathbf{37.29 \pm 5.55}$	0.865 ± 0.140
LPD (40000 images)	$\mathbf{1.908 \times 10^{-4} \pm 3.422 \times 10^{-4}}$	37.26 ± 5.39	$\mathbf{0.868 \pm 0.136}$
FBPConvNet	$\mathbf{1.206 \times 10^{-4} \pm 2.581 \times 10^{-4}}$	$\mathbf{39.66 \pm 5.39}$	$\mathbf{0.923 \pm 0.083}$

Table 4.1 Evaluated image metrics using different methods of reconstruction, under standard conditions

We also present the results from the original paper [1]. There were two sets of results, one for randomly generated ellipse images and one for human abdomen phantoms. Note that the results of human abdomen phantoms for variants of LPD (e.g. Learned PDHG, Learned Primal) weren't available in the original paper and therefore weren't quoted.

Method	Image Metric			
	PSNR (Ellipses)	SSIM (Ellipses)	PSNR (Human)	PSNR (Human)
FBP	19.75	0.597	33.65	0.830
TV	28.06	0.929	37.48	0.946
Learned PDHG	28.32	0.909	N/A	N/A
Learned Primal	36.97	0.986	N/A	N/A
LPD	**38.28**	**0.989**	**44.11**	**0.969**
FBPConvNet	29.20	0.944	41.92	0.941

Table 4.2 Evaluated image metrics using different methods of reconstruction, from the original paper

In both tables, LPD demonstrates substantial better performance compared to FBP and TV across all metrics. Moreover, compared to TV (the better performing classical reconstruction method), the computation time of LPD is significantly lower, making it a more desirable reconstructor for clinical settings. LPD also beats other baseline 'learned' models, such as Learned PDHG and Learned Primal, which is expected due to its more sophisticated architecture.

However, the largest divergence in results occurred on the performance of FBPConvNet. In the original paper, LPD netted superior performances for both human phantoms and ellipse images, but in our investigation, LPD performed marginally worse than FBPConvNet. We theorise that this contradiction in results may be attributed to differences between datasets. In contrary to the ellipse images and 512×512 human abdomen phantoms used in the original investigation, the images we used are 362×362 scans of lungs, which are quite different in nature. Therefore, the particular details learned by the networks may deviate. Also, considering that our geometry setup has very high number of projection angles (1000), the amount of noise present in the FBP reconstructions are limited, which makes the denoising network a lot easier to learn. Therefore, FBPConvNet will be additionally evaluated under more adversarial conditions in section 4.7 to assess its overall capability.

Below, in figure 4.1, we compare the performances of the aforementioned methods empirically; from raw inspection, the differences between the methods don't seem too significant. However, upon zooming in particular subsections, it can be seen that LPD was the best at capturing small, local features compared to the other models. Its only caveat is that it suffers somewhat from over-smoothness.

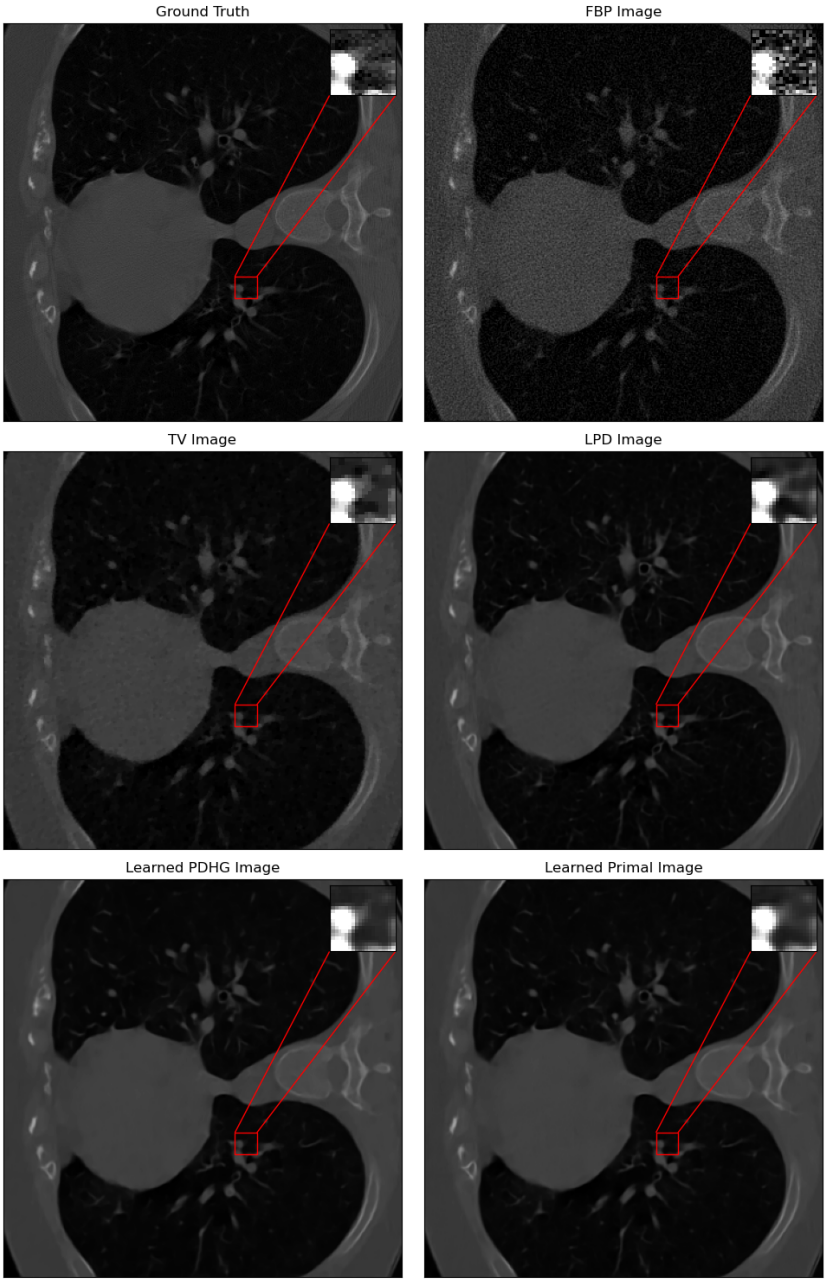
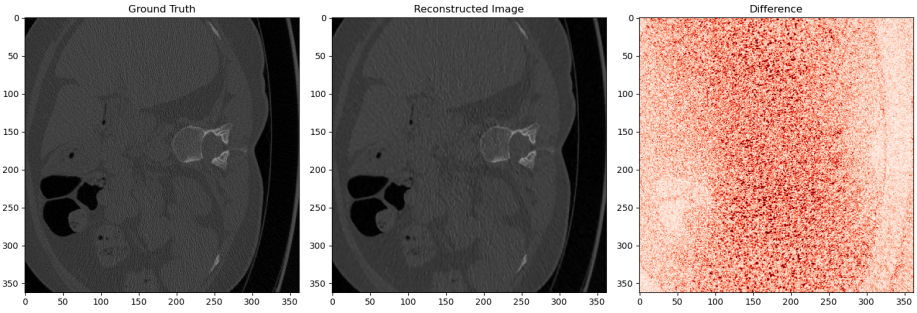
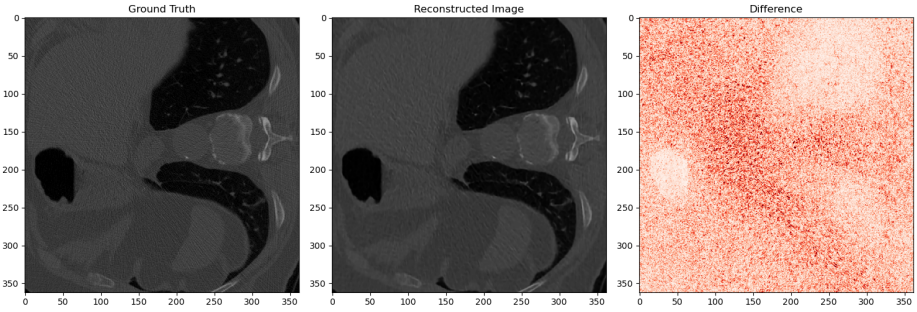


