Photoacoustic Image Reconstruction using Continuous Neural Representations

MSc Thesis

Arnon A.B.

08/05/2025

Graduation Committee:

Prof. Dr. C. Brune
Prof. Dr. S. Manohar
Prof. Dr. Ir. I.M. Vellekoop
Dr. S.M. Glas
Dr. B. De Santi

Abstract

Photoacoustic imaging techniques are capable of creating high-resolution images with high optical contrast. It has the potential to be used for breast cancer screening, but there are still some hurdles to overcome. One of them is the need for faster reconstruction algorithms, which is the main topic of this thesis. Reconstructing photoacoustic images over large fields of view is computationally expensive, not just because of the large domains involved, but also because the measurements can be sparse, requiring many iterations of an iterative algorithm. In this thesis, several deep learning techniques, known as continuous neural representations, are explored to see whether they can speed up photoacoustic reconstruction and improve the reconstruction quality by providing a continuous view of the underlying objects. Experimental results show significant speed-ups, up to a factor of 100 for small computational domains. It is also shown that these continuous neural representation reconstructions can be evaluated on an arbitrarily fine grid. However, this does not imply that arbitrarily small features are actually recovered, which raises questions about how "continuous" these representations really are.

Acknowledgements

I have spent the past year at the University of Twente working on an assignment in Srirang's group, Multi Modality Medical Imaging (M3I), and Christoph's group, Mathematics of Imaging and AI (MIA). Even though I already had a solid background, I still learned many new things throughout the project, thanks to the support of my supervisors, which helped me complete this thesis.

I would like to thank Bruno for supervising me throughout the project. I really appreciate how you were always willing to help, and your positivity was very contagious, which was especially helpful when I was feeling very pessimistic.

Also, a big thank you to Srirang for supervising me. I really appreciate how you always tried to understand my terrible explanations, and how you asked great questions that helped me move forward.

Finally, I would like to thank Christoph for all the supervision and the many long meetings we had. I feel like I have learned a lot from you. Thanks for taking the time and having the patience to go through everything with me!

Contents

Chapter 1

Introduction

1.1 Assignment Description

Breast cancer is the most common cancer type in The Netherlands for women, and 1 in 7 of all women in The Netherlands get breast cancer [1]. The large prevalence of breast cancer is the reason why a national screening program has been set up by the Dutch government [2]. An often reported downside of the mammography screening is the need for the breasts to be compressed, which can be quite painful. Besides that, tumours can be missed, especially in dense breast tissue. To solve these problems, other methods are investigated, one them being photoacoustic imaging.

The 3rd generation photoacoustic mammoscope (PAM3) is a powerful breast imaging system which is capable of providing high resolution images of the blood vessels and quantitative speed of sound maps of the breast. This device can image the full breast with a high resolution and is capable of imaging vessels up to 0.4mm [3]. The PAM3 can offer high optical contrast without the need for compressing the breast. The PAM3 does comes with various drawbacks though, the following two of which will be studied in the report:

1. **Slow speed of sound corrected image reconstruction**: Currently, a classical iterative optimization procedure is used for photoacoustic (PA) image reconstruction which compensates for an inhomogeneous speed of sound (SOS) [3]. The most time consuming step in each iteration is simulating the wave propagation, which takes 10 minutes on a modern GPU. Two of these wave simulations are performed in each iteration, resulting in a computation time of 3 hours and 20 minutes for just 10 iterations. This is just for a single wavelength. Image reconstruction for a single wavelength is not enough for quantitative PA imaging, which requires images to be reconstructed for multiple wavelengths. Reconstructing the PA images for 5 different wavelengths takes 20 hours, which exceeds any acceptable time limit for PAM to be used for screening purposes.

2. **Detector sparsity**: The PAM3 has 512 detectors [3] which is not enough to reconstruct photoacoustic images of the desired quality (detector sparsity). In order to increase the number of detectors artificially, the array of detectors can be moved to a new position by rotating the bowl that holds all the detectors, and performing multiple measurements for different positions of the imaging bowl. This whole process of data acquisition at each position of the image bowl followed by rotating the bowl to a new position takes 5 minutes for a standard imaging protocol [3], during which data can be corrupted, for example as a result of breathing artifacts.

In this report, an attempt will be made to solve these problems using a novel class of deep learning techniques, specifically, neural networks that treat objects and operators as continuous objects instead of discretizing them: continuous neural representations.

1.2 Report Outline

The main topic of this report is investigating several techniques that solve the sparsity problem and the problem of slow reconstructions, introduced in the previous section. Before discussing the different methods investigated for solving these problems, some necessary background knowledge is required to understand the rest of this report. This general background knowledge will be provided in **chapter 2**. Next, three chapters follow, each discussing a method for solving the previously stated problem. If we think of forward and inverse imaging problems as two domains connected by operators, as shown in figure 1.1, we can place the three different chapters in different locations in this image. **Chapter 3** discusses a technique that works in the measurement domain. A technique based on neural functions is described for inpainting missing data. Next, so-called neural operators are introduced in **chapter 4**, which can be used as surrogates for classical wave solvers to speed up the reconstruction process. They can not only describe the forward problem, described by the wave-equation, but also time-reversal and adjoint wave simulations. **Chapter 5** discusses deep operator networks for describing the inverse operator, which takes objects in the measurement domain and outputs the corresponding source in the image domain. In the two remaining chapters, **chapter 6** and **chapter 7**, the results will be discussed and an outlook for the future will be given.

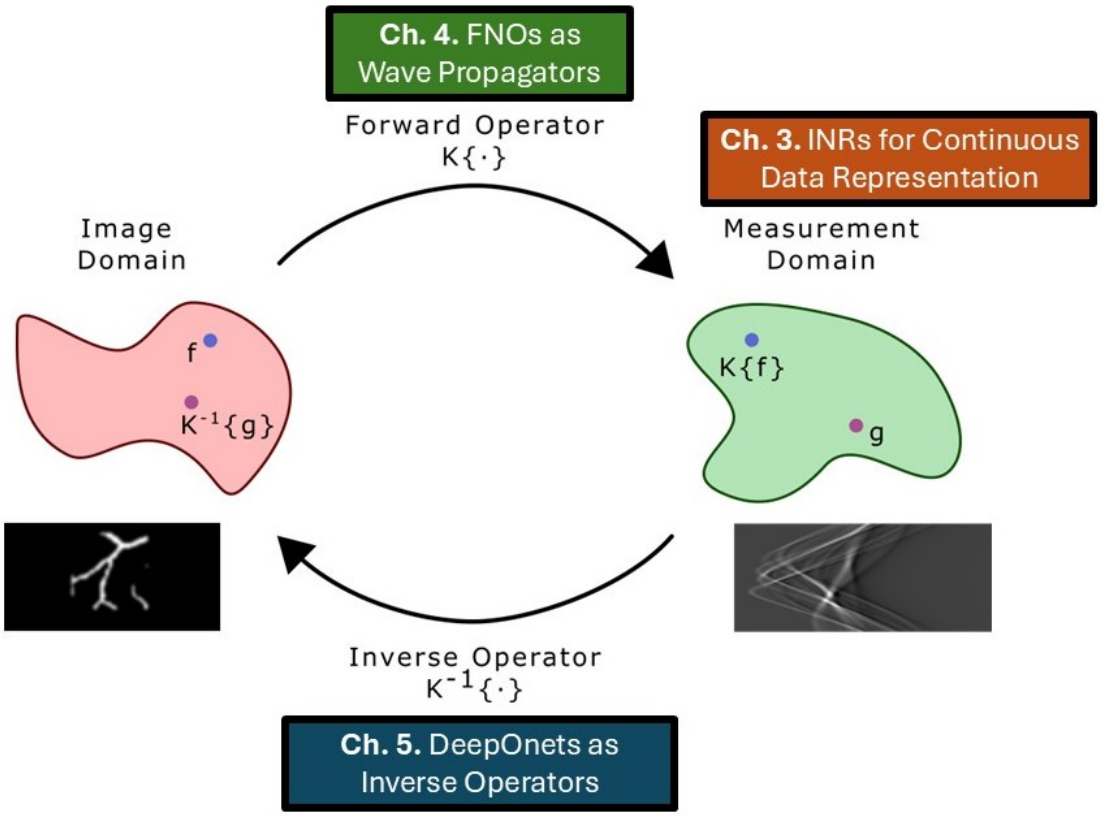


Figure 1.1: Model-based image reconstruction focuses on inverting a forward operator that maps objects from one domain to another, referred to here as the image domain and the measurement domain. The three main chapters of this thesis explore various techniques that intervene at different points in and between the two domains, as illustrated in the overview image.

Chapter 2

General Background

In this chapter, I will provide some basic background information necessary for understanding the rest of this report. The reader can skip the sections with which they are already familiar. I will begin with an introduction to the physical mechanisms fundamental to photoacoustic imaging. Next, I will discuss a specific photoacoustic imaging system, namely the 3rd generation photoacoustic mammoscope (PAM3). In the second half of this chapter, I will give a brief introduction to the use of continuous neural representations for scientific machine learning.

2.1 Photoacoustic Imaging

Imaging from coupled physics refers to imaging modalities that make use of one type of signal generated by another. These approaches often produce high-resolution, high-contrast images, unlike modalities that rely on a single type of physical signal, which typically provide either good resolution or good contrast, but not both [4]. An example of an imaging technique based on coupled physics is photoacoustic imaging.

Photoacoustic imaging is an imaging technique that can provide images of light absorbing structures. One application of this technique is breast cancer detection based on images of tumour blood vessels, which is possible thanks to the high optical absorption of haemoglobin for infrared wavelengths [5]. Images of these light absorbing structures can be formed by detecting ultrasound signals that are generated by these structures after pulsed illumination, an effect known as the photoacoustic effect, the physics of which will be discussed in subsection 2.1.1. Several different imaging architectures exist to detect these acoustic signals, one of which is the PAM3, which will be introduced in subsection 2.1.2. After detection, inversion based on the detected signals is necessary to obtain an image of the light absorbing structures. Solving this inverse problem is not an easy task, as will be further elaborated on in subsection 2.1.3.

2.1.1 Signal Generation through the Photoacoustic Effect

Photoacoustic imaging (PAI) is based on the detection of ultrasound (US) waves formed after illumination of an absorbing object with a short laser pulse, an effect known as the photoacoustic (PA) effect. When pulsed light is absorbed, absorbing molecules will convert the light into heat, which results in a local pressure increase through a process know as thermoelastic expansion. In the case of sufficiently short laser pulses (stress confinement [6]), the pressure increase, p_0, is given by[1]

$$p_0(r) = \Gamma(r)H(r), \tag{2.1}$$

where H, is the absorbed energy per unit volume due to light absorption, Γ is the dimensionless Grüneisen parameter and r is the spatial coordinate [7]. The heating function, H, is given by

$$H(r) = \mu_a(r)\Phi(r; \mu_a, \mu_s, g), \tag{2.2}$$

where μ_a is the absorption coefficient, μ_s is the scattering coefficient, g is the anisotropy factor and ϕ is the fluence distribution [5]. Note that equation 2.2 is highly nonlinear due to H being a product of μ_a and Φ, and μ_a also being an argument of Φ.

The two-step process of photoacoustic signal generation in human tissue starts with the optical transport problem described by the radiative transport equation [7]. Light at the surface of the tissue is transported throughout the tissue, which results in an inhomogeneous fluence distribution due to an inhomogeneous distribution of absorption coefficients, scattering coefficients and anisotropy factors. It is often tried to obtain a fluence distribution at the surface of the tissue that is as homogeneous as possible, because in this case the generated initial pressure distribution, even though not exact, gives a good approximation of the optical absorption coefficients in tissues [5]. The second part of the two-step process is the acoustic forward problem. The generated pressure, given by equation 2.1, propagates according to the wave equation. An overview of this two-stage signal generation process is shown in figure 2.1.

[1] Wavelength dependence of all variables will be ignored in this thesis.

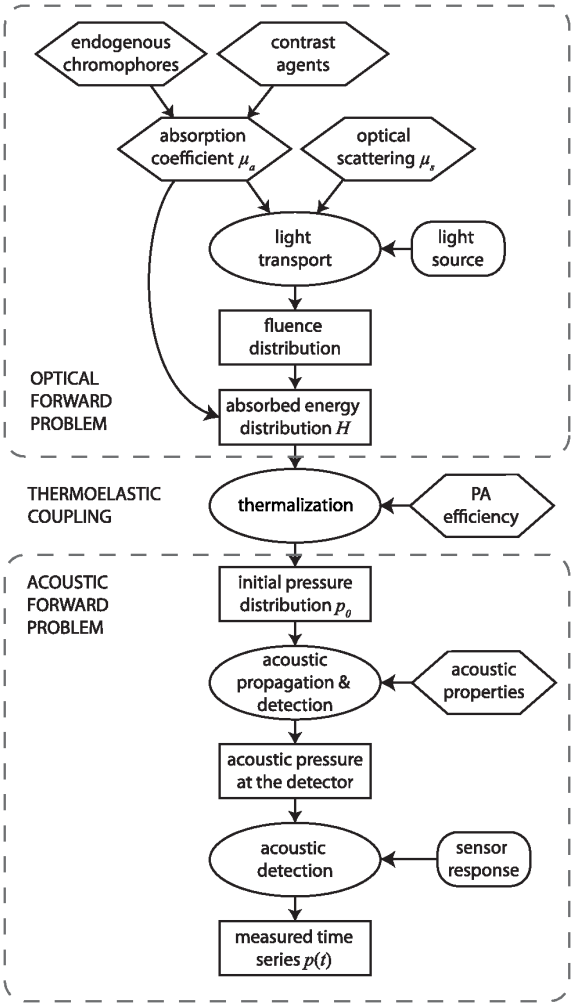


Figure 2.1: The forward problem resulting in photoacoustic signal generation consists of the optical forward problem followed by the acoustic forward problem [7].

2.1.2 The PAM3

Signals generated according to the process described in section 2.1.1 need to be detected to make image reconstruction possible. Several different photoacoustic imaging systems exist that can detect these signals, most of which use one of three different geometries: planar, cylindrical or spherical [5]. The imaging system that is studied in this thesis is the PAM3 [3] which is a breast imaging system.

The PAM3 consists of a hemispherical bowl containing 512 1 MHz US transducers (⌀3mm) and 40 optical fibres that provide a homogeneous illumination of a chosen wavelength on the surface of the breast (with a pulse repetition frequency of 10 Hz). The whole bowl with detectors can rotate in order to move the transducers to a new position. Due to the clever spiral positioning of the transducers, none of the measurements will be redundant. In addition to photoacoustic imaging, the PAM3 can also be used for ultrasound tomography, to obtaining speed of sound (SOS)

maps. The photoacoustic resolution of the system is 0.426mm and the resolution of the speed of sound maps is approximately 2mm [3]. An overview of the PAM3 is shown in figure 2.2.

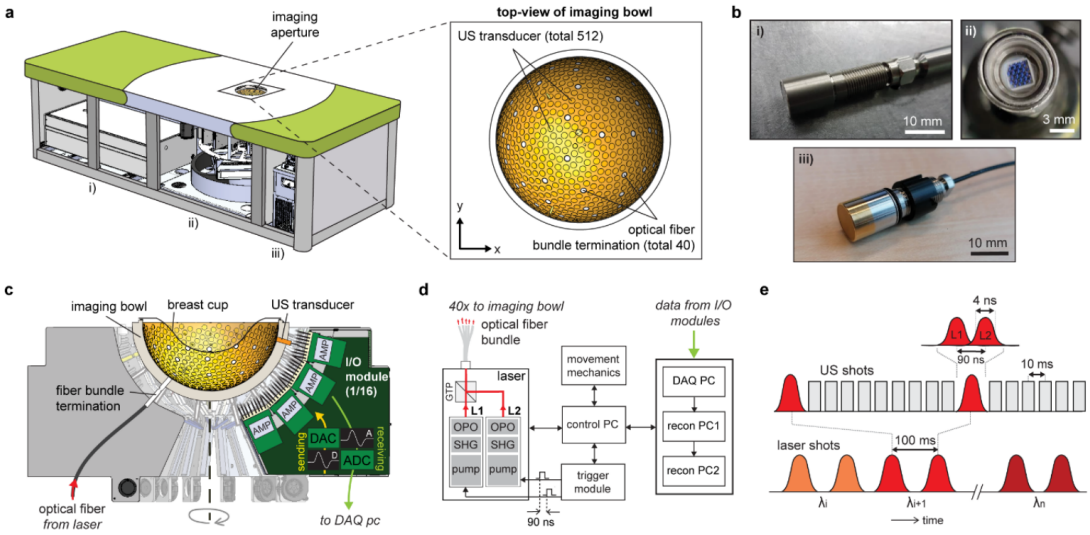


Figure 2.2: Overview of the PAM3. a) shows the bed-top on which a patient lies. The breast of the patient is placed in the hemispherical bowl which is lined with 512 US transducers and 40 optical fibres. b) shows a fibre bundle, microlens array on the fibre bundle and a transducer. c) shows a cross section of the system. d) shows a block diagram of the connections between different components of the PAM3. e) shows a graphical depiction of an imaging protocol. There is a minimum 100 ms between two laser pulses. During these 100 ms, US pulses are fired, which are used for the SOS reconstruction [3].