Fig. 4.1 Comparison of reconstruction methods, including FBP, TV, LPD, Learned PDHG and Learned Primal. Pixel values in zoomed-in subsections are clipped to ensure apparent comparison between the different reconstructions

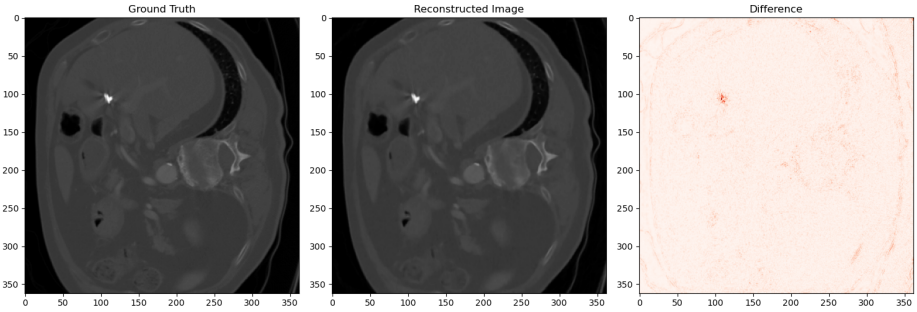
In figure 4.2, representative samples of both good and poor quality reconstructions are visualised, along with the ground truth images and difference images.



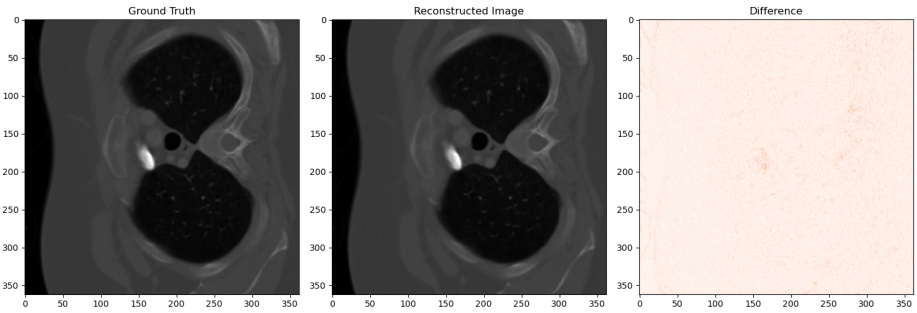
(a) Bad Quality LPD Reconstruction sample 1 (PSNR=24.50, SSIM=0.449)



(b) Bad Quality LPD Reconstruction sample 2 (PSNR=25.78, SSIM=0.479)



(c) Good Quality LPD Reconstruction sample 1 (PSNR=45.04, SSIM=0.979)



(d) Good Quality LPD Reconstruction sample 2 (PSNR=46.89, SSIM=0.986)

Fig. 4.2 Representative cases of good/bad quality reconstructions. From left to right: a) Ground Truth Image; b) Reconstructed Image; c) Difference Image.

In bad quality cases, the reconstructed images appear to not have been denoised completely, evidenced by the rightmost plots in rows 1 and 2 of figure 4.2. Its silver lining is that even in poor quality reconstructions, structural fidelities of objects are still preserved, indicating that the error is not systematic. On the other hand, in good quality reconstructions, the only visible caveat is loss of sharpness around edges and small, distinctive local features. This can be most evidently seen in the difference image of the second last row in figure 4.2.

Interestingly, the model appears to perform better when there is a clear distinction between signal and background. For example, in good quality samples from figure 4.2, the 'signals' in the ground truth images are very apparent, evidenced by the bright spots in the binary images; on the other hand, the range of attenuation values in images with poor reconstructions tend to be smaller, as majority of the image consists of only grey background and very little brightness.

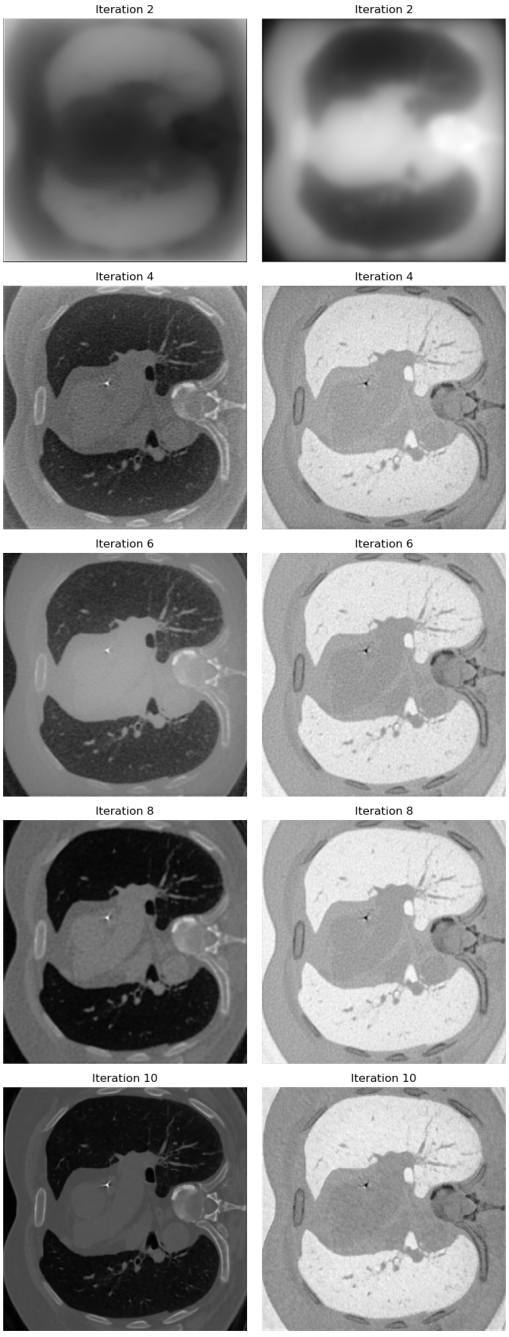
To this end, the effect of 'spread' of pixel values on quality of reconstructions is explored; using PSNR < 25 / SSIM < 0.6 as the threshold for 'poor' quality reconstruction and PSNR > 45 / SSIM > 0.98 as 'good' quality reconstruction, the range and interquartile range (IQR) of pixel values for images in both categories are calculated. We present the findings in the table below:

Good / Bad quality reconstructions	Range / IQR	
	Range	IQR
Bad quality reconstructions	0.66	0.06
Good quality reconstructions	0.90	0.20

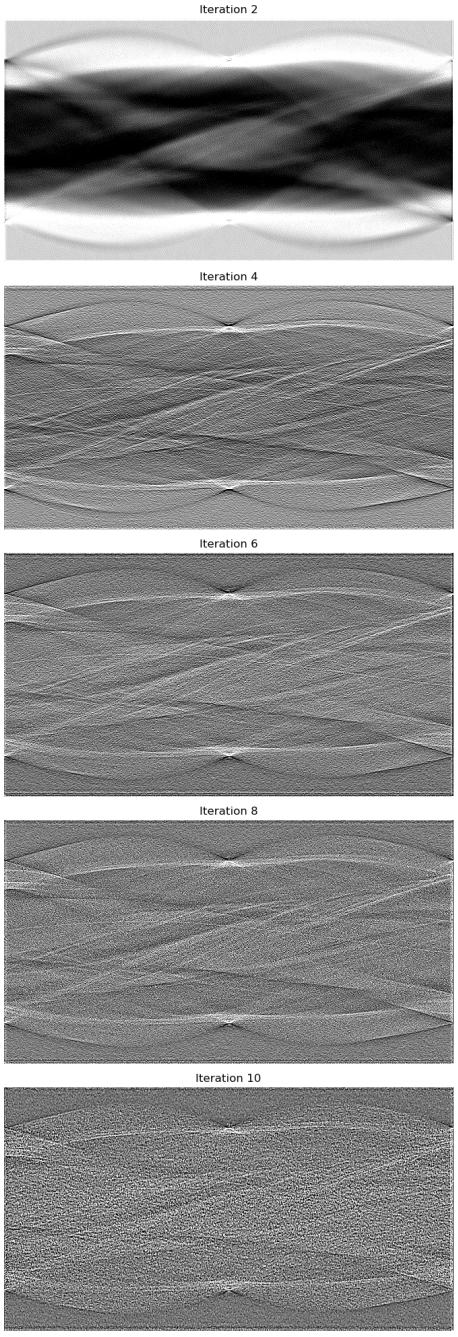
Table 4.3 Spreads of pixel values for reconstructions of differnent qualities

The differences between the two categories does seem significant enough to suggest that the dynamic range in pixel values can affect the quality of reconstruction. We hypothesise that this is because images with bright 'signals' have more prominent structures, thus are more robust against noise.

Lastly, we also visualise the primal and dual variables used in the feed forward process (i.e. $\mu^{(2)}, \nu^{(1)}$ in algorithm 4) as well as the primal variable used for reconstruction (i.e. $\mu^{(1)}$ in algorithm 4) for every 2 unrolled iterations; they are presented in figures 4.3(a) and 4.3(b). We also visualise the same plots from the original paper in figures 4.4(a) and 4.4(b):