2.1.3 PAT Inverse Problem

After detection of photoacoustic signals, using a system like the one described in section 2.1.2, a reconstruction method is used to invert the forward problem as is shown abstractly in figure 2.3. To be more specific, the goal is to find the initial condition that propagated according to the wave equation and produced the pressure measurements at a discrete set of locations and for a discrete number of time steps. Relating this to figure 2.3, f is the initial pressure that is transformed to the measurement domain by a forward operator, K. The goal is to reconstruct this initial pressure by finding the operator K^{-1} or an approximation of K^{-1}.

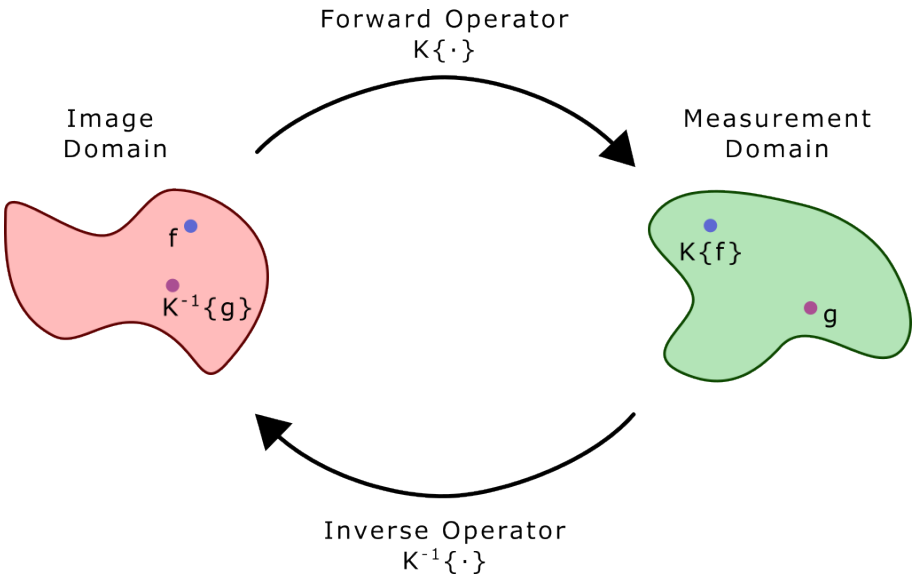


Figure 2.3: The general idea of inverse imaging problems is that an imaging operator, K, transforms an object or image, f, to a measurement domain, resulting in the measurement Kf. Often, we only have access to the data, g, in the measurement domain, and want to find the corresponding image, $K^{-1}g$, by applying an inverse operator, K^{-1}. In a simple and perfect world, we would have that $KK^{-1} = K^{-1}K = I$, so $K^{-1}Kf = f$, but often things are more complex.

I will now describe more precisely what the photoacoustic inverse problem is. The problem can be described in the following way (see also figure 2.4). Let $\Omega \subset \mathbb{R}^3$ denote the domain of interest, with spatial coordinates $r \in \Omega$, which is the hemispherical imaging bowl in which the breast is placed and let $\partial\Omega$ denote the boundary of Ω.

Let $t \in [0, T]$ denote the time since the laser pulse, which illuminates the tissue, has fired, where $0 < T \in \mathbb{R}^+$. T is the time it takes to perform the measurement. The moment a laser pulse is fired, a pressure distribution, $p_0(r) \in C_0^\infty(\Omega)$[2], is formed almost instantaneously [9], as explained in section 2.1.1.

p_0 travels according to the wave equation, where the medium is assumed to be inhomogeneous, given by [6]:

$$\left(\frac{1}{c^2(r)}\frac{\partial^2}{\partial t^2} - \nabla^2\right) p(r,t) = 0, \tag{2.3}$$

The speed of sound is often assumed to be a constant, but this assumption introduces artefacts in the reconstructed images if the SOS of sound is very inhomogeneous, like in of breast tissue where the speed of sound varies from 1400 to 1540 m/s[6].

[2]The assumption that $p_0 \in C_0^\infty(\Omega)$ is justified because there is always some heat diffusion present in the thermalisation process [8].

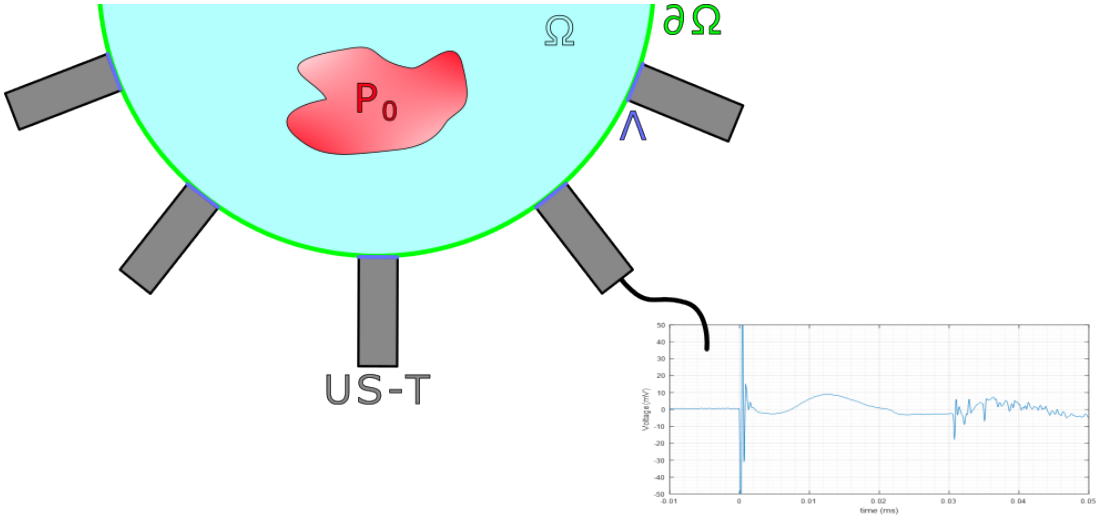


Figure 2.4: An initial pressure at $t = 0$, p_0, propagates according to the wave-equation. Ultrasound transducers (US-T) detect the pressures at the boundary of our domain Ω.

Detectors are placed at the boundary of Ω. Let $\Lambda \subset \partial\Omega$ denote the set of coordinates that make up all the detectors. The operator $\mathcal{P} : C_0^\infty(\Omega) \to C_0^\infty(\partial\Omega \times [0,T])$ propagates p_0 to the boundary $\partial\Omega$. An operator representing the detectors[3], $\mathcal{D}_k : C_0^\infty(\Lambda_k \times [0,T]) \to \mathbb{R}^{N_{det} \times N_t}$, transforms the pressures at Λ_k to discrete detected signals, where N_{det} is the number of detectors and N_t is the number of time samples. Figure 2.4 shows the detected signal for one of the transducers for a single laser pulse.

The inverse problem is [10]:

Find $p_0(r) \in C_0^\infty(\Omega)$ given the following PDE with initial and boundary conditions:

$$\begin{cases} \left(\dfrac{1}{c^2(r)}\dfrac{\partial^2}{\partial t^2} - \nabla^2\right) p(r,t) = 0, & t \geq 0,\ r \in \mathbb{R}^3 \\ p(r,t=0) = p_0(r), \quad \left.\dfrac{\partial p}{\partial t}\right|_{t=0} = 0 & \\ p(r,t)\Big|_\Lambda = \mathcal{D}\left\{\mathcal{P}\left\{p_0(r)\right\}\right\}^4, & r \in \Lambda,\ t \in [0,T] \end{cases} \tag{2.4}$$

This inverse problem would be uniquely solvable if Ω is surrounded with a smooth and closed detection surface and if we measure for a time long enough for the sound waves to have disappeared[10]. For practical reasons, a closed surface is rarely an option in biomedical applications, instead planar, cylindrical and hemispherical

[3]The operator $\mathcal{D}$ includes factors like integration over the detector surface area and convolution in time to account for the finite size and the time response of the detectors respectively.

[4]In real situations, the data will be noisy, but let's ignore that for now.

detection surfaces are often used [6]. Theoretically, a unique reconstruction is still possible in these cases, but the reconstruction process is unstable [10].

Many different reconstruction techniques exist. See for example [6] and [10] for a nice overview. Three common ways of solving the photoacoustic inverse problem [8] will be mentioned here :

1. **Inverse Spherical Radon Transform:** In the case of a constant speed of sound and with the absence of attenuation the signal can be projected back along arc's of a constant radius. In case of an inhomogeneous speed of sound, the signals are still projected back, but this time along arcs that have a constant time-distance, i.e. signals coming from points along this arc arrive at the same time at the detector.

2. **Time Reversal:** The pressures detected by the detectors are played back in time reversed order as Dirichlet boundary conditions. The pressure distribution at $t = 0$ is then an estimate of p_0. An advantage of this technique is that it can deal with inhomogeneous acoustic properties, like an inhomogeneous SOS.

3. **Variational Formulation:** The reconstruction problem can be rewritten to the form $\arg\min_{p_0} \left\{ \frac{1}{2} \| \mathcal{D} \{ \mathcal{P} \{ p_0(r) \} \} - f \| + \lambda \mathcal{J}(p_0) \right\}$, where f is the detected noisy data, $\lambda > 0$ is the regularization parameter and $\mathcal{J}$ is a regularization functional. The variational formulation is especially useful in cases of incomplete data because of the option to add a regularizer which includes prior knowledge of the solution p_0.

2.2 Continuous Neural Representations

The first mathematical models of artificial neurons were already introduced in 1943 [11]. Still, it took many decades, and several deep learning winters [12], before multiple artificial neurons were combined to form multi-layered deep learning networks that exhibited significant learning capabilities [11]. Since then, deep learning models have permeated many different fields of science and everyday life. One scientific field that has greatly benefited from deep learning is the study of partial differential equations, which will be the main focus of this report.

Not all deep learning methods can be used for all deep learning applications: One size does not fit all. A good example of this is that some of the AI models which have shown awe-inspiring results for computer vision tasks, like transformers and convolution neural networks (CNNs), are unsuitable for scientific modelling [13]. This is because scientific modelling deals with real-world continuous objects, unlike many other deep learning fields that solely deal with discrete objects like images and language. This means new methods had to be developed to deal with the continuity of functions and operators in real-world physical systems. In the rest of this section, I will explain how both functions as well as operators can be represented with neural

networks in a continuous way. Basic knowledge of deep learning is assumed. Readers not familiar with deep learning are referred to [14] for a basic introduction.

2.2.1 Continuous Neural Function Representations

Functions and operators are ubiquitous through all fields of science and engineering, and are the two important objects when studying partial differential equations (PDE). No standard definitions exist in the literature to distinguish between functions and operators. However, for clarity and ease of explanation, I will adopt the following definitions for functions and operators in this thesis:

Definition 2.1. A function is any mapping that takes as an input an element $x \in \mathcal{D} \subseteq \mathbb{R}^{N_{in}}$, called a **coordinate**, and assigns to this element a single unique element $f(x) \in \mathbb{R}$ or a finite vector of length d, $\mathbf{f}(x) \in \mathbb{R}^d$. $\mathcal{D}$ is called the domain of the function.

Example 2.2. The function p takes as an input the tuple $(x, y, t) \in \mathbb{R}^3$, and outputs the pressure value at the spatio-temporal coordinate, or both the pressure and the speed of sound.

Definition 2.3. An operator is any mapping that takes as an input an element $f \in \mathcal{F}$ and outputs a single unique element $g \in \mathcal{G}$, with $\mathcal{F}$ and $\mathcal{G}$ being function spaces, i.e. a vector space whose elements are functions.

Example 2.4. The operator $\mathcal{F}$, called the 1-dimensional Fourier transform operator, takes as an input a function $f \in L^1(\mathbb{R})$ and outputs a function $\mathcal{F}\{f\} \in C(\mathbb{R})$, with $L^1(\mathbb{R})$ the first order lesbesgue space, and $C(\mathbb{R})$ the space of continuous functions, both containing functions whose domain is $\mathbb{R}$ [15].

Example 2.5. Operators can be used to find the solution of a PDE. An operator can take the initial and boundary data, represented by functions, as an input and output the solution to the PDE. In the case of the wave equation, a solution operator could take the initial pressure as an input and output the pressure as a function of space and time.

If we want to model real-world phenomena with computers, we need to come up with a discrete way of representing functions and operators. One way of discretizing a function is sampling the function for a certain set of coordinates. Off course, this method has the downside that we could lose information about the function values at coordinates that lie in between the sampled coordinates. However, if we have some prior knowledge about the function we want to represent, for example that the function is a third degree polynomial of the form $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$, then all we need to store in the computer is the discrete vector (a_0, a_1, a_2, a_3). So, storing the function coefficient allows us to represent a continuous object in a discretized way, and this function can be evaluated on all coordinates in the domain of the function.

Imagine that we have a situation where we are given a set of N coordinates and their corresponding function values, $\{x_i, f(x_i)\}_{i=1}^{N}$, and want to obtain an expression for the function, $f(\cdot)$, that represents this data. We can now distinguish two different scenarios, either we have a model for the process that generated the data, and thus know the form of the function $f(\cdot)$, up to a set of unknown parameters, or we have no knowledge about the function $f(\cdot)$ and the process through which the data was generated has to be treated as a black box. In case of the first scenarios, techniques like least squares fitting, can be used to find the optimal set of parameters that describe the data. The second scenario is more challenging, but can be solved using neural networks since any (Borel measurable) function can be approximated using neural networks [16]. In this case, the function is represented by a neural network, like a fully connected network, that takes a coordinate, x, as an input, and the output is the function value, $f(x)$. A fully connected neural network consists of layers where each neuron is connected to every neuron in the previous layer. Let the input be a vector $\mathbf{x} \in \mathbb{R}^d$, and the output of the network be $f(\mathbf{x}) \in \mathbb{R}^m$. A fully connected neural network with L layers can be written compactly as:

$$\mathcal{N}(\mathbf{x}) = A_L \left(\sigma \left(A_{L-1} \left(\cdots \sigma \left(A_1 \mathbf{x} + \mathbf{b}_1\right) \cdots\right) + \mathbf{b}_{L-1}\right)\right) + \mathbf{b}_L \tag{2.5}$$

where:

- Each $A_l \in \mathbb{R}^{n_l \times n_{l-1}}$ is a weight matrix for layer l,
- $\mathbf{b}_l \in \mathbb{R}^{n_l}$ is the corresponding bias,
- σ is a non-linear activation function (e.g., ReLU),
- $\mathcal{N}(\mathbf{x}) = f(\mathbf{x}) \in \mathbb{R}^m$ is the final network output.

An important network to mention is the Sinusoidal Representation Network (SIREN), which uses the sine function as activation functions: $\sigma(z) = \sin(\omega z)$, where the hyperparameter ω is a frequency parameter. This type of network will be used later in the report due to its excellent ability to represent data [17].

Similar to the case of representing a continuous polynomial function by saving a discrete set of polynomial coefficients, any continuous function can be represented by a neural network represented by the discrete set of weights and biases that describe the network. The weights and biases that best describe the data are found by optimizing some kind of loss function. With this self-supervising deep learning method a function, represented by a neural network, is fitted to data, so there is no 'learning' from data.

Several different names are used for these neural function representations, like coordinate-based neural networks [18], neural fields [19], neural radiance fields (NeRFs) [20], implicit neural representations (INRs) [21] and physics informed neural networks (PINNs) [22]. In this thesis, the term Neural Function (NF) will be used to refer to these coordinate-based networks. These NFs can be evaluated at any coordinate. Therefore, it is often stated that neural functions offer an infinite resolution

[18]. I will go deeper into what this means exactly in chapter 3.

2.2.2 Continuous Neural Operator Representations

Just like functions, real-world operators are continuous objects, i.e. they are maps between function spaces, like the integral from example 2.4. Surrogates for operators, represented by neural networks, are termed neural operators. Not every type of neural network is considered a neural operator. Something that differentiated a neural operator from other networks like transformers or CNNs is that they can generalize to different discretizations due to their discretization invariance [13].

It is easy to see that the types of networks used for neural functions are not considered neural operators. Those networks take the coordinates as an input and represent functions instead of taking the the whole function as an input.

Some of the neural networks that are neural operators are the Fourier neural operator (FNO)[23] and the deep operator network (DeepOnet)[24] be described in the following subsections.

The general idea of operator learning is that we are given pairs of data, $\{f_i, g_i\}_{i=1}^N$, just like we saw before for the neural functions, but in this case f_i and g_i are both functions living in function spaces, and they are connected through an operator $\mathcal{A}$ such that $\mathcal{A}\{f\} = g$ [25]. The goal of operator learning is to approximate the operator $\mathcal{A}$ using a neural network $\mathcal{N}$. What follows now is a brief introduction to two different neural operator architectures.

2.2.2.1 Deep Operator Network

We have seen before how neural function can be used to overfit to data in a self-supervised manner in section 2.2.1, resulting in a continuous function representation. One example is solving the wave equation for $p(r, t)$ given an initial condition $p_0(r)$. For each initial condition, the NF has to be trained to find the solution $p(r, t)$ making the technique slow, and unable to learn from patterns in the data. Several different groups have come up with ideas to create adaptive neural functions, where the neural function is modified based on sensor data. One possibility is the use of conditional neural fields [26], which make use of a so-called hypernetwork [27] which outputs the parameters of a second network, which is the actual NF. Another network architecture is the deep operator network (DeepOnet) introduced by Lu et al. [24], which is usually termed a neural operator, but can be considered as a conditional neural field if we realize that a weighted sum of NFs results in a new NF:

$$\mathcal{D}(x) = \sum_{k=1}^{p} b_k \mathcal{T}_k(x)$$

where the NF, $\mathcal{D}(\cdot)$, is formed by a sum of p networks $\mathcal{T}_k$, which can be seen

as basis functions, weighted by the coefficients b_k. The idea behind DeepOnets is to condition the basis coefficients based on the values of sensor data, to create an adaptive NF. The value of the basis coefficients is determined by another set of p neural network with as an input some sensor data, u, given on a finite fixed set of locations, $x_1, \ldots, x_m$. This results in the following architecture [24]:

$$\mathcal{D}\{f\}(x) = \sum_{k=1}^{p} \mathcal{B}_k\{u(x_1), u(x_2), \ldots, u(x_m)\}\, \mathcal{T}_k(x) \tag{2.6}$$

where $\mathcal{B}\{\cdot\}$ is called the branch network, which gives the basis coefficient, and $\mathcal{T}_k(\cdot)$ is called the trunk network, and provides the continuous basis functions. A problem here is that using p branch and trunk networks gives problems with scaling this architecture. Therefore Lu et al. came up with the idea of using for both the branch and the trunk networks, a single network that output a vector of length p, thus merging all the branch networks and trunk network into a single network [24]. In this case, $\mathcal{B}_k$ in equation 2.6 refers to the kth output of the branch network [5]. See figure 2.5

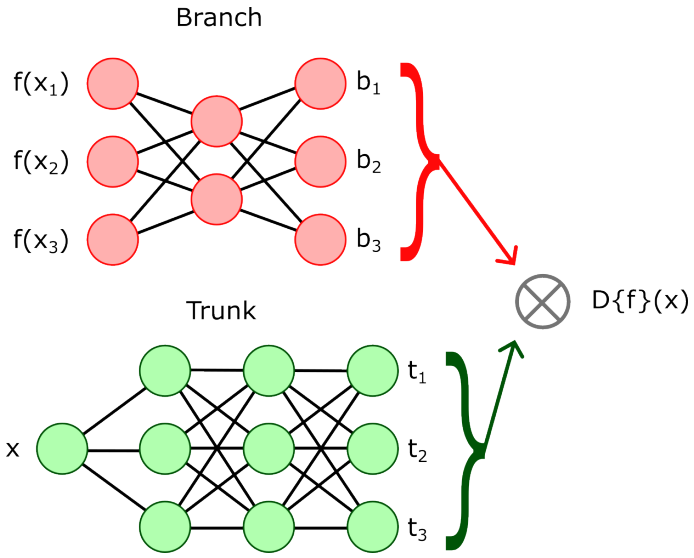


Figure 2.5: The DeepOnet architecture consists of two networks. Shown are two fully connected networks, where the branch network takes sensor data at discrete fixed sensor locations, $f(x_k)$, as an input, and the trunk takes the coordinate at which the ouptut function is evaluated as an input. The output of the DeepOnet is formed by the dot product of the output vectors of both networks, i.e. $\mathcal{D}\{f\}(x) = \sum_{k=1}^{3} \mathcal{B}_k\{f\}\, \mathcal{T}_k(x) = \sum_{k=1}^{3} b_k t_k$.

Lu et al. have proven that the DeepOnet architecture is a universal approximator for operators [24] and it has been shown experimentally that many different

[5] This is similar to the conditional neural fields introduced before, with the branch network being a hypernetwork that only outputs the network parameters of a final linear layer.

operators can be approximated using DeepOnets with great accuracy [28].

2.2.2.2 Fourier Neural Operator

Another very popular neural operator network besides the DeepOnet from the previous section, is the Fourier neural operator (FNO) [23]. The FNO is inspired directly from equation 2.5, but instead of using discrete objects and matrix multiplication, functions and kernel integration are used [25], resulting in network layers of the form:

$$u_{i+1}(x) = \sigma\left(\int_{\Omega_i} K^{(i)}(x,y)u_i(y)dy + b_i(x)\right), \quad x \in \Omega_{i+1}, \tag{2.7}$$

where we are trying to learn the bias functions, b_i, and the kernels, $K^{(i)}$. Calculating the integrals and approximating the kernels $K^{(i)}$ is computationally expensive. To overcome these problems, the FNO makes two assumptions, first of all that the input and output domain are d-dimensional toruses, $\mathbb{T}^d$, and that the input and output functions are sampled on a fixed and equally spaced grid, allowing the use of the fast Fourier transform (FFT) for evaluating convolutions and secondly that the kernels $K^{(i)}$ are translation invariant, i.e. $K^{(i)}(x,y) = k^i(x-y)$. The kernel integration in equation 2.7 can thus be written as:

$$\int_{\Omega_i} k^{(i)}(x-y)u_i\,dy = \mathcal{F}^{-1}\left\{\mathcal{F}\{k^{(i)}\}\mathcal{F}\{u_i\}\right\}(x) = \mathcal{F}^{-1}\left\{\mathcal{R}\cdot\mathcal{F}\left\{u_i\right\}\right\}(x), \quad x \in \Omega_i. \tag{2.8}$$

To make use of the FFT, all the function in equation 2.8 have to be discretized:

- $u_i \in \mathbb{R}^{C\times H\times W}$, so u has C channels, each of size $H \times W$.
- $\mathcal{F}\left\{\cdot\right\}$ and $\mathcal{F}^{-1}\left\{\cdot\right\}$ are applied channel-wise.
- $\mathcal{R} \in \mathbb{C}^{C\times C\times k_{H,max}\times k_{W,max}}$, and the product $\mathcal{R}\cdot\mathcal{F}\left\{u_i\right\}$ is a tensor product, which is the same as applying a matrix of size $C\times C$ to each of the Fourier coefficients of $\mathcal{F}\left\{u_i\right\}$ separately. $k_{H,max} \times k_{W,max}$ are chosen maximum frequencies, the rest of the frequencies are set to zero.

So, instead of learning the parameters of k^i in the coordinate space, we learn the parameters of $\mathcal{R}$ directly in Fourier space. The FNO consists of multiple of these Fourier layers. The full FNO architecture is shown in figure 2.6, where the learnable parameters are the coefficients of the tensor $\mathcal{R}$ and the vector b [29].

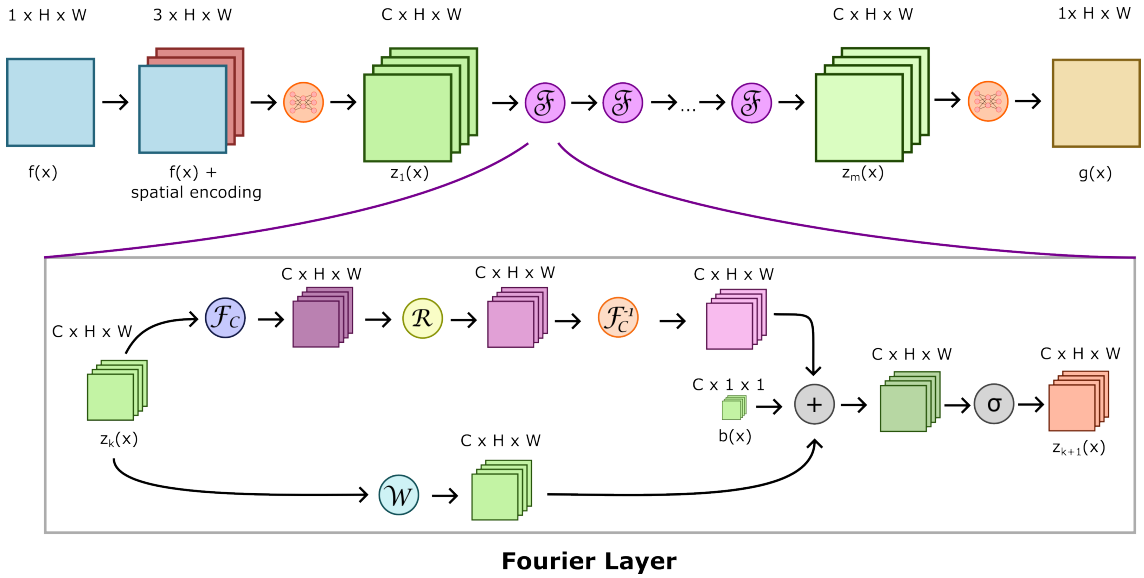


Figure 2.6: This figure shows the FNO architecture for a 2D input of size H by W and 1 channel. Above each object the size of the array is written. An input function is combined with features, like spatial encoding, before going through a channel-wise FCN that lifts the input to a higher dimensional representation with C channels. Next, the input goes through several Fourier layers, before a second channel-wise FCN projects the function back to the desired dimension. The Fourier layer consists of two paths. The top path takes the input and applies in order a channel-wise Fourier transform, $\mathcal{F}_C$, a filter, $\mathcal{R}$, and a channel-wise inverse Fourier transform $\mathcal{F}_C^{-1}$. The bottom path mixes the channels by applying a linear layer in the spatial domain. Both paths are combined, a bias, $b(x)$ is added, and an activation function, σ, is applied element-wise to produce the output of the Fourier layer.

Chapter 3

Neural Functions in the Measurement Domain

Photoacoustic image reconstruction is the process of estimating the object that generated the measured data, typically recorded at the boundary of the domain. In an idealized scenario with perfect measurements, theoretical results have shown that complete enclosure of the object by the detection surface is not strictly necessary for reconstruction. In fact, under certain conditions, it is possible to uniquely reconstruct an object from measurements taken over only a small portion of the surrounding surface [10]. So why does the PAM3, introduced in section 2.1.2, still collect so many measurements? The first reason is that real-world conditions are far from ideal. Photoacoustic signals are extremely weak and highly susceptible to noise. Secondly, even in a noise-free setting, reconstructions based on data from only a small part of the detection surface are extremely unstable, making practical reconstruction virtually impossible [10].

Therefore, many measurements are needed to obtain a decent reconstruction quality with the PAM3. However, acquiring more measurements takes time, and for practical reasons, there is a limit to how long a scan can take. To address this, several techniques have been proposed to reduce the number of required measurements. For example, [30] discusses a method called compressed sensing, which leverages inherent sparsity to reconstruct objects from far fewer measurements than what the Shannon-Nyquist sampling theorem would suggest. Another approach, described in [31], uses deep learning to inpaint missing data based on the learned distribution of commonly scanned images.

The neural functions, introduced in section 2.2.1, accept coordinates as an input, and output the function value at the input coordinate and therefore it is often stated that they offer a continuous view, so offering an infinite resolution [19]. This inspired Sun et al. [18] to use these networks for data inpainting in the context of X-ray computed tomography, which relies on the implicit regularization properties of NFs. There is a lot of overlap between the reconstruction methods used for x-ray computed tomography (CT) and photoacoustic tomography, so the use of neural functions for

photoacoustic tomography could possibly also improve the reconstruction results.

The central question of this chapter is whether NFs can be used for photoacoustic data inpainting for the PAM3 to improve the reconstruction quality by reducing the sparsity in the measurements. Similar reconstruction methods are used in photoacoustic imaging and CT imaging, like the inverse (spherical) Radon transform. The planar CT geometry is easier to study than the spherical geometry used for photoacoustic imaging. That is why the case of CT reconstruction will play an important role in this chapter. The first section of this chapter, section 3.1, discusses data inpainting using linear interpolation. Linear interpolation is fast and intuitive to understand and is a good method to compare the NFs to which offer a continuous representation of the measured data and form the subject of section 3.2. The chapter is finished with a discussion and conclusion in section 3.3.

3.1 Sinogram Inpainting Using Linear Interpolation

This section starts by introducing the mathematical forward and inverse operators used for planar tomographic imaging. Next, an analytical description follows for how linear interpolation in the sinogram domain affects the reconstruction result for the case of the 2D planar Radon transform.

The forward operator that is used for CT imaging is the Radon transform. The Radon transform is a well-studied operator, see for example [32] and [33], and many theoretical results are available. The Radon transform is often defined as a map between Schwartz spaces, because many of the interesting properties of the Radon transform are easy to proof when the restriction is made to Schwartz spaces, even though many of the results can be extended to larger spaces [34].

The Schwartz space is defined as

Definition 3.1 (Schwartz Space [33])**.** The Schwartz space, also referred to as the space of rapidly decreasing functions, is the function space

$$\mathcal{S}(\mathbb{R}^n) = \left\{ f \in \mathcal{C}^\infty(\mathbb{R}^n) | x^\alpha \partial^\beta f(x) \in \mathcal{C}_0(\mathbb{R}^n) \quad \forall \alpha, \beta \in \mathbb{Z}_+^n \right\}$$

where $\mathcal{C}^\infty$ is the space of continuous functions that are infinitely differentiable, $\mathcal{C}_0$ is the space of continuous functions that go to 0 for $|x| \to \infty$ and the multi-index notation is used, i.e. $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$ and $\partial^\beta = \partial_1^{\beta_1} \partial_2^{\beta_2} \dots \partial_n^{\beta_n}$.

The forward operator that describes the CT imaging problem is the Radon transform operator, which is defined by taking integrals of an object along straight lines:

Definition 3.2 (Radon Transform [32])**.** The 2D Radon transform operator, $\mathcal{R} : \mathcal{S}(\mathbb{R}^2) \to \mathcal{S}(\mathbb{R} \times \mathbb{S}^1)$ of an object $f \in \mathcal{S}(\mathbb{R}^2)$ is given by

$$g(l, \theta) = \mathcal{R}\{f\}(l, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos\theta + y \sin\theta - l) \, dx \, dy,$$

where $g \in \mathcal{S}(\mathbb{R} \times \mathbb{S}^1)$ is called the sinogram data , $l \in \mathbb{R}$ is the detector position and $\theta \in \mathbb{S}$ is the angle of the detector array.

Note that g is an even function, i.e., $g(l, \theta) = g(-l, -\theta)$, which means that the data obtained by rotating the detectors over more than 180 degrees is redundant.

A graphical depiction of definition 3.2 is given in figure 3.1. The sinogram data, g, is obtained by integrating the object, f, over parallel lines for different angles, θ, and detector positions, l.

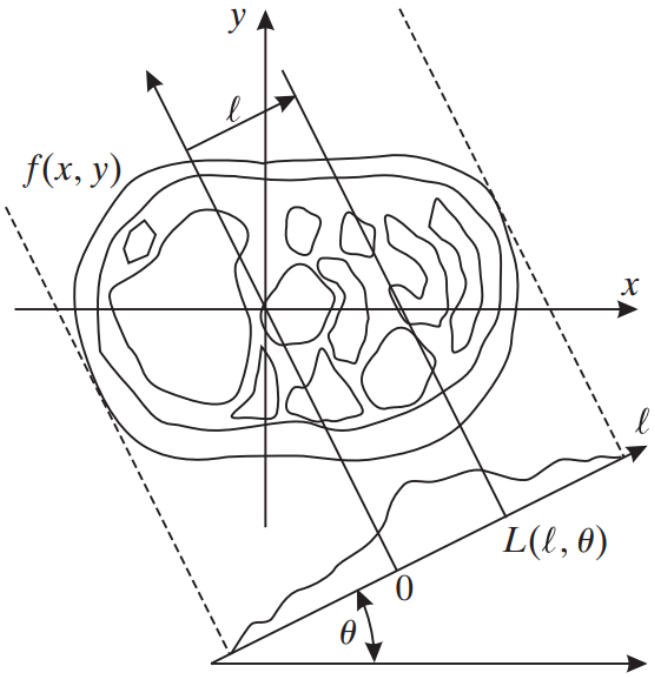


Figure 3.1: Geometry of the Radon transform. Due to the sifting property of the delta function, integration in definition 3.2 is performed over lines, $L(l, \theta)$ (the notation $g(l, \theta)$ is used in this chapter), whose direction is perpendicular to θ and is shifted away from the origin in a direction perpendicular to the line by a distance l [35].