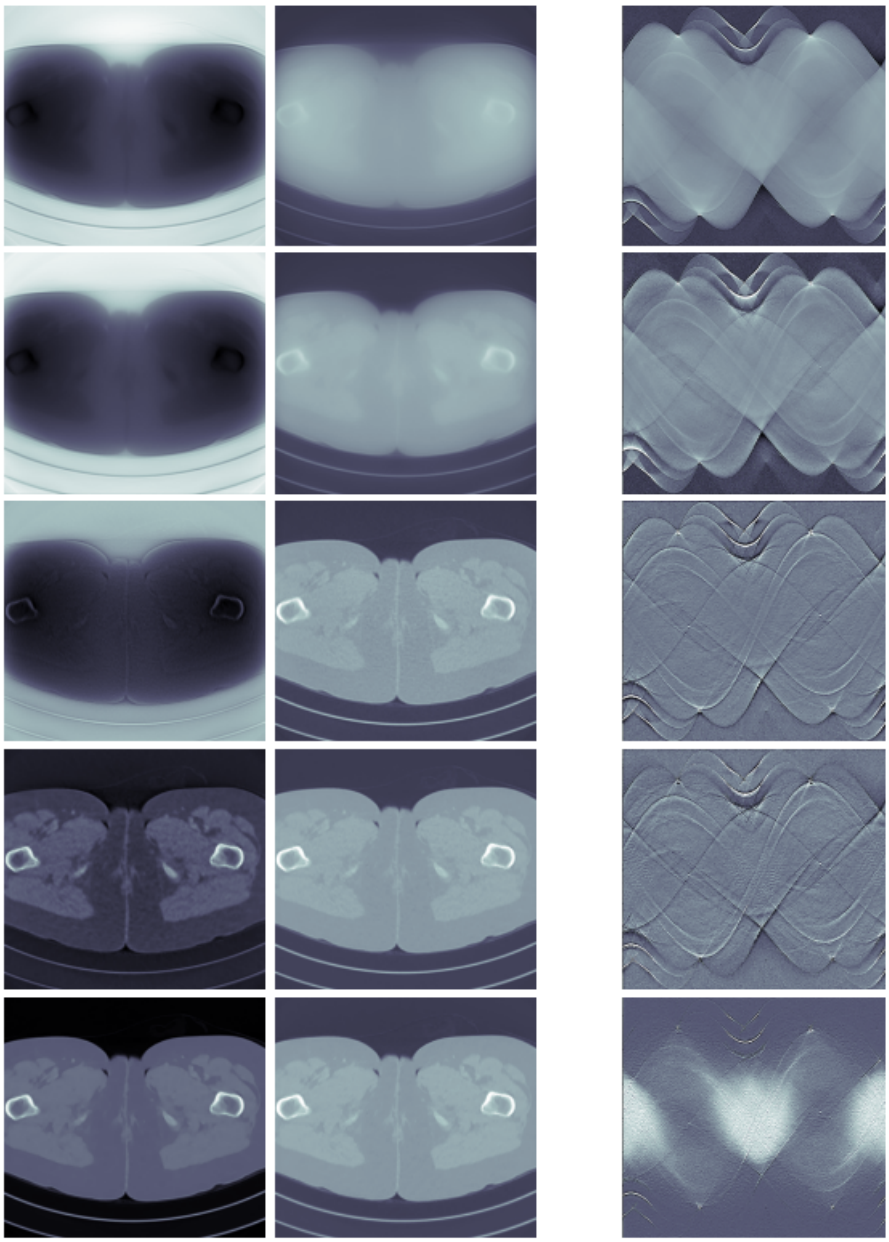


(a) Plot of the first and second primal variables at unrolled iterations 2, 4, 6, 8, 10



(b) Plot of the first dual variables at unrolled iterations 2, 4, 6, 8, 10

Fig. 4.3 Illustrations of $\mu^{(1)}, \mu^{(2)}, \nu^{(1)}$ at iterations 2, 4, 6, 8, 10



(a) Plot of the first and second primal variables at unrolled iterations 2, 4, 6, 8, 10

(b) Plot of the first dual variables at unrolled iterations 2, 4, 6, 8, 10

Fig. 4.4 Illustrations of $\mu^{(1)}, \mu^{(2)}, \nu^{(1)}$ at iterations 2, 4, 6, 8, 10, from the original paper ([1])

Interestingly, our plots (fig. 4.3) differ from the original paper's (fig. 4.4). In the original primal iterate plots 4.4(a), the reconstruction iterate ($\mu^{(1)}$) only starts to resemble the ground truth image at around iteration 8. However, in our plot (4.3(a)), it can be seen that the main

structure of the original image was already captured as early as iteration 2, and a reasonable reconstruction is achieved at iteration 4; later iterations after 4 only seemed to denoise and edge enhance on the existing reconstruction. As for the dual variables, the author claimed that original plot (4.4(b)) demonstrated low pass and high pass filtering being done in the unrolled iterations. However, in figure 4.3(b), the iterate plots demonstrate no such behaviour. We suspect that the iterates' observations seen in the original paper were only specific for their dataset, but doesn't hold true in general. Nevertheless, since our quantitative results from table 4.6 largely align with the findings in the original paper, divergences in intricate details as such are acceptable.

4.7 Extension Results

In section 3.5, a variant of LPD was proposed - namely - the TV-LPD model, where the proximals from the **Total variation** Chambolle-Pock algorithm (alg. 2) (instead of the standard Chambolle-Pock algorithm (alg. 1)) are replaced with learned neural networks. We hypothesise that due to Total Variation's strong performance in edge preservation, a neural network model which utilises its optimisation algorithm will also inherit this strength.

Below, TV-LPD's performance is compared with LPD's; since LPD is already very good at reconstruction under the standard conditions (original physical setup from section 2.1), both models were additionally trained under more adversarial conditions to see whether bigger distinctions can be made between their performances:

1. Sparse View condition: very low number of projection angles (60), very low initial intensity (1000 photons) and full rotation (0 to 180 degrees).

2. Limited View condition: very low number of angles (60), very low initial intensity (1000), limited rotating geometry (0 to 60 degrees).

We also compare FBPConvNet's performance with these learned methods under adversarial conditions to further assess its overall capability as a model.

In table 4.7, we present the results of training both under standard conditions as well as adversarial conditions:

Condition	Image Metric		
	MSE	PSNR	SSIM
LPD Standard	$1.94 \times 10^{-4} \pm 3.41 \times 10^{-4}$	37.29 ± 5.55	0.865 ± 0.140
TV-LPD Standard	$1.88 \times 10^{-3} \pm 3.40 \times 10^{-4}$	37.68 ± 5.73	0.872 ± 0.138
FBPConvNet Standard	$\mathbf{1.21 \times 10^{-4} \pm 2.58 \times 10^{-4}}$	$\mathbf{39.66 \pm 5.39}$	$\mathbf{0.923 \pm 0.083}$
LPD Limited	$1.35 \times 10^{-3} \pm 6.42 \times 10^{-4}$	25.28 ± 2.11	0.643 ± 0.142
TV-LPD Limited	$\mathbf{1.21 \times 10^{-3} \pm 5.80 \times 10^{-4}}$	$\mathbf{25.74 \pm 2.11}$	$\mathbf{0.655 \pm 0.144}$
FBPConvNet Limited	$2.36 \times 10^{-3} \pm 2.11 \times 10^{-3}$	23.34 ± 2.60	0.536 ± 0.110
LPD Sparse	$3.00 \times 10^{-4} \pm 4.30 \times 10^{-4}$	33.70 ± 4.10	0.795 ± 0.161
TV-LPD Sparse	$\mathbf{2.78 \times 10^{-4} \pm 4.24 \times 10^{-4}}$	$\mathbf{34.39 \pm 4.41}$	$\mathbf{0.808 \pm 0.161}$
FBPConvNet Sparse	$4.02 \times 10^{-4} \pm 4.44 \times 10^{-4}$	31.45 ± 3.17	0.729 ± 0.150

Table 4.4 Evaluated image metrics using different methods, under adversarial conditions

Observably, TV-LPD outperforms the standard LPD model in all metrics. In terms of MSE and SSIM, the image metrics improve the most (proportionally) under limited angles conditions, with MSE obtaining an $\approx 10\%$ decrease and SSIM obtaining an $\approx 1.2\%$ increase. On the other hand, PSNR improved the most under sparse view condition, boasting an $\approx 2\%$ improvement. The learned reconstructors also prove to be more robust under adversarial conditions compared to FBPConvNet, as they obtained better results across all image metrics under both Sparse View and Limited View. This evidences FBPConvNet's performance limitation under extremely low-dose environments.

Also, from an efficiency perspective, considering that FBPConvNet has $\approx 3 \times 10^7$ parameters, storing models of such magnitude requires considerable storage resources ($\approx$ 400MB). On the other hand, the learned models only has $\approx 3 \times 10^5$ parameters, translating to $\approx$ 3MB, making them much more storage-friendly.