One way of trying to recover the original object, f, is by smearing out all the measurement data over parallel lines, a technique known as backprojection.

Definition 3.3 (Backprojection [32])**.** The 2D backprojection operator, $\mathcal{R}^* : \mathcal{S}(\mathbb{R} \times \mathbb{S}^1) \to \mathcal{S}(\mathbb{R}^2)$ of sinogram data, $g \in \mathcal{S}(\mathbb{R} \times \mathbb{S}^1)$, is given by

$$f_b(x, y) = \mathcal{R}^* \{g\} (x, y) = \int_0^{2\pi} g(x \cos\theta + y \sin\theta, \theta)\, d\theta,$$

where $f_b \in \mathcal{S}(\mathbb{R}^2)$ is called the back-projected object and $\mathcal{R}^*$ is the adjoint of $\mathcal{R}$.

A downside of image reconstruction using backprojection is that low-frequency components are oversampled. To compensate for this, the convolution backprojection algorithm was developed, which gives the following reconstruction

Definition 3.4 (Filtered Backprojection [35])**.** The 2D filtered backprojection operator, $\mathcal{R}^{fb} : \mathcal{S}(\mathbb{R} \times \mathbb{S}^1) \to \mathcal{S}(\mathbb{R}^2)$ of sinogram data, $g \in \mathcal{S}(\mathbb{R} \times \mathbb{S}^1)$, is given by

$$f_{fb}(x, y) = \mathcal{R}^{fb}\{g\}(x, y) = \int_0^{2\pi} [c(l) * g(l, \theta)]_{l=x\cos\theta+y\sin\theta} \, d\theta,$$

where $f_{fb} \in \mathcal{S}(\mathbb{R}^2)$ is called the filtered back-projected object and $c(l)$ is a filter that compensates for the enhanced low-frequency components, whose exact form is not important for now. Note that for $c(l) = \delta(l)$, we recover the backprojection summation as given in definition 3.3.

Having introduced the forward and inverse operators used in tomographic image reconstruction above, I will now describe analytically what happens in the image domain if the missing parts of the sinogram are filled using linear interpolation.

In definition 3.2 the assumption is made that we have an infinite number of detectors, i.e. $l \in \mathbb{R}$, and measure for an infinite number of angles, i.e. $\theta \in \mathbb{S}^1$. Experimentally, we are always dealing with a discrete number of angles and a discrete number of detectors. However, the assumption will be made here that there is an infinite number of detectors and a discrete number of angles for which a measurement is performed. The reason for this is to stay closer to the photoacoustic imaging geometry of the PAM3. For the PAM3 the sampling in time is high enough to capture all the frequency content in the detected signals, and for the analogy to the planar Radon transform used for CT imaging, this corresponds to an infinite number of detectors.

Let's assume that we have an infinite number of detectors, so $l \in \mathbb{R}$, and the measurements are performed for the discrete angles $\theta \in [0, \Delta\theta, 2\Delta\theta, \ldots, \pi - \Delta\theta]$, where the angular spacing is given by $\Delta\theta = \frac{\pi}{N_\theta}$, with $N_\theta \in \mathbb{N}$ being the number of detector angles for which a measurement is performed. In this discrete case, an approximation of the object to be reconstructed, f^*, is given by

$$f^*(x, y) = \sum_{n=0}^{N_\theta - 1} \Delta\theta \, [c(l) * g(l, n\Delta\theta)]_{l=x\cos\theta+y\sin\theta} \, . \tag{3.1}$$

The quality of the reconstructed object can be improved by using a finer sampling of the angles. The question is whether it is possible to reduce the sparsity of measurements artificially using interpolation techniques, to improve the reconstruction quality. For example, the number of measurements can be doubled using linear interpolation. The interpolated measurements at the angles $\theta_{intp} \in [\Delta\theta/2, 3\Delta\theta/2, \ldots, \pi - \Delta\theta/2]$ are then given by

$$g_{intp}(l, \frac{2n-1}{2}\Delta\theta) = \frac{g(l, (n-1)\Delta\theta) + g(l, n\Delta\theta)}{2}, \qquad \text{for} \quad n = 1, \ldots, N_\theta. \tag{3.2}$$

The effect on the reconstruction using the inpainted sinogram data, for which the number of angles for which a measurement is performed is increased by a factor of 2, is shown in appendix B. A more interesting case of filling the whole sinogram, not limited to an increase of a factor 2 but an arbitrary number of measurements, will be discussed here. First the follwing definitions need to be introduced.

Definition 3.5. Given two projection measurements, $g(l, \theta_1)$ and $g(l, \theta_2)$, for the neighbouring measurement angles θ_1 and θ_2 respectively, a new linearly interpolated measurement at an intermediate interpolated angle $\theta_{intp} = (1 - \lambda)\,\theta_1 + \lambda\theta_2$ is given by $g(l, \theta_{intp}) = (1 - \lambda)\,g(l, \theta_1) + \lambda g(l, \theta_2)$, with $\lambda \in [0, 1]$.

Definition 3.6. The rotation operator $\mathcal{R}_{\Delta\theta}$ rotates, in a clockwise direction, a function, in polar coordinates, by an angle $\Delta\theta$, i.e.

$$\mathcal{R}_{\Delta\theta}\left\{f\right\}(r, \theta) = f(r, \theta - \Delta\theta),$$

where $r \in \mathbb{R}^+$ is the distance from the origin and $\theta \in \mathbb{S}^1$ is the angle.

Definition 3.7. The triangular function, also known as the hat function, is defined as

$$\Lambda(x) = \begin{cases} 1 - |x| & \text{if } |x| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3.8. The indicator function is defined as

$$\mathbb{1}_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3.9. Given two functions, $f, g \in \mathcal{S}\left\{\mathbb{R}^+ \times \mathbb{S}\right\}$, where we are using polar coordinates, the angular convolution, denoted by $*_\theta$, is defined as

$$\left[f *_\theta g\right](r, \theta) = \int_0^{2\pi} f(r, \tau) g(r, \theta - \tau)\, d\tau.$$

Proposition 3.10. *[36] Let $f \in \mathcal{S}(\mathbb{R}^n)$, then*

$$\mathcal{F}\left\{\mathcal{R}_\theta\left\{f\right\}\right\} = \mathcal{R}_\theta\left\{\mathcal{F}\left\{f\right\}\right\}.$$

Let's now say that we have used filtered back projection using the sinogram data, $g(l, \theta)$, for $l \in \mathbb{R}$, and discrete angles $\theta \in [0, \Delta\theta, 2\Delta\theta, \ldots, \pi - \Delta\theta]$, where the angular spacing is given by $\Delta\theta = \frac{\pi}{N_\theta}$, with $N_\theta \in \mathbb{N}$ being the number of detector angles for which a measurement is performed, to obtain an approximation $f^*(x, y)$, of the object $f(x, y)$. The relation between a reconstruction that is obtained after artificially filling the empty parts of the sinogram using linear interpolation according to definition 3.5, denoted by $f_{intp,\infty}$, and the sparse reconstruction, f^*, is given by proposition 3.11[6].

[6]The convolution kernel from definition 3.4 is ignored in the following proof to simplify notation, but the proof can be extended easily thanks to the linearity of the convolution operator.

Proposition 3.11. *The backprojected reconstruction based on sparse data and the reconstruction based on inpainted data according to definition 3.5, under the same circumstances as defined above, are related to each other by:*

$$f_{intp,\infty}(r,\theta) = f^*(r,\theta) *_\theta \frac{\Lambda\left(\frac{\theta}{\Delta\theta}\right)}{\Delta\theta}.$$

Proof. Knowing that the interpolated data is defined according to definition 3.5, and using the Fourier slice theorem (see appendix B), it is easy to see that the following relations holds[7].

$$\mathcal{F}\left\{f_{intp,\infty}\right\}(\rho,\theta) = \int_0^1 (1-\lambda)\mathcal{R}_{\lambda\Delta\theta}\left\{\mathcal{F}\left\{f^*\right\}\right\}(\rho,\theta) + \lambda\mathcal{R}_{-(1-\lambda)\Delta\theta}\left\{\mathcal{F}\left\{f^*\right\}\right\}(\rho,\theta)\,d\lambda$$

Using proposition 3.10, followed by applying the inverse Fourier transform, we get

$$\begin{aligned}
f_{intp,\infty}(r,\theta) &= \int_0^1 (1-\lambda)\mathcal{R}_{\lambda\Delta\theta}\left\{f^*\right\}(r,\theta) + \lambda\mathcal{R}_{-(1-\lambda)\Delta\theta}\left\{f^*\right\}(r,\theta)\,d\lambda \\
&= \int_0^1 (1-\lambda)f^*(r,\theta-\lambda\Delta\theta) + \lambda f^*(r,\theta+(1-\lambda)\Delta\theta)\,d\lambda \\
&= \int_0^1 (1-\lambda)f^*(r,\theta-\lambda\Delta\theta)\,d\lambda + \int_0^1 \lambda f^*(r,\theta+(1-\lambda)\Delta\theta)\,d\lambda \\
&= \int_{-1}^0 (1-\lambda)f^*(r,\theta-\lambda\Delta\theta)\,d\lambda + \int_{-1}^0 (\lambda+1)f^*(r,\theta-\lambda\Delta\theta)\,d\lambda \\
&= \int_{-1}^1 \mathbb{1}_{[0,1]}(\lambda)(1-\lambda)f^*(r,\theta-\lambda\Delta\theta) + \mathbb{1}_{[-1,0]}(\lambda)(\lambda+1)f^*(r,\theta-\lambda\Delta\theta)\,d\lambda \\
&= \int_{-1}^1 f^*(r,\theta-\lambda\Delta\theta)\left(\mathbb{1}_{[0,1]}(\lambda)(1-\lambda) + \mathbb{1}_{[-1,0]}(\lambda)(\lambda+1)\right)\,d\lambda \\
&= \int_{-\infty}^{\infty} f^*(r,\theta-\lambda\Delta\theta)\Lambda(\lambda)\,d\lambda \\
&= \frac{1}{\Delta\theta}\int_{-\infty}^{\infty} f^*(r,\tau)\Lambda\left(\frac{\theta-\tau}{\Delta\theta}\right)\,d\tau = \frac{1}{\Delta\theta}\left(f^*(r,\theta) *_\theta \Lambda\left(\frac{\theta}{\Delta\theta}\right)\right)
\end{aligned}$$

■

To see what the results given by proposition 3.11 means, a quick experiment is performed. First, assume we have an object whose sparse reconstruction can be written as $f^*(r,\theta) = f^*(r)\delta(\theta)$. According to proposition 3.11, the resulting interpolated reconstructed is then given by $f_{intp,\infty}(r,\theta) = f^*(r)\frac{\Lambda\left(\frac{\theta}{\Delta\theta}\right)}{\Delta\theta}$.

For the experiment, an object which has the value of 1 on the x- and y-axis and zero else is used, and is shown in figure 3.2. A measurement was performed

[7] If the reader does not see why this holds, see the proof of proposition B.2 in appendix B.

for 30 equally spaced angles between 0 and 180 degrees. The sinogram was filled by increasing the number of measurement angles by a factor of 100 using linear interpolation. The result is shown in figure 3.2. A case with Gaussian noise, with an amplitude of of 5% of the maximum value in the image, added to the measurement is also shown.

It can be seen how linear interpolation in the sinogram domain results in a convolution in the image domain along the angular direction. The shape of the convolution kernel, the hat function, is clearly visible. and the shape of the convolution kernel is visible in the reconstruction of the crossed-line phantom.

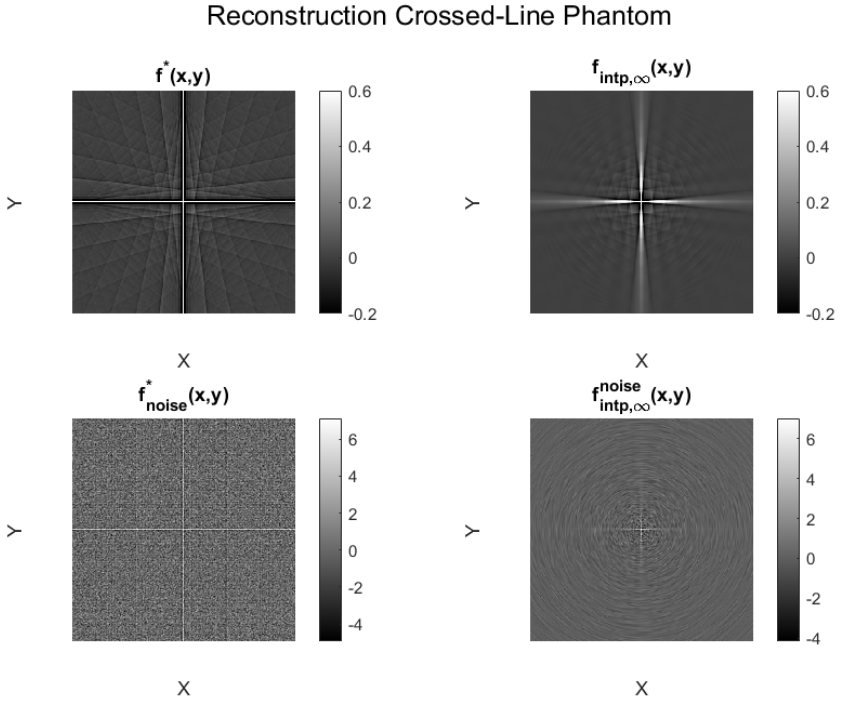


Figure 3.2: Top left: reconstruction after a sparse measurement. Top right, the reconstruction followed after filling the empty parts of the sinogram using linear interpolation. Similar for the bottom row, but with noise added to the measured data. Notice how rotational smearing reduces the noise and the artifacts, but also smears out the features of the object.

The methods of linear interpolation for data inpainting, was also applied on a more complex phantom than the crossed-line phantom, namely the Shepp-Logan phantom. The Shepp-Logan phantom is defined on a 256 by 256 grid shown in figure 3.3. Artificial sinogram data is generated by interpolating the pixel values on a 320 by 320 grid and applying the forward radon transform operator, from 0 to 180 degrees in steps of 0.01 degrees, to the phantom followed by adding 1% noise and interpolating back from 320 to 256 pixels. The interpolation steps to different grid resolutions are performed to prevent an inverse crime. An inverse crime is committed when the same model is used for both the forward and inverse operator

resulting in 'trivial inversion' [37]. The sparse data is generated by subsampling the fine resolution data for the angles from 0 to 180 degrees with steps of 2 degrees and a sinogram with steps of 6 degrees. The filtered back projected reconstructions, with a Ram-Lak filter, will be used as a reconstruction method. The reconstruction with these sparse measurements, and the measurements after inpainting using linear interpolation, are compared.

The generated sinogram data and corresponding reconstructions for sparse data are shown in figure 3.3. Notice how the the sparse sampling artifacts become more pronounced when increasing the sparsity. The results after filling the sinogram domain using linear interpolation is shown in figure 3.4. The sparse sampling artifacts are less pronounced after filling the sinogram using this artificial data. The amount of noise is also reduced. For the case with a 6 degree angular spacing, the rotational convolution as predicted by proposition 3.11, getting wider for larger radii, is clearly visible. Especially for the small circles at the bottom of the phantom, this results in decreased visibility of these fine features.

The spherical case of photoacoustic reconstruction is also considered. Time reversal is used as a reconstruction method, which is explained in section 2.1.3. An image of a blood vessel was used as a source, surrounded by a half-ring of 30 detectors. Data was generated using the popular wave propagation toolbox k-Wave [38]. The reconstruction using a sparse measurement with 30 detectors, a measurement with a full set of 90 detectors and a reconstruction after linear interpolation to increase the number of measurements artificially is shown in figure 3.5.