Empirically, we compare the performances of LPD and TV-LPD in similar manners as figures 4.1 and 4.2. Figures 4.5, 4.6, 4.7 and 4.8 showcase representative samples where TV-LPD significantly outperformed LPD:

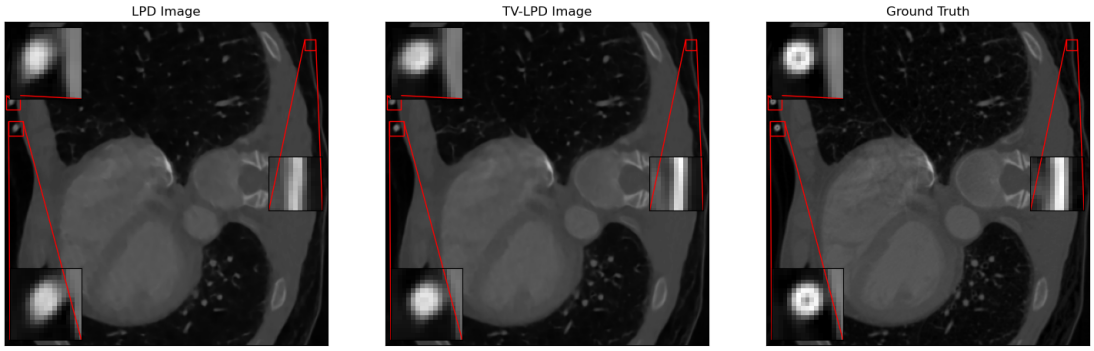


Fig. 4.5 Comparison of TV-LPD and LPD reconstructions under ‘Sparse View’ condition, zoomed in areas where TV-LPD improved the most

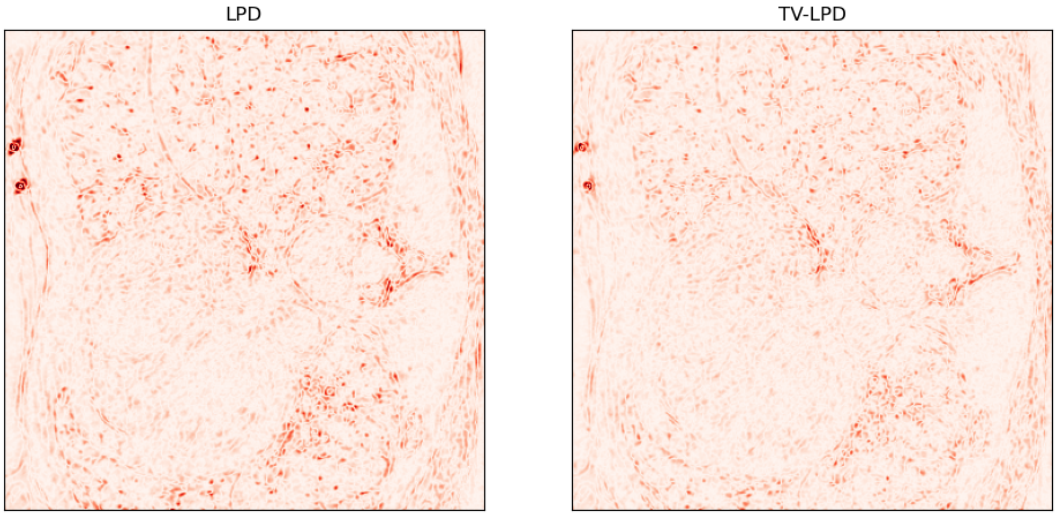


Fig. 4.6 Comparison of reconstruction difference images of TV-LPD and LPD, under ‘Sparse View’ condition.

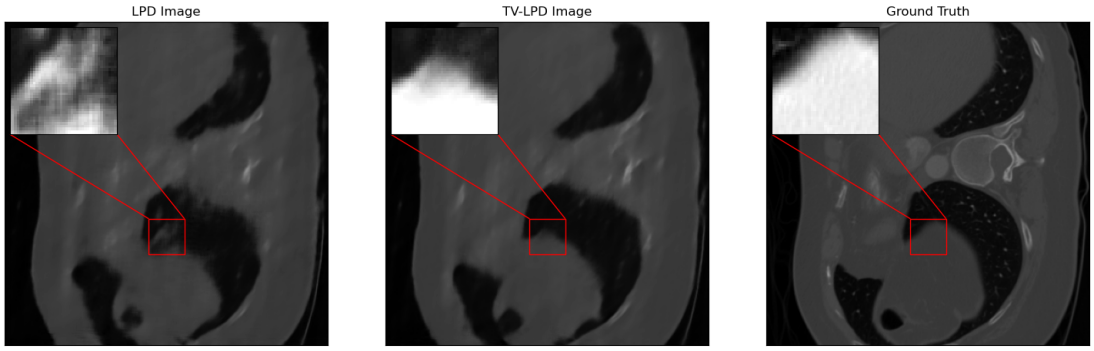


Fig. 4.7 Comparison of TV-LPD and LPD reconstructions under 'Limited View' condition, zoomed in areas where TV-LPD improved the most

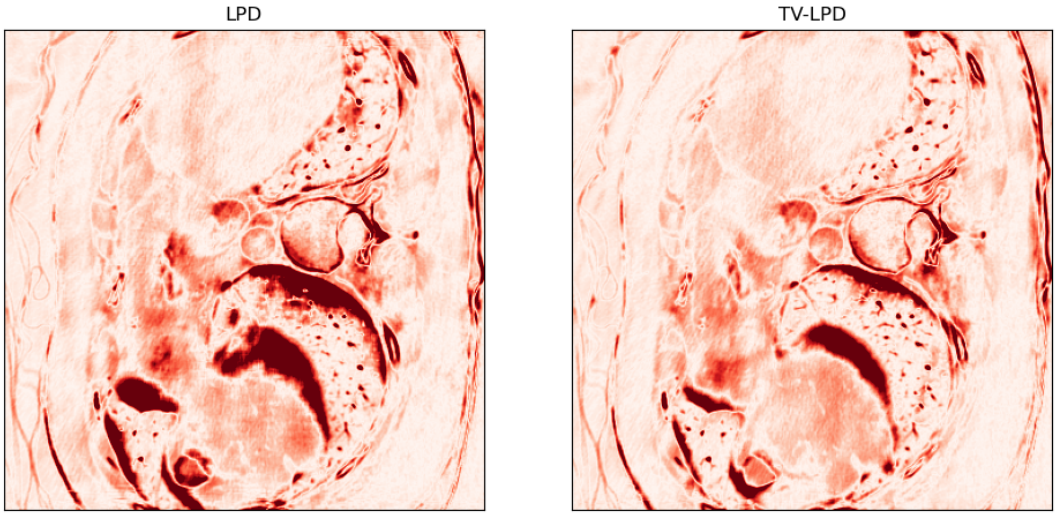


Fig. 4.8 Comparison of reconstruction difference images of TV-LPD and LPD, under 'Limited View' condition.