We have seen how linear interpolation in the measurement domain results in an angular convolution in the image domain for the planar CT geometry. Visual inspection tells us that the benefit of linear interpolation are suppression of the sparse view artifacts, and noise reduction by smearing out the noise. A downside is loss of details and resolution by also smearing out the fine details of the actual object. This gets even worse for larger radii. It is very clear that linear interpolation is a very 'stupid' way of filling the empty spaces of the measurement domain. None of the characteristics of the imaging operator are taken into account. Linear interpolation can be seen as a regularizer that favours solutions that are smooth in the angular direction, which is hard to justify for the usual objects of interest, like the internal organs of human patients. For the spherical geometry used for photoacoustic imaging and time reversal as a reconstruction method, there is a considerable improvement in the reconstruction quality. It is known that time reversal does not perform well for sparse detector geometries [39] and that continuous detector surfaces perform much better. Clearly, even just using linear interpolation to simulate measurements for extra detectors improves the reconstruction quality.

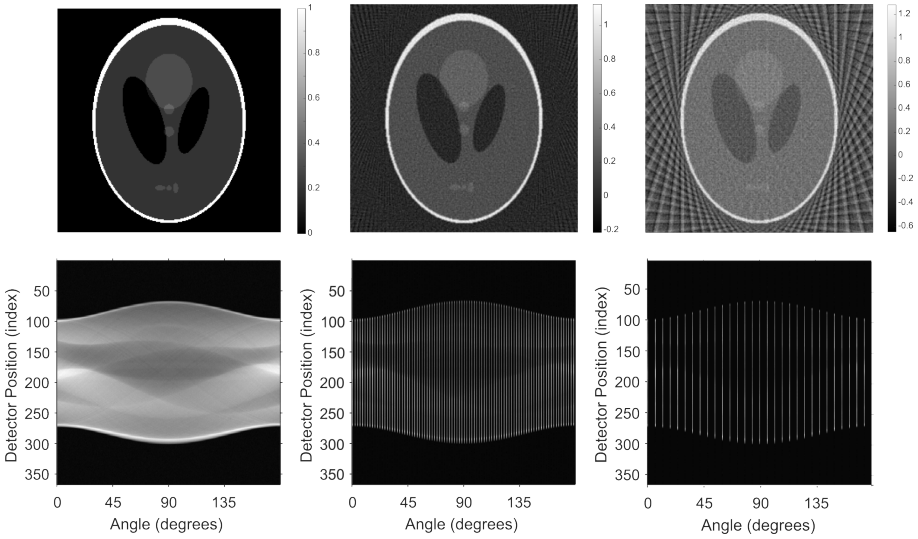


Figure 3.3: In the left column, the Shepp-Logan phantom is shown alongside its sinogram. The middle column displays a sinogram with an angular spacing of 2 degrees and the corresponding reconstruction obtained using filtered back projection. The right column presents a similar setup, but with an angular spacing of 6 degrees.

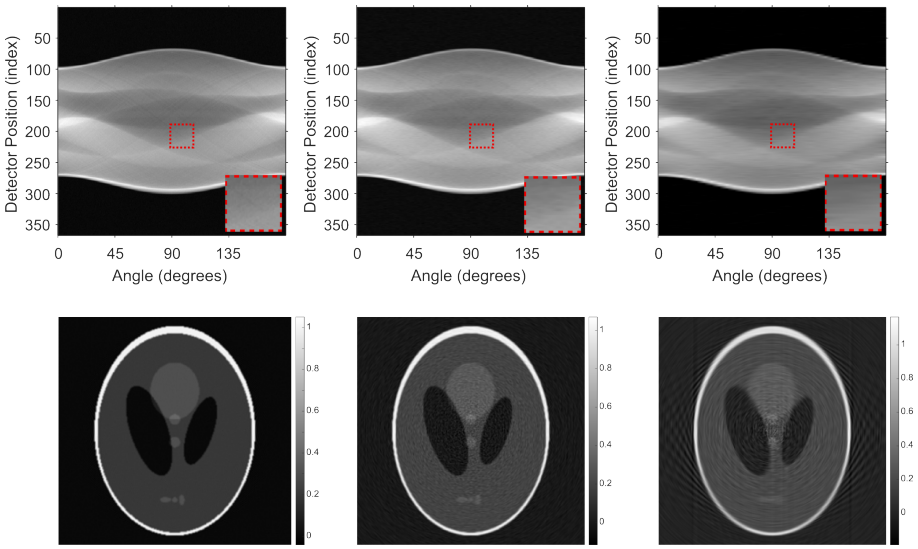


Figure 3.4: From left to right: the filled sinogram for all detector angles, for an angular step size of 2 degrees and an angular step size of 6 degrees, with in the bottom row the corresponding reconstructions obtained using filtered back projection. Notice the rotational smearing in the sparsely sampled reconstructions and the horizontal smearing in the sinograms (see insets).

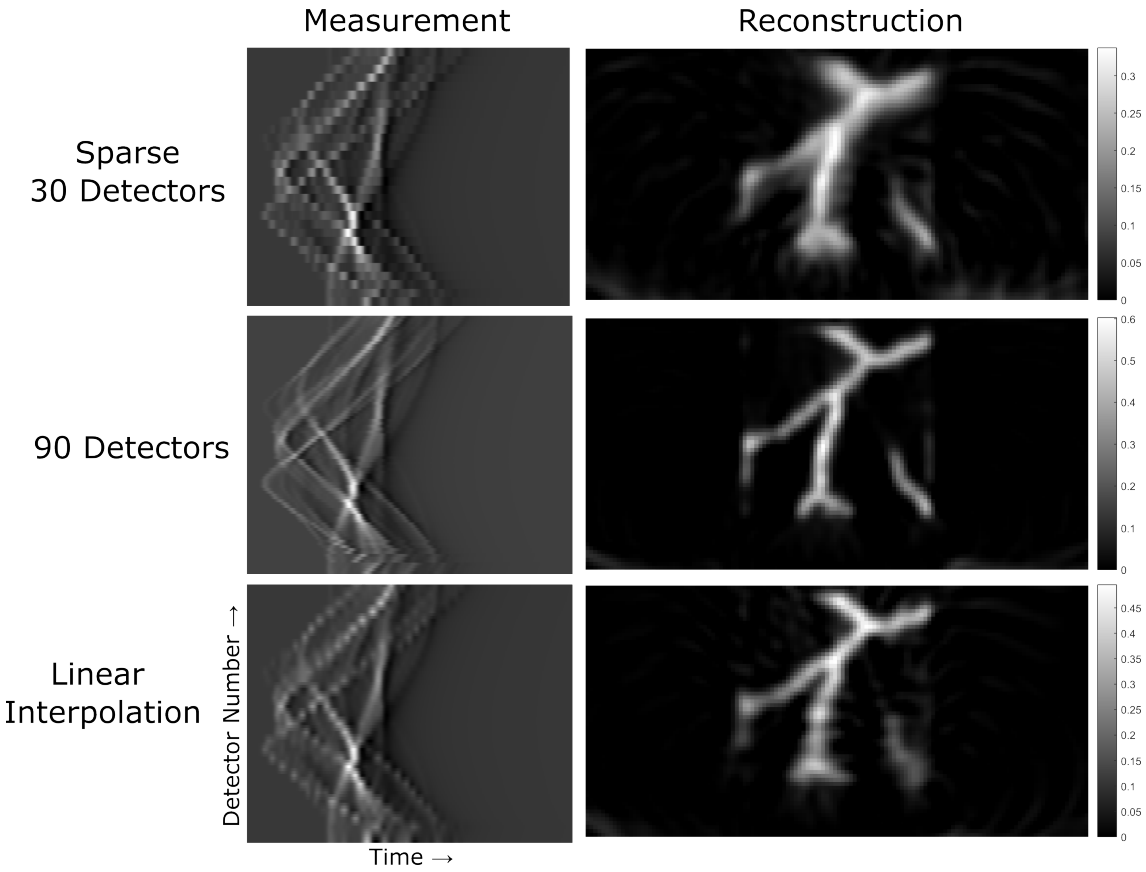


Figure 3.5: The photoacoustic reconstruction are shown for 30 and 90 detector geometries. Also shown, is the reconstruction for 30 detectors after inpainting in the measurement domain using linear interpolation.

3.2 Sinogram Inpainting Using Neural Functions

In the previous section, we saw the effect of a simple interpolation method in the sinogram domain on the resulting reconstructions. In this section, method used by Sun et al. [18] will be used. Sun et al. used neural functions to represent data. An advantages of these neural functions, as often stated in literature, is that they can be used to create images of arbitrary resolution [18]. In the article by Sun et al. [18] a multi-layer perceptron (MLP) with Fourier feature mapping is used. Instead of an MLP a Fourier feature mapping, the SIREN architecture introduced in section 2.2.1, will be used as a neural function because it has been shown that SIRENs perform better than Fourier features mappings for representing details in data [17]. Thanks to the implicit regularization properties of NFs, i.e. the inherent smoothness of NFs, a smooth continuous representation of data can be obtained while still preserving all the fine details.

The architecture for the MLP was taken from [17]. The input is a 2D coordinate that goes through 4 hidden layers of 256 nodes per layer and is transformed to a

single output for the network. The frequency parameter for the activation function of the SIREN, ω, is set to 30. The network is trained for 10000 epochs using an Adam optimizer with a step size of 0.0001 and a mean squared error loss. The result for the reconstructions using a SIREN network, is shown in figure 3.6. The case with a 6 degree angular detector spacing was also trained after reducing the frequency parameter, ω, to 20, which helps to increase the distance over which the output is smooth. Notice how the case with a 2 degree angular detector spacing is very similar to the results using linear interpolation in figure 3.4. For the 6 degree angular spacing, the reconstruction quality improved a lot after decreasing omega to 20. The result for a 6 degree spacing might look better than when using linear interpolation at first sight, since the sparse view artifacts are a lot less pronounced, but small features are also reduced in visibility (see the small circles at the bottom of the phantom. The NF is also again applied to the photoacoustic geometry. The result is shown in figure 3.7. Notices how the result improved compared to the sparse detector geometry in figure 3.5. The result is even better then for linear interpolation. Because the detector are very far apart, linear interpolation does not capture the curves in the sinogram well, something which the NF can do.

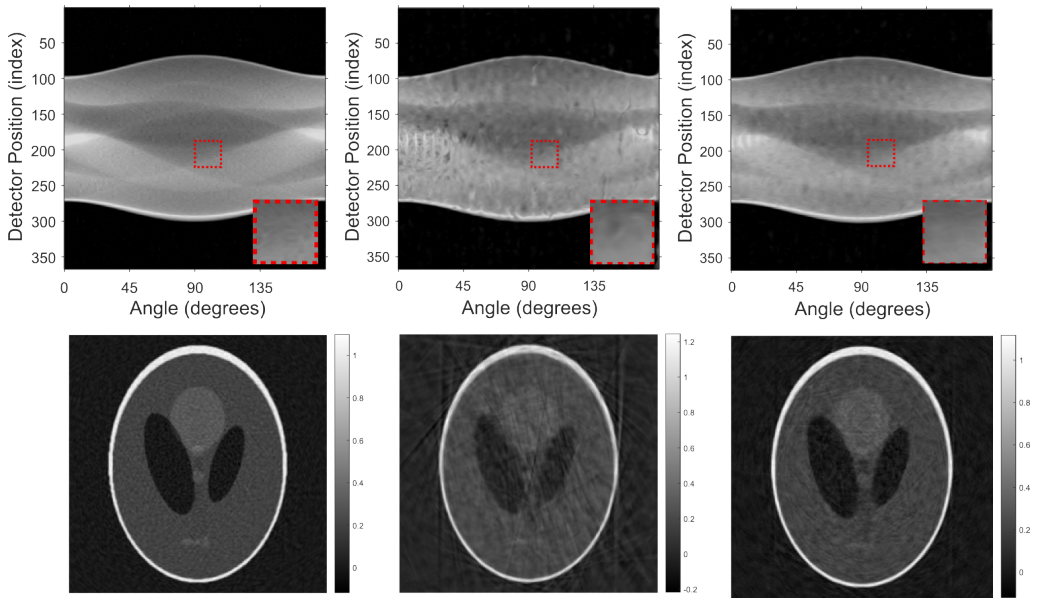


Figure 3.6: The left column shows the filled sinogram that was obtained with an angular spacing of 2 degrees and the corresponding reconstruction. The middle column is for the angular spacing of 6 degrees. The last column is also for an angular spacing of 6 degrees, but with the frequency parameter of the SIREN reduced to $\omega = 20$.

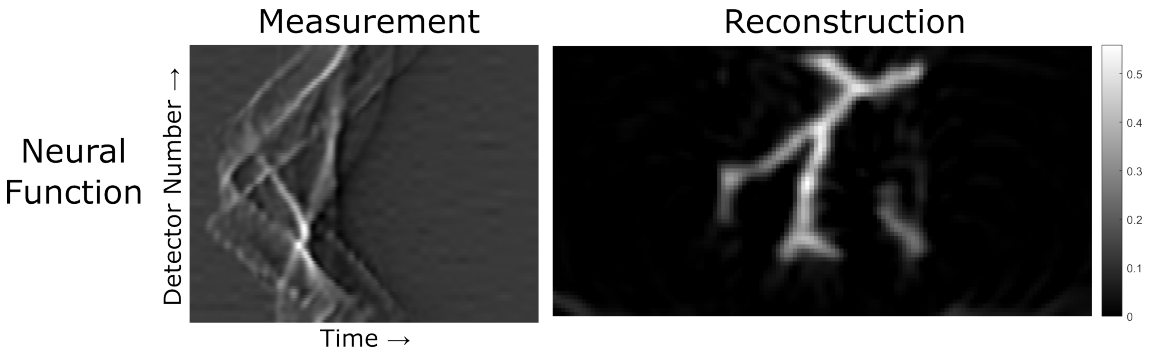


Figure 3.7: Enter Caption

3.3 Discussion and Conclusion

In this chapter, we have seen that linear interpolation in the measurement domain for the planar geometry used in CT imaging has the same effect as angular convolution. While this smooths out noise and artifacts, it also blurs actual features in the image, making it a poor choice for improving image quality in most cases, as was confirmed experimentally. Interestingly, for the spherical geometry used in photoacoustic imaging, there was some improvement in image quality. However, this is likely due to the specific reconstruction method used for PA imaging.

For interpolation using neural fields, similar outcomes were observed in the CT setting. We also saw that the frequency parameter has a significant impact on reconstruction quality. No hyperparameter tuning was performed in this chapter, so it is possible that better results could be achieved by optimizing the frequency parameter and the network architecture for specific applications. In the photoacoustic case, neural fields outperformed linear interpolation, which failed to capture the curved structures present in the sinogram. However, this does not necessarily mean that NFs are better than classical methods in general. Other approaches like spline interpolation may be more suitable for approximating curves.

Overall, the use of NFs for data inpainting relies on implicit regularization, meaning their outputs are inherently smooth. While NFs can produce outputs at arbitrarily high resolution, this refers to grid resolution, not the ability to recover finer details. As such, their usefulness for photoacoustic imaging appears to be quite limited.

Chapter 4

Neural Operators for Wave Simulations

Large scale photoacoustic image reconstruction is a computationally expensive task. As an example, photoacoustic image reconstruction for the PAM3 takes 3 hours and 20 minutes [3]. The bottleneck in the current iterative reconstruction scheme, are the forward and backward wave simulations, which take take 10 minutes per simulation on a powerful GPU [3]. The popular MATLAB toolbox k-Wave [38], a pseudo-spectral time domain solver, is the current standard for wave simulations for photoacoustic image reconstruction, but is a limiting factor for speed up of the reconstruction process.

Several attempts have been made to create classical wave solvers that are faster than k-Wave. See for example [40]–[42]. New to the field of solvers for PDEs are the so-called neural operators. These deep learning techniques can approximate maps between function spaces by learning from data [13]. Impressive speed ups, up to a factor of 700,000 increase in computational speed [13], have been reported in literature for simulating various scientific phenomena, like weather forecasts, with neural operators.

An initial attempt of using neural operators, specifically the FNOs introduced in section 2.2.2.2, for speeding up wave simulations was made by Guan et al. [43]. They have shown that FNOs can be used to simulate waves on a square domain with a line detector and a homogeneous medium. They hypothesize that FNOs can be used for the simulation of waves in media with inhomogeneous properties, but do not demonstrate this.

In this chapter, I will explore the use of FNOs as fast solvers of the wave equation and reconstruction methods for the PAM3 geometry. I will start by using the FNO as a forward wave solver in section 4.1 followed by using FNOs as backward wave solvers in section 4.3. I will go a step further than Guan et al. [43], by not only simulating waves in homogeneous media, but also in media with inhomogeneous speed of sound distributions. Next, I will show how these surrogates for forward and backward wave solvers can be used in an iterative reconstruction framework in section 4.4. Finally, I will study and comment on the claimed super-resolution capabilities of FNOs [23] in section 4.5.