Under 'Limited View' conditions, although good improvement can be obtained under some cases (such as the sample showcased in figures 4.7 and 4.8), such magnitude of improvement isn't always guaranteed; in fact, in approximately 10% of all test images, the standard LPD algorithm outperforms TV-LPD in at least one of the image metrics, making it difficult to claim that TV-LPD is a consistently better model than LPD.

On the other hand, under 'Sparse View' conditions, TV-LPD boasted better performance in all image metrics for all test images, though the extent of improvement is relatively marginal. As mentioned in section 4.6, standard LPD suffers somewhat from over-smoothing around edges.

This can be particularly seen in the leftmost plot in figure 4.8, where the areas with largest difference compared to the ground truth are around the boundaries of the object. TV-LPD salvages this to a small extent: in figure 4.7, the two methods of reconstructions are compared to the ground truth, specifically zoomed in on the areas where the LPD reconstructions have the largest divergence. As can be seen, the brightness of the zoomed in subsections from TV-LPD reconstructions resemble more to the ground truth image. Also, the shades of red in difference image for TV-LPD from figure 4.8 appear lighter than its counterpart on the left, evidencing that there's less divergence from the ground truth in TV-LPD's reconstruction.

Lastly, it's worth noting that both training and validation loss of TV-LPD decrease more per epoch on average, at the expense of slightly longer training time. In figures below, we present the evolutions of both training and validation losses under the two different schemes in 50 epochs. The increase in training time is $\approx 10\%$, but the same level of training/validation loss of training 50 epochs under standard LPD can usually be achieved by TV-LPD in $25 - 35$ epochs, which is more advantageous in a training efficiency point of view.

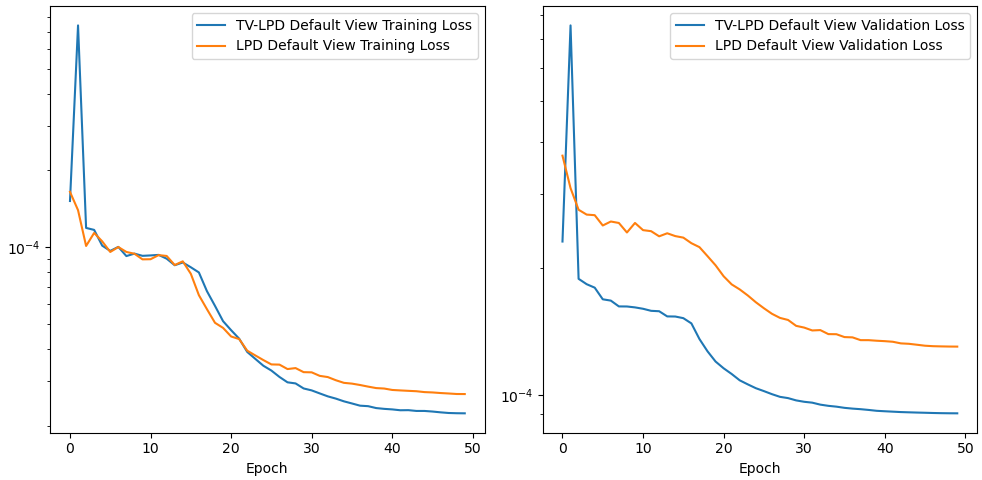


Fig. 4.9 Evolution of training Loss (left) and Validation Loss (right) of TV-LPD and LPD, under default geometry.

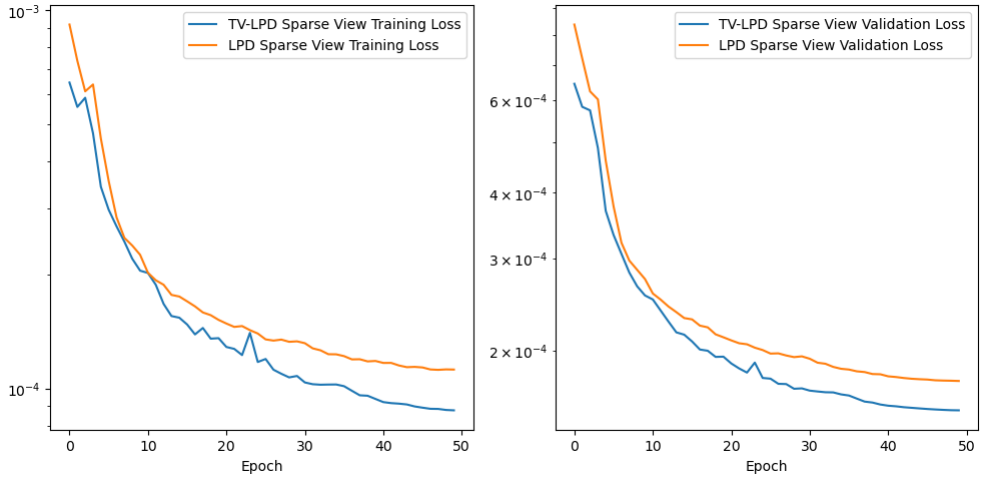


Fig. 4.10 Evolution of training Loss (left) and Validation Loss (right) of TV-LPD and LPD, under 'Sparse' geometry.

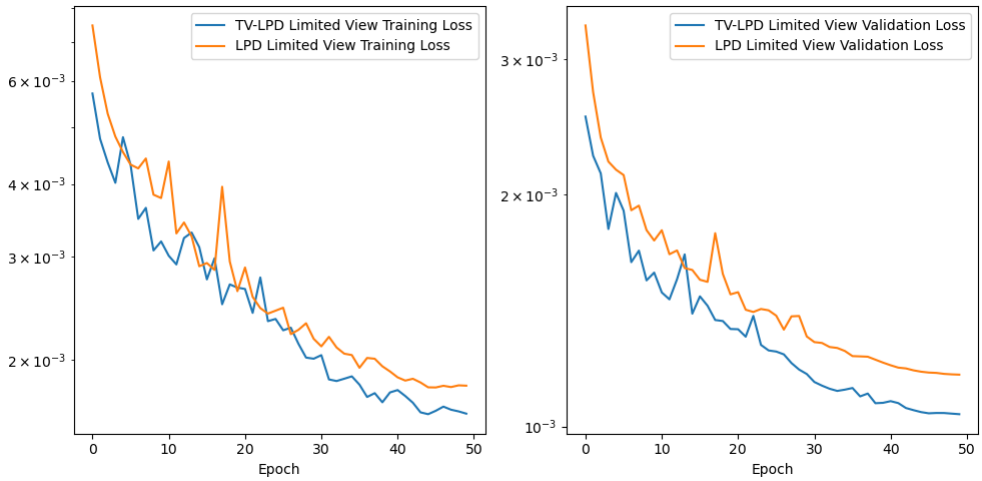


Fig. 4.11 Evolution of training Loss (left) and Validation Loss (right) of TV-LPD and LPD, under 'Limited' geometry.