4.1 Homogeneous Forward Wave Simulations

Simulating the propagation of waves can often easily be described in k-space, see for example the pseudo-spectral solver used by k-Wave [38] or the angular spectrum method [44]. Therefore, I will try to use FNOs as a fast surrogate for wave solvers, since FNOs are known to perform well for problems that have a simple representations in k-space [25]. In this first chapter, FNOs will be trained to work as a forward wave solver, so describing the acoustic forward problem in figure 2.1. As mentioned before, Guan et al. [43] have already made a first attempt at solving the wave equation using FNOs, and their article forms a basis for this chapter.

Before describing the used FNO architecture and the method used to train the FNO, it is important to mention the following. An FNO that describes the acoustic forward problem takes an initial pressure as its input, and should output the solution to the wave equation, which is a pressure as a function of space and time. The initial pressure, $p_0(x, y)$, $x, y \in [0, 1]$, will be defined on a 2D domain. The solution to the wave equation, $p(x, y, t)$, $x, y \in [0, 1], t \in [0, T_{final}]$, will then live on a 3D domain. FNOs have the limitation that the input and output domains have to be the same. There are two possible solutions to solve this problem [28]:

- **Recurrent 2D FNO**: An FNO can be applied in a recurrent way, where the input is $p(x, y, t)$, $x, y \in [0, 1]]$ and the output is $p(x, y, t + \Delta T)$. By using the output as an input, we can predict the pressure for all t, albeit with a step size of ΔT.

- **3D FNO**: The initial pressure can also be lifted to 3D space, and by applying the FNO to this new input, $p_0^{3D}(x, y, t) = p_0(x, y)$, we obtain the complete solution, $p(x, y, t)$, to the wave equation for all $t \in [0, T_{final}]$.

An advantage of the 2D FNO is that it can extrapolate beyond the final training time, and an advantage of the 3D FNO is that it gives the whole time series at once, without have a finite step size for the time. 3D FNOs are known to be more expressive and easier to train than their 2D counterparts [23] which is also why Guan et al. [43] use the 3D FNO.

The architecture that will be used for the FNO that acts as a forward wave solver is inspired by Guan et al. [43]. The used FNO contains 4 Fourier layers with 5 hidden channels for each layer. All frequency components in the spatial direction are kept in $\mathcal{R}$ (see equation 2.8) and 64 frequency components are kept along the time dimension. After the 4 Fourier layers, a small channel mixing MLP transforms the data to 128 channels before outputting a single channel. The network is trained with a batch size of 4, for 2000 epochs with an Adam optimizer with a learning rate of 0.001 and a cosine annealing scheduler that slowly brings the learning rate down to 0 after 2000 epochs. A Sobolev loss will be used, because it is better at capturing high frequency details than the standard L2 loss function [45]. The used 1st order Sobolev norm is defined as the square root of the sum of the squared L2

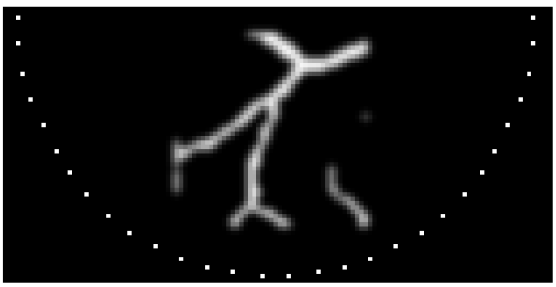


Figure 4.1: An initial pressure representing blood vessels is placed in the centre of a rectangular domain and is surrounded by a half-ring of perfect point-like ultrasound detectors.

norm of a function and its derivative, i.e. $\|f\| = \left(\|f\|_2^2 + \sum_{i=1}^{\#dim} \|\frac{\partial f}{\partial x_i}\|_2^2\right)^{1/2}$, where $\#dim$ is the dimensionality of the input coordinate of f. For the discrete inputs, the derivatives are calculated using a central differencing scheme along all the different dimensions separately.

The training data consists of 528 training pairs, consisting of the initial pressure and the solution to the wave equation. In a computational domain of 64 by 128 pixels, a small patch of blood vessels is placed in the centre of a ring of detectors, as shown in figure 4.1. This will be the initial condition. The solution to the wave equation is found using k-wave [38], as described in more detail in appendix C.

Training was performed on a 48GB NVIDIA A40 GPU and took 53 hours. The performance of the FNO is evaluated on an unseen test set of 120 images. The average relative mean squared error for the total test set is 0.0066. The result for one of the images in the test set is shown in figure 4.2.

The network has been trained and evaluated using images of blood vessels. It is essential for neural networks to generalize well for the output of the network to be trustworthy. As an example, for the PAM3, you do not want the network to remove signals coming from a circular tumour if it has only been trained on images of blood vessels. The capabilities of the network to generalize to unseen inputs will be tested by evaluating the network on the standard 'cameraman' test image [46]. This image is chosen, because unlike the vessels which consist of fine lines, the camera image consists of large high intensity areas with a stronger presence of low frequency components. The results are shown in figure 4.3. The relative mean squared error is 0.0087.

The main reason for using FNOs is the possible increase in simulation speed. The speed of the wave simulations using FNOs was evaluated determining the average time it takes to perform wave simulations for 50 initial pressures. This test was performed on a consumer laptop with a NVIDIA RTX 1000 Ada GPU and an intel core ultra 7 155h processor. The FNO and k-Wave wave simulations were both performed on the GPU. The average time it took for the FNO to perform the simulations was 0.162 seconds, and for k-wave it was 3.832 seconds, so there is a

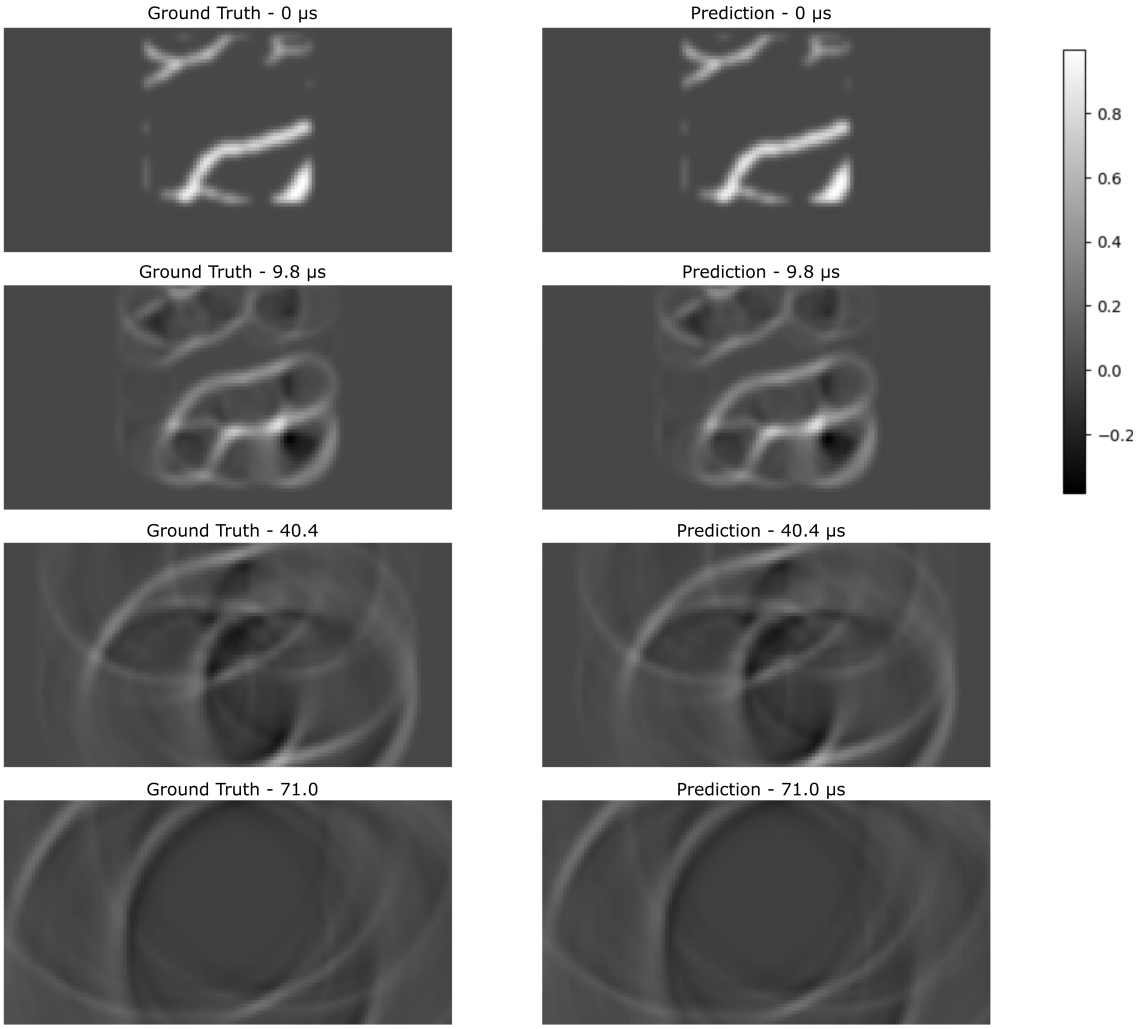


Figure 4.2: This figure shows a comparison of the output of the FNO and the ground truth provided by k-Wave for an unseen instance from the test set for the forward wave simulations. Visually, it is impossible to tell both results apart.

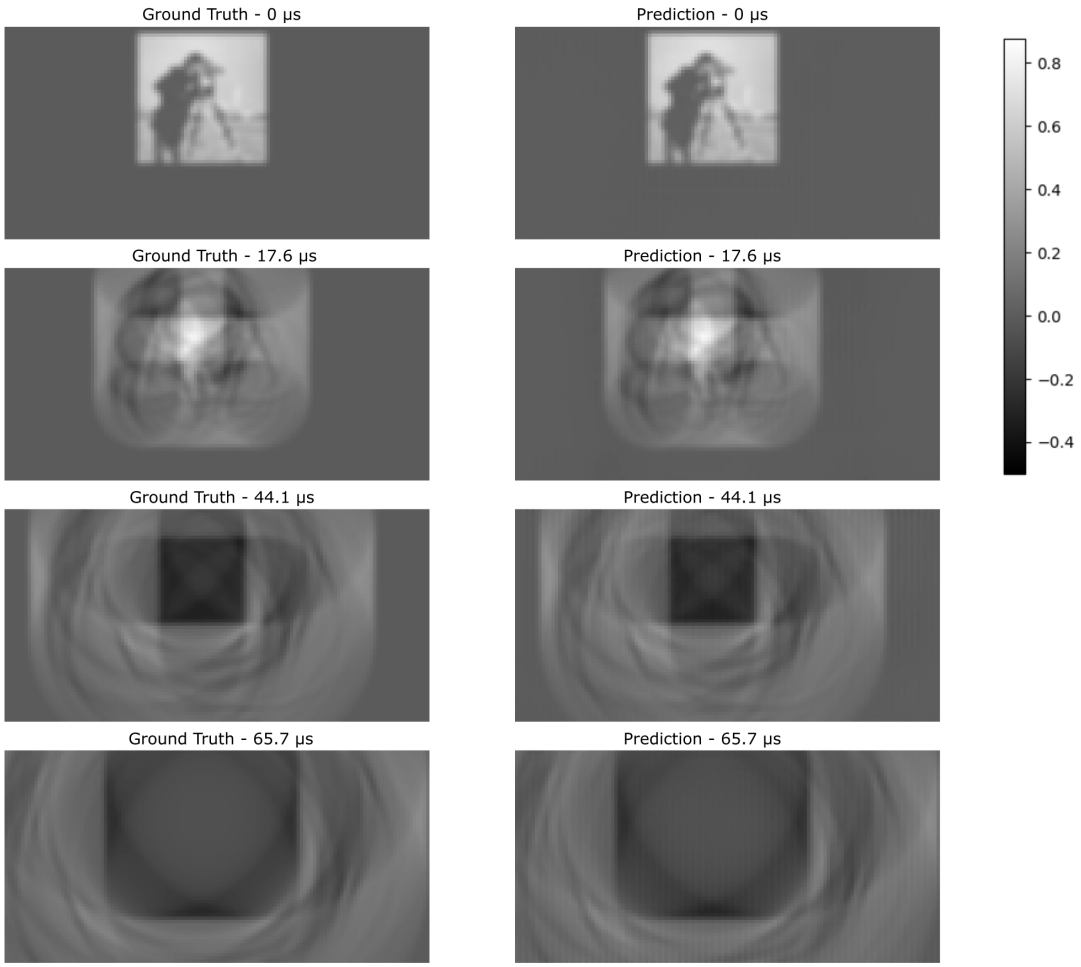


Figure 4.3: The forward FNO is tested on an out-of-distribution input. Some vertical stripes are apparent, especially for the later time points.

factor 24 increase in speed. For small domains, the the k-waves CPU code might actually be faster than the GPU code, so the CPU code was also tested, and after optimizing the number of threads used for optimal speed, it was possible to reduce the simulation time to an average of 2.080 seconds. The increase in speed is thus a factor of 13, similar to the results reported by Guan et al. [43].

4.2 Inhomogeneous Forward Wave Simulations

We have seen before in section 2.1.3 that breast tissue is not acoustically homogeneous. Artifacts are introduced in the wave simulations when, incorrectly, a constant speed of sound assumption is made, as was done in the previous section. With the PAM3, the speed of sound distributions can be determined, by using the transducers in emission mode for ultrasound tomography, and using a fast time of flight reconstruction method [47]. In this section, the inclusion of the SOS map as an input of FNOs will be studied to test the abilities of FNOs to correctly describe wave propagation in inhomogeneous media.

FNOs can accept multiple channels as their input. Up till now, only one channel has been used, containing the initial pressure distribution. A second channel can be added containing the SOS distribution. This second channel is constructed in a similarly to the first channel, where the SOS map is copied to generate a 3D input for the FNO, i.e. $c^{3D}(x, y, t) = c(x, y)$, where $c(\cdot)$ is the SOS map, and $c^{3D}(\cdot)$ contains stacked copies of $c(\cdot)$ to get the input and output dimensions of the FNO to match.

For the following experiments, the same initial pressure distributions, representing blood vessels, will be used as before. A second channel, representing an inhomogeneous SOS distribution, is created by first setting the background pixels to the value of 1470 m/s. Next, 6 elliptically shaped regions are selected and the speed of sounds of the first three is set to 1540 m/s and the other three is set to 1400 m/s, which is approximately the range of SOSs that can be found in breast tissue [6]. The ellipsoids have a random orientation, the semi-major axis has a length between 10 and 17 pixels, and the semi-minor axis has a minimum size of 10 pixels and are smoothened with a Blackman window. The ellipsoids are placed completely within the the rectangular area 15 pixels away from the boundary of the complete domain. See figure 4.4 for 2 examples of generated speed of sound distributions.

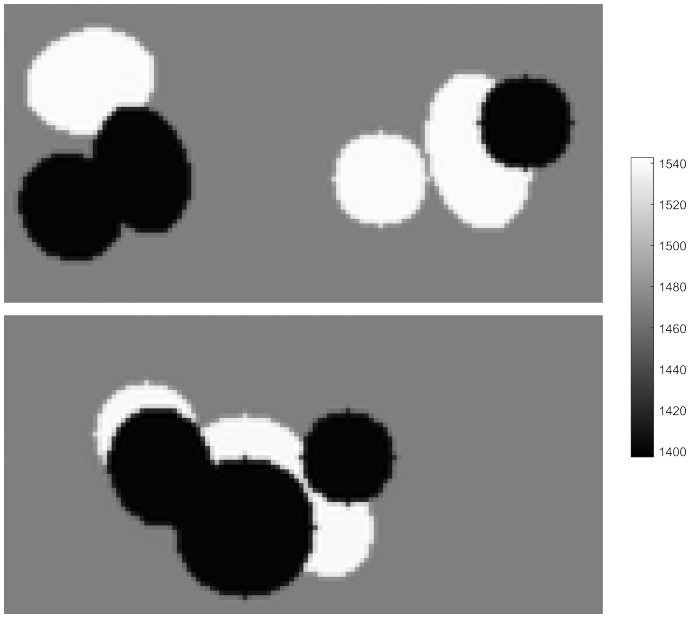


Figure 4.4: This figure shows two of the randomly generated speed of sound maps. The colorbar shows the value for the SOS in m/s.

The same training conditions and architectures were used as for the homogeneous case. This new FNO is tested, and the results of the wave simulations are again compared and shown in figure 4.5.

To see whether that there is actually a benefit of including the SOS map, the ground true wave simulation provided by k-Wave, simulated in a medium with an inhomogeneous SOS distribution, is also compared to the output of the network that assumes a homogeneous SOS. The results are shown in figure 4.6 and figure 4.7. The output of the FNO that has the speed of sound map as an input is much closer to the ground truth than the FNO which assumes a constant SOS. The relative MSEs are 0.1610 and 0.4905 respectively.

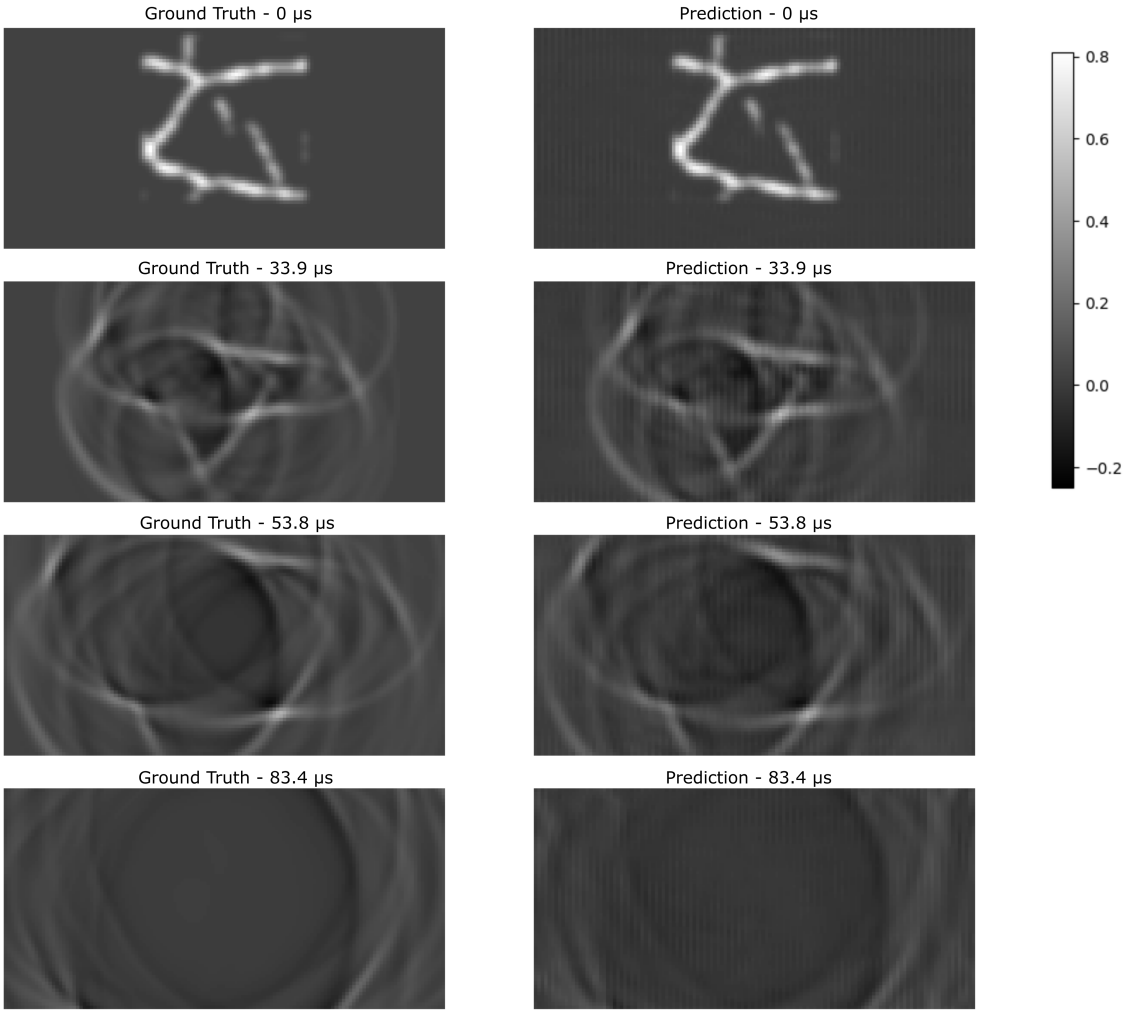


Figure 4.5: The ground truth, provided by k-Wave, is compared to the prediction of the FNO. Note the harmonic oscillations present in the output of the FNO

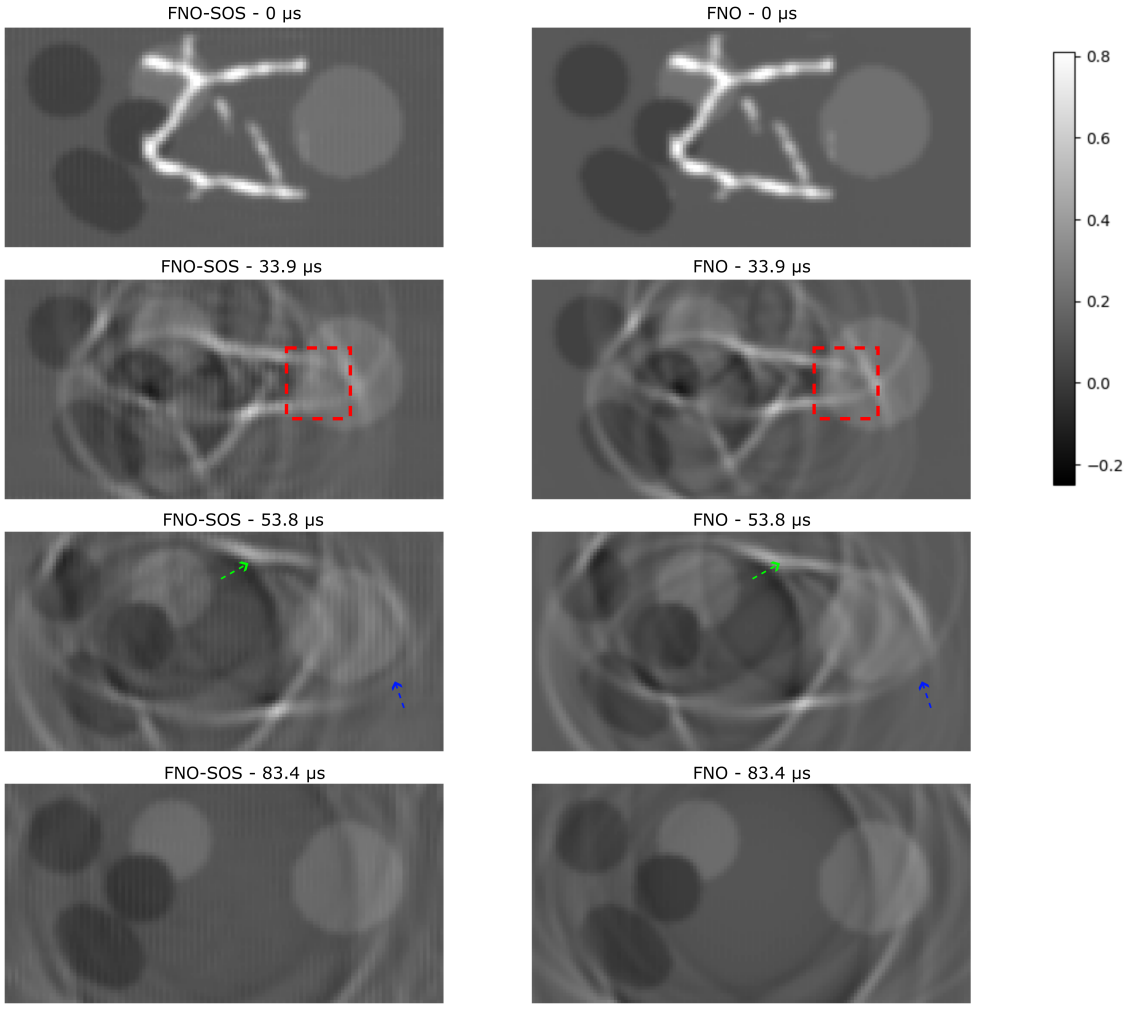


Figure 4.6: The solution to the wave equation, given an inhomogeneous SOS, predicted by the FNO which assumes a inhomogeneous SOS is compared to the FNO that takes the SOS as an input (FNO-SOS). The SOS distribution is shown in the background. The white circles correspond to locations with a speed of sound of 1540 m/s and the black circles to a SOS of 1400 m/s, for the rest of the medium, the SOS is 1470 m/s. Some obvious differences are marked by the red box, and the green and blue arrows. Red box: the two wave fronts in the high-SOS region are further apart in the FNO-SOS prediction than the FNO prediction. Green arrow: the wave front travelling upward, which passed through a high-SOS region, has travelled further in the prediction by the FNO-SOS than the FNO prediction. Blue arrow: The wavefront travelling downwards, just exciting the high-SOS region, has travelled further in the SOS-FNO prediction than the FNO prediction. The differences of the SOS-FNO are correctly predicted as can be seen in figure 4.7.

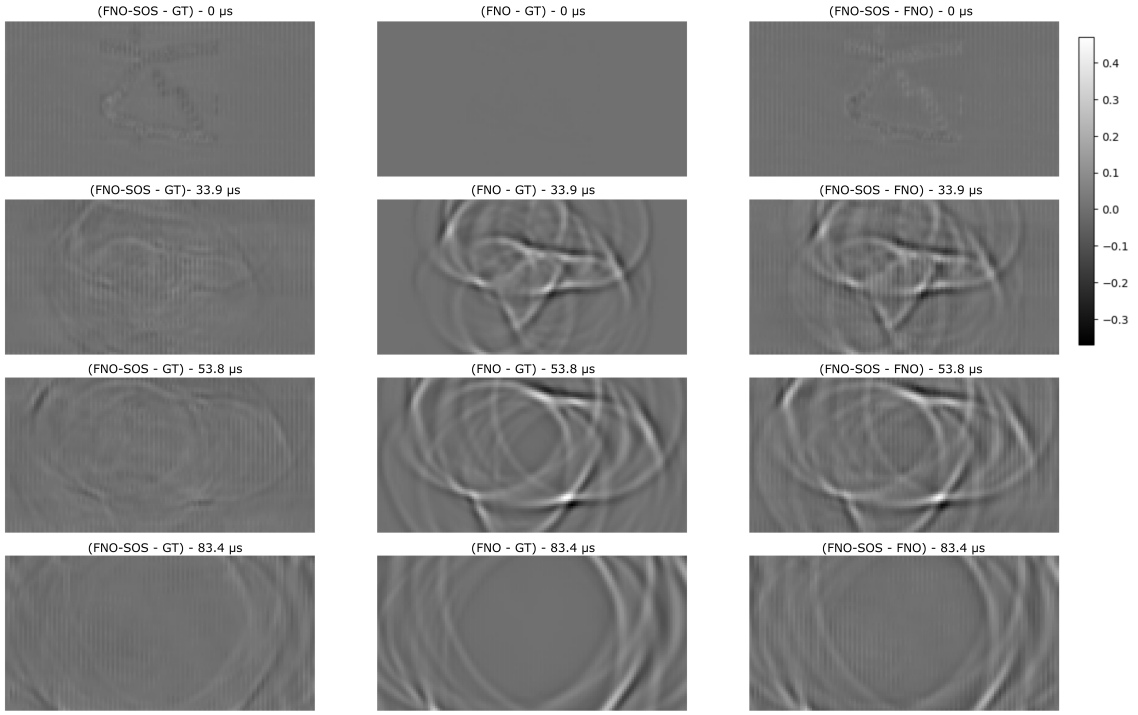


Figure 4.7: The difference between the ground truth and the FNO-SOS is shown in the left column. The error is relatively small compared to the middle column, showing the error between the FNO and the ground truth. Comparing the errors of the FNO and FNO-SOS, we find that much of the error in the FNO, arising from the incorrect assumption of a homogeneous SOS, is mitigated by incorporating the SOS as an input in the FNO-SOS.

4.3 Time Reversal Wave Simulations

In section 4.1, we have seen how FNOs can accurately describe the forward problem in figure 2.3. For image reconstruction, an operator is needed that transforms objects from the data domain back to the image domain. Off course, it would be optimal to have access to an inverse, but often this is not possible, for example in the case of sparse measurements. In those cases, alternatives like a pseudo-inverse [48] or iterative scheme in which the adjoint [8] and forward operator are used to go back and forth between the image and object domain can be used. Another option, which can be explained in a very intuitive way, is time reversal, which is the technique that will be studied in this section.

Time reversal takes the measurements at the sensors, reverses them in time, and plays them back, where the sensors act as the source of the ultrasound waves. After solving the backward wave equation, using these time dependent Dirichlet boundary conditions, the reconstruction, corresponding to the initial pressure, is given by the

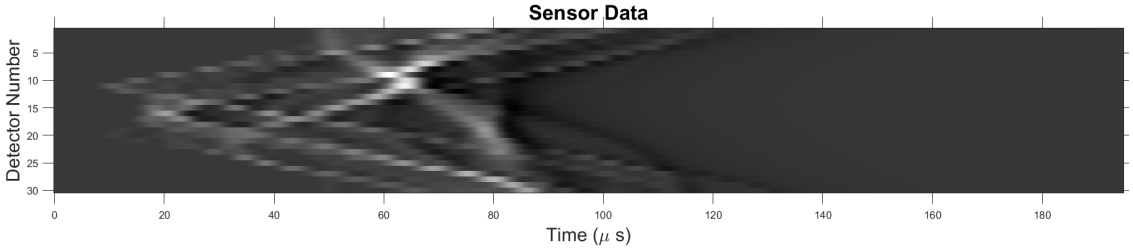


Figure 4.8: Sensor data can be represented as a function of detector number and time since the start of the measurement.

pressure at t=0 [8].

Training an FNO to act as a time reversal operator is similar to the method used for training an FNO as a forward operator. The main difference being the way the data is represented. Sensors in the photoacoustic imaging setup of figure 4.1 measure the pressure as a function of time. We can represent this measurement as shown in figure 4.8. Just like for the case of the forward wave simulations, we have the problem that the domain on which the sensor data is represented in figure 4.8 is different from the domain on which the solution to equation 2.4 lives. This problem can be solved by representing the sensor data in a different way. The output of the network is a pressure as a function of space (2D) and time (1D). We can represent the sensor data by setting all values in a 3D array to zero, except for the voxels at the sensor locations. These voxels are given the value of the pressure the sensor measured, but then in time reversed order. So, the input to the network, p^{TR}_{input}, representing the sensor measurement, is given by:

$$p^{TR}_{input}(x,y,t) = \begin{cases} 0, & \text{if } (x,y) \text{ is a sensor location (see figure 4.1),} \\ p(x,y,T-t), & \text{else.} \end{cases} \tag{4.1}$$

where $p(x,y,t)$ is the solution to the forward problem, and $t \in [0,T]$ is defined like before. This input is graphically depicted in figure 4.9.

The output part of the training data, which is the time reversal simulation, was simulated using k-Wave. The same training parameters were used as in section 4.1. The average relative mean squared error for the images in the test set was 0.0063. The result for one of the instances in the test set is shown in figure 4.10. Note how there is a large difference in the frequency content between the input and output array, but there this does not seem to cause any problem, and just like for the forward simulations, it is practically impossible to tell the ground truth and the network prediction apart.

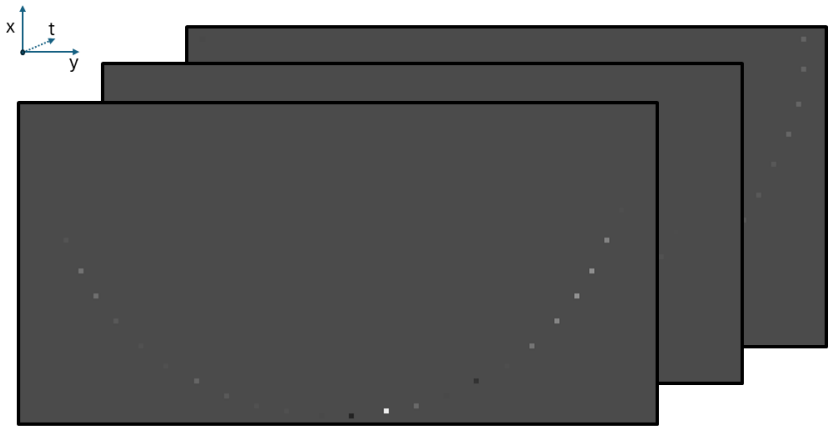


Figure 4.9: The pressures measured at the detector locations, are reversed in time, and placed in the 3D grid at the locations of those detectors to satisfy the need for the input and output of the domains of the FNO to be equal.