Chapter 5

Concluding Remarks

In summary, this thesis extensively explored both LPD's theoretical underpinnings as well as its empirical performance compared other benchmark models. In Chapters 2 and 3, we began by walking through relevant mathematical underpinnings of CT imaging and classical non-machine based learning reconstruction methods. The discussion of one of the methods, Total Variation, naturally led to the introduction of proximals and Chambolle-Pock algorithm, which provide the theoretical foundations of LPD. In Chapter 4, results from various different models are presented and analysed. Additionally, beyond baseline reproduction of the original paper, we also introduced originality through proposing a custom model inspired by LPD, whose performance was competitive compared to other state-of-the-art models.

To conclude the thesis, the reproducibility and limitations of LPD are discussed in the following subsections. We also highlight potential future extensions for improvements.

5.1 Reproducibility

From a technical perspective, there weren't significant road blocks in implementing the algorithms from the original paper. The code repository (hosted at `https://github.com/adler-j/learned_primal_dual/tree/master/reference`) combined with the paper contained more than sufficient information. The main challenge for exact reproduction of results occurred due to differences in images between datasets, as our images contained different dimensions and physical setup compared to the original, thus causing details learned by the networks to deviate. The resultant effect are evidenced by divergences in results (table 4.6) and empirically observed iteration plots (figures 4.3(a), 4.3(b)).

Another obstacle faced, was that the models are very resource intensive. For all models, training requires between $10 \sim 36$ hours, even after leveraging HPC's efficient GPU resources. Moreover, the dataset used for training exceeded over 100GB, making it very difficult to host and train locally.

In our implementation, to ensure reproducibility of results, all training and evaluation were done with fixed seeds. The specific environment configurations used are detailed in `environment.yml`, and an identical blueprint of the training environment can be replicated from it. Moreover, `Docker` was used for containerisation, so all code can be ran on virtual machines with identical settings even across different operating systems. Lastly, appropriate documentation and example notebook are included to detail users on the implementations of functions/classes as well as commands used to run the scripts.

5.2 Limitations of the model

One of LPD's major flaws, is that it can sometimes create details which aren't present in the original image, particularly when the projection angles are limited, as exemplified in figure 5.1. This is problematic in clinical settings, as it may lead to false diagnosis. In scenarios such as breast scans and dental scans, it's often physically impossible for machines to make full rotation around the patient's body. LPD's effectiveness is hindered in these situations.

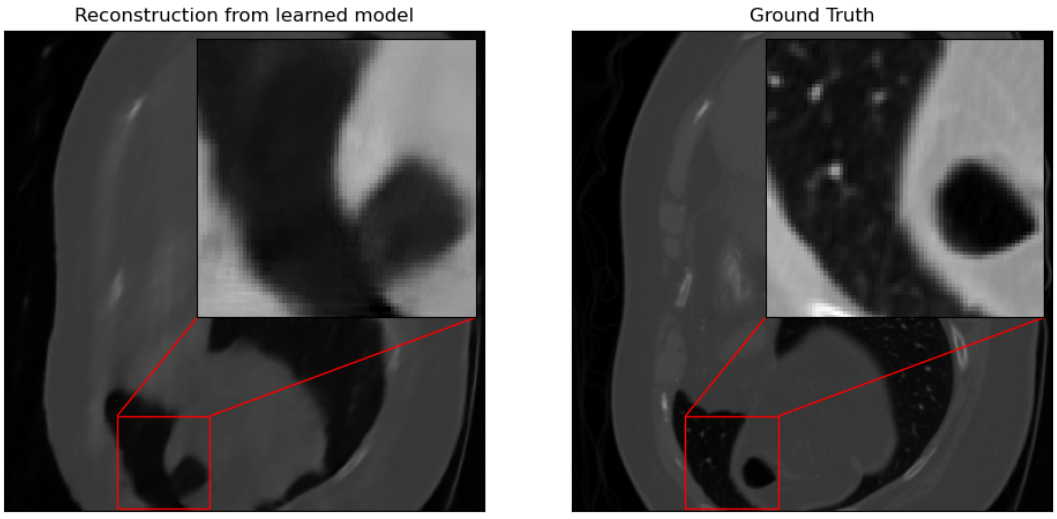


Fig. 5.1 Example of inconsistent Geometry being created under limited view

Another flaw is that the trained models are not very generalisable; their performances deteriorate rapidly with changes in physical setup. For example, when evaluating the LPD model trained under standard settings ($I_0 = 4096$) on images with marginally different noise levels (e.g. $I_0 = 3000$), the evaluated metrics dropped substantially. This lack of flexibility may particularly pose problems in clinical settings, as any changes in equipment would require the models being retrained. This may not be practical due to the extensiveness of resources needed to train these models to good standards.

5.3 Potential extensions and future work

In conclusion, whilst LPD and our custom extension TV-LPD demonstrated impressive results, there is still major room for improvement. Aside from the aforementioned flaws, a potential area to explore is changing the structures of the component subnetworks $\Gamma_{\theta_n^{dual}}$ and $\Lambda_{\theta_n^{primal}}$. Across all implemented models, these subnetworks share the same structure, consisting of three 3×3 convolution blocks with 32 channels. Replacing them with more sophisticated network architectures, such as UNet, could potentially result in improved performances.

On another front, taking inspiration from implementations of diffusion models [10], an alternative extension idea would be inclusion of positional embeddings such as [20]. Referring back to algorithm 4, separate networks are learned for each unrolled iteration. Although LPD already contains relatively minimal number of parameters ($\approx 3 \times 10^5$), this number could potentially be reduced even further; instead of learning separate networks for each iteration, the same networks $\Gamma_{\theta^{dual}}$, $\Lambda_{\theta^{primal}}$ can be used across all iterations, with an embedding of the iteration number being passed as an additional input as informant for current position in the reconstruction process. Under such scheme, the altered LPD algorithm would be:

Algorithm 8 LPD with positional embedding

$\mu_0 \leftarrow \mathbf{0}$, $\nu_0 \leftarrow \mathbf{0}$ ▷ Initialise the starting primal and dual values

for $n = 1, 2, 3, \cdots, N$ **do**

$n_{embed} = Embedding\_MLP(n)$ ▷ Transform n to an embedded input

$\nu_{n+1} \leftarrow \Gamma_{\theta^{dual}}(\nu_n, A\mu_n^{(2)}, p, n_{embed})$ ▷ Update the Dual variable

$\mu_{n+1} \leftarrow \Lambda_{\theta^{primal}}(\mu_n, A^T\nu_{n+1}^{(1)}, n_{embed})$ ▷ Update the Primal Variable

end for

Return μ_N

Lastly, as per workings from the recent paper [18], we also experimented with incorporating neural ODEs [5] in LPD. This implementation revolves around replacing discrete convolution blocks with continuous ODEs, which are empirically proven to be more robust under adversarial conditions. Unfortunately, there was limited success to this end, as our models suffered from numerical instability during training and couldn't converge. Nevertheless, the model and training scripts are still included in repository for completeness.

References

[1] J. Adler and O. Oktem, "Learned primal-dual reconstruction," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, p. 1322–1332, June 2018. [Online]. Available: http://dx.doi.org/10.1109/TMI.2018.2799231

[2] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," 2016. [Online]. Available: https://arxiv.org/abs/1606.04474

[3] M. Benning and M. Burger, "Modern regularization methods for inverse problems," *Acta Numerica*, vol. 27, p. 1–111, 2018.

[4] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, no. 1, p. 120–145, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1007/s10851-010-0251-1

[5] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," 2019. [Online]. Available: https://arxiv.org/abs/1806.07366

[6] A. Collette, *Python and HDF5*. O'Reilly, 2013.

[7] S. Dr. Gao, "Plot curly brace with matplotlib," 2019. [Online]. Available: https://github.com/iruletheworld/matplotlib-curly-brace

[8] P. C. Hansen, J. Jørgensen, and W. R. B. Lionheart, *Computed Tomography: Algorithms, Insight, and Just Enough Theory*, P. C. Hansen, J. Jørgensen, and W. R. B. Lionheart, Eds. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2021. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611976670

[9] A. Hendriksen, D. Schut, W. J. Palenstijn, N. Viganò, J. Kim, D. Pelt, T. van Leeuwen, and K. J. Batenburg, "Tomosipo: Fast, flexible, and convenient 3D tomography for complex scanning geometries in Python," *Optics Express*, Oct 2021. [Online]. Available: https://doi.org/10.1364/oe.439909

[10] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[11] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017. [Online]. Available: https://doi.org/10.1109/TIP.2017.2713099

[12] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. Society for Industrial and Applied Mathematics, 2001. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898719277

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[14] J. Leuschner, M. Schmidt, D. O. Baguer, and P. Maass, "LoDoPaB-CT, a benchmark dataset for low-dose computed tomography reconstruction," *Sci. Data*, vol. 8, p. 109, 2021. [Online]. Available: https://doi.org/10.1038/s41597-021-00893-z

[15] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2017. [Online]. Available: https://arxiv.org/abs/1608.03983

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[17] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992. [Online]. Available: https://doi.org/10.1016/0167-2789(92)90242-F

[18] C. Runkel, A. Biguri, and C.-B. Schönlieb, "Continuous learned primal dual," 2024. [Online]. Available: https://arxiv.org/abs/2405.02478v1

[19] E. Y. Sidky, J. H. Jørgensen, and X. Pan, "Convex optimization problem prototyping for image reconstruction in computed tomography with the chambolle–pock algorithm," *Physics in Medicine Biology*, vol. 57, no. 10, p. 3065, apr 2012. [Online]. Available: https://dx.doi.org/10.1088/0031-9155/57/10/3065

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

Appendix A

Derivation of filtered backprojection

We first write $\mu(x,y)$ as the inverse Fourier transform of its Fourier transform $\hat{\mu}(u,v)$:

$$\mu(x,y) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\hat{\mu}(u,v)e^{i(ux+vy)}dudv \tag{A.1}$$

Switching to polar coordinates, the integrand can be rewritten as:

$$\mu(x,y) = \frac{1}{2\pi}\int_{0}^{2\pi}\int_{0}^{\infty}\hat{\mu}(\omega\cos\theta,\omega\sin\theta)e^{i(\omega\cos\theta x+\omega\sin\theta y)}\omega d\omega d\theta \tag{A.2}$$

By Fourier Slice Theorem, the inner integral can be rewritten in terms of $\hat{g}(\theta,\omega)$, as following:

$$\mu(x,y) = \frac{1}{2\sqrt{2}\pi}\int_{0}^{2\pi}\int_{0}^{\infty}\hat{g}(\theta,\omega)e^{i(\omega\cos\theta x+\omega\sin\theta y)}\omega d\omega d\theta \tag{A.3}$$

By noting that $\hat{g}$ is even in ω, (as g is a real-valued function), equation A.3 can be rewritten as:

$$\begin{aligned}\mu(x,y) = \frac{1}{4\sqrt{2}\pi}\Bigg(&\int_{0}^{2\pi}\int_{0}^{\infty}\hat{g}(\theta,\omega)e^{i(\omega\cos\theta x+\omega\sin\theta y)}\omega d\omega d\theta \\ &-\int_{0}^{2\pi}\int_{-\infty}^{0}\hat{g}(\theta,\omega)e^{i(\omega\cos\theta x+\omega\sin\theta y)}\omega d\omega d\theta\Bigg)\end{aligned} \tag{A.4}$$

which can be simplified to:

$$\mu(x,y) = \frac{1}{4\pi}\int_{0}^{2\pi}\frac{1}{\sqrt{2\pi}}\left[\int_{-\infty}^{\infty}\hat{g}(\theta,\omega)|\omega|e^{2\pi i\omega\mathbf{x}\phi}d\omega\right]d\theta \tag{A.5}$$

where the $\mathbf{x}$ term in the final line represents $(x\cos\theta, y\sin\theta)$.

Appendix B

Ackowledgement of Generative AI tools

During the completion of thesis, generative tools such as ChatGPT and CoPilot were used supportively and minimally. All code involving any algorithms or calculations were entirely produced by myself; Copilot was only partially used for docstring generation and plotting, and ChatGPT was only used for latex syntax queries. Examples of ChatGPT prompts include:

"Arrange the two following to figures such that they are aligned properly in latex."