4.4 Iterative Reconstruction

In the one-step reconstruction of the photoacoustic image in figure 4.10, clear artifacts are visible, caused by sparse distribution of detectors. It is known that time reversal will give artifacts in non-ideal imaging circumstances like sparse-view and limited-view imaging, but this can be solved using an iterative approach [39]. The basis for many iterative photoacoustic reconstruction algorithms is the same, and the idea is the following:

1. Start with an initial guess for p_0.
2. Use the forward model to generate the sensor data p_0 would produce (forward wave simulation).
3. Compare the sensor data to the actual measured sensor data.
4. Update p_0 based on the difference between the actual sensor data and the sensor data p_0 would have generated using a backward wave simulation.

The trained FNOs from section 4.1 and section 4.3 are very flexible and can be used as surrogates for all wave simulations in any reconstruction method, as long as the size of the domain is the same as the size of the training domain. These FNOs will now be tested for iterative time reversal reconstruction. Let $K\{\cdot\}$ denote the forward FNO, and $K^{TR}\{\cdot\}$ the time reversal FNO, then the iterative time reversal algorithm is given by:

Algorithm 4.1 (Iterative time reversal with a non negativity constraint)**.** Iterative time reversal is a reconstruction method used for photoacoustic image reconstruction

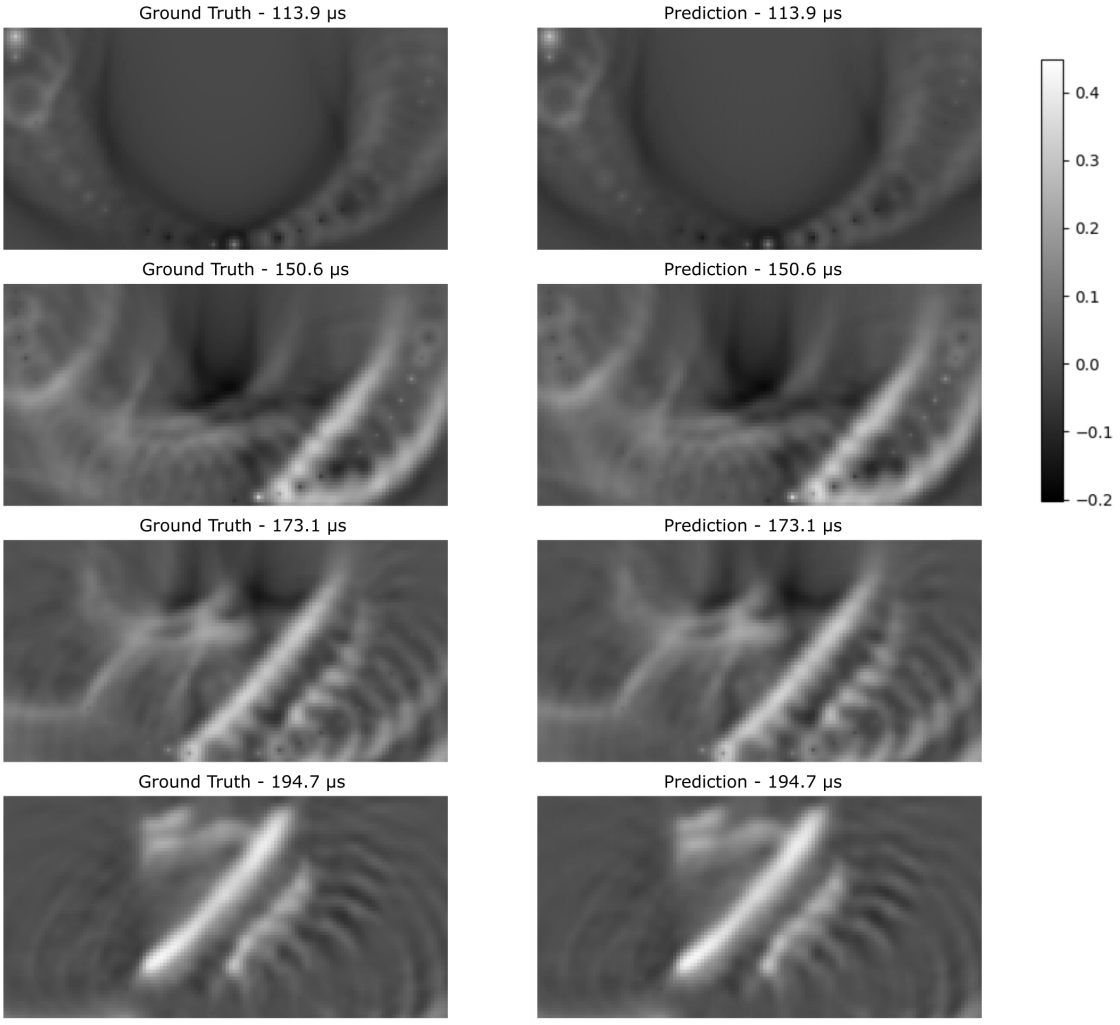


Figure 4.10: The time reversal solutions provided by the FNO is compared by the ground truth solution provided by k-wave. Visually, it is impossible to tell the difference between both results. Note the clear circular artifacts caused by sparse sampling.

and consists of the following steps:
The initial guess for p_0, p_0^1, based on the measured data, $\mathcal{S}$, is given by:

$$\hat{p}_0^1 = K^{TR}\{\mathcal{S}\}$$
$$p_0^1 = \mathcal{P}\big|_{\mathbb{R}^+}\left\{\hat{p}_0^{11}\right\}$$

Each next iteration is then given by:

$$\hat{p}_0^{n+1} = p_0^n + K^{TR}\{\mathcal{S} - K\{p_0^n\}\}$$
$$p_0^{n+1} = \mathcal{P}\big|_{\mathbb{R}^+}\left\{\hat{p}_0^{n+1}\right\},$$

where $\mathcal{P}\big|_{\mathbb{R}^+}$ is the projection operator that projects the input onto the positive reals.

The previously trained surrogates for the forward and time reversal wave simulations were implemented in the iterative time reversal framework and tested on one of the images of blood vessels in the test set. The result of iterative reconstruction is shown in figure 4.11. Note how the error is getting smaller for each iteration and how the absolute amplitude is getting closer to the true amplitude. After a few iterations, sinusoidal patterns start to appear for the FNO reconstruction. This gets even worse if the algorithm is run for more iterations (see appendix D).

4.5 Zero-Shot Super Resolution?

In the previous sections we have seen that FNOs can be used as a fast alternative for the classical wave solver k-Wave. The results are very close to the true solutions, both for the forward and backward wave simulations, and even in an iterative framework. But why did we use neural operator, specifically FNOs, and not any other deep learning technique, like convolutional neural networks? The reason is that an FNO is a neural operator, which differentiates itself from standard networks like multi-layer perceptions in the sense that they can map between function spaces [13], so describing maps between functions defined on continuous domains and have infinite-dimensional inputs [25]. Neural operators are also said to be very different from standard networks like convolution neural networks because they are not limited to learning and making predictions at higher resolutions than of the training data [13] and they are discretization invariant [29].

In this section, these claimed benefits will be tested. To do this, one image of the test set is taken and transformed to different grid resolutions using Fourier interpolation. The Fourier interpolated image contains exactly the same frequency content as the original image, as long as we are in the representation equivalent zone. Next, the FNO is applied, and the output is transformed back to the original grid. If the FNO is truly discretization invariant, the result should not change, as long as we are in the representation equivalent regime. The error as a function

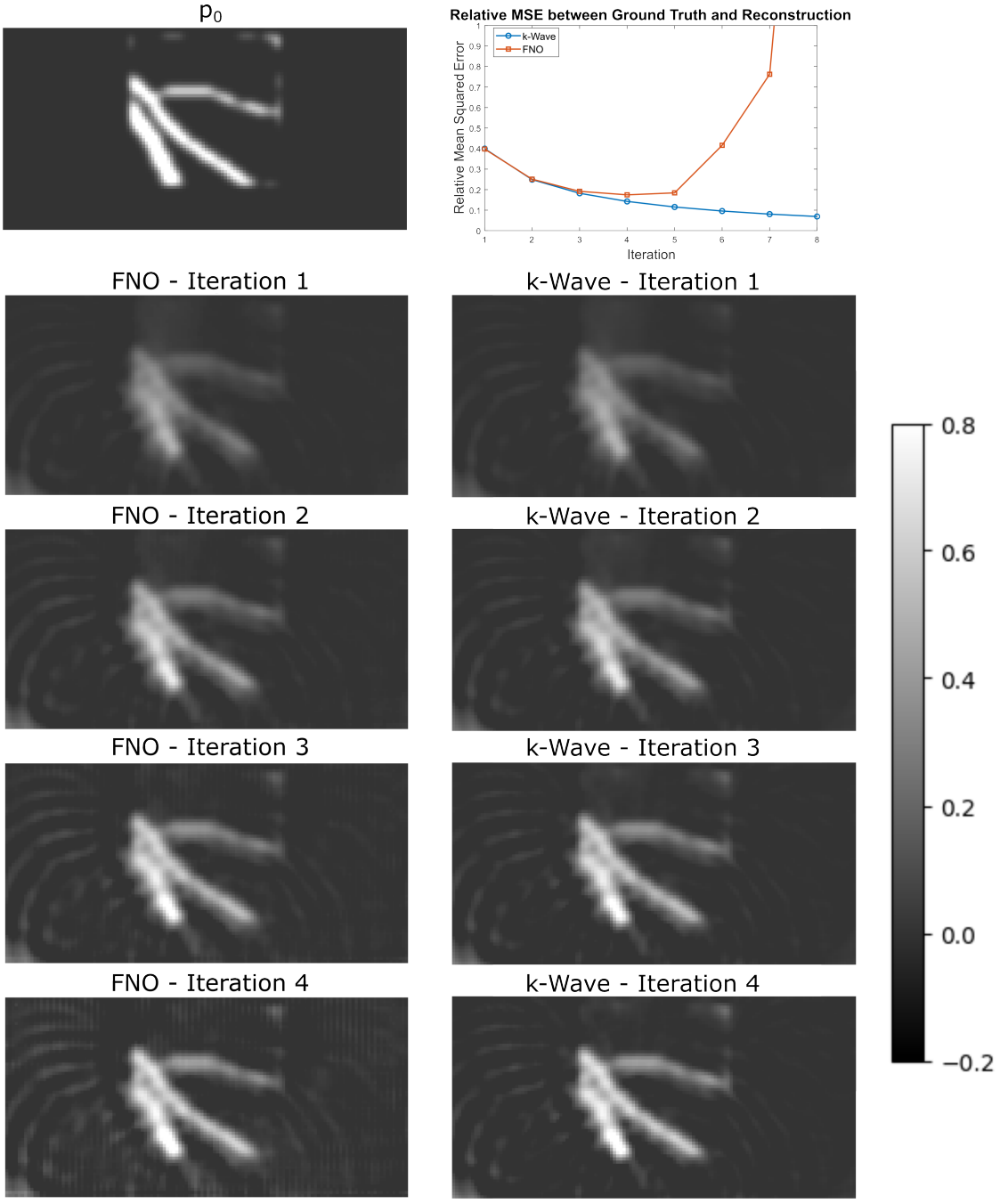


Figure 4.11: The previously trained FNOs are used for photoacoustic image reconstruction using the iterative time reversal algorithm. The results for each iteration is shown and compared to the result provided by k-Wave. After the 4th iteration the vertical stripes, which we have seen before, become very pronounced making the results unusable (see appendix D for iteration 5-8).

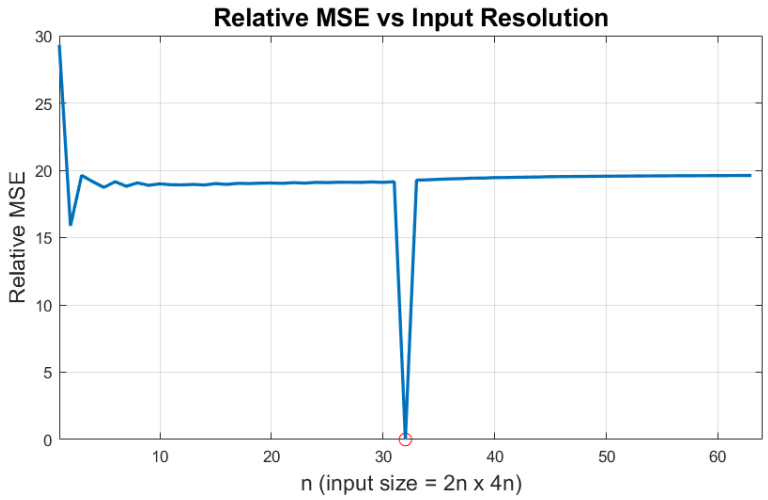


Figure 4.12: A single input image is transformed to different grid resolutions using Fourier interpolation. After applying the FNO, trained on images of size 64 by 128, the output is transformed back to a resolution of 64 by 128 pixels and compared to the ground truth. We can see that the FNO performs very well at the training resolution, but a huge decrease in performance is seen at different resolutions.

of grid resolution is shown in figure 4.12 and for several resolutions, the output of the network is shown in figure 4.13. We can see that the FNO performs very well at the training resolution, but a lot worse for different resolutions. For resolutions below the training resolution, the sinc oscillations corresponding to filtering out the higher frequency components with a rectangular window, become visible. for the higher grid resolution, high frequency artifacts are introduced, possibly because high frequencies can be created in the skip connection, which is not frequency filtered, after applying the nonlinear activation function. See also [49] in which similar results are observed.

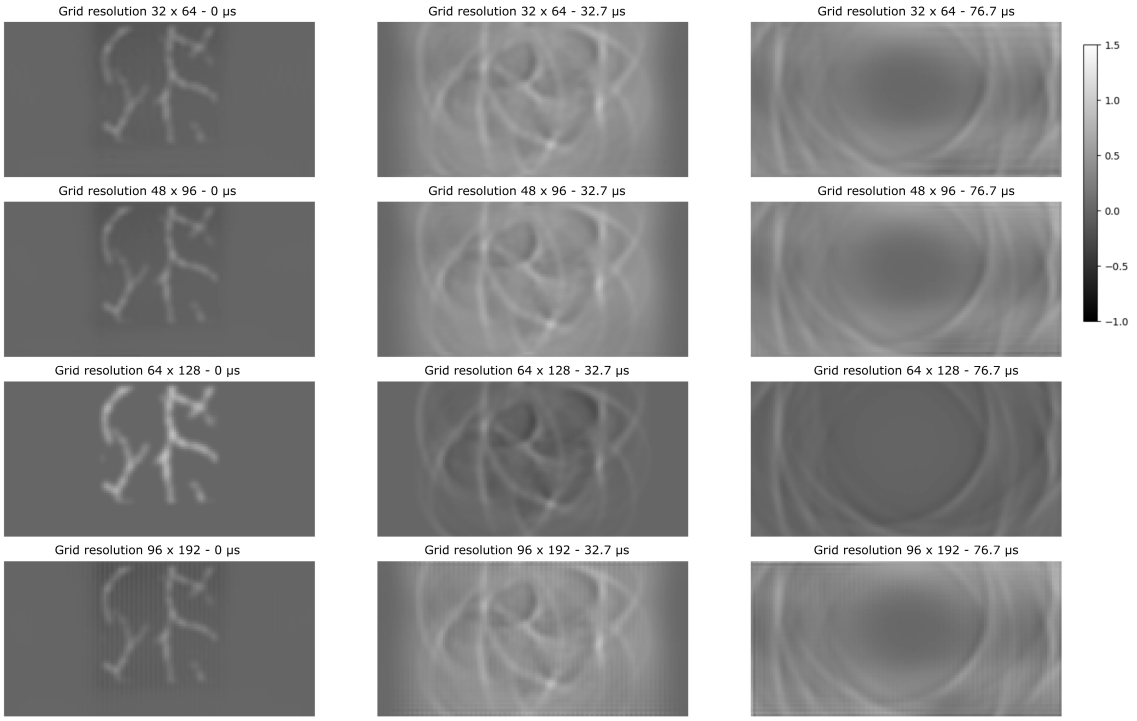


Figure 4.13: The FNO is applied on different discretizations of an image in the test set. For grid resolutions different from the training resolution, the performance is very poor. Note the sinc oscillations for lower grid resolutions and high frequency errors for high grid resolutions.

In a second experiment, an image that actually contains higher frequency components than the training set, is used as an input. The image has a size of 128 by 256 pixels, so twice the resolution, and the resulting k-wave simulation contains 955 time steps, resulting in the same T_{final} as used for the training set. The ground truth and FNO prediction are shown in figure 4.14. The outputs have been downsampled again to 64 by 128 pixels. The full output is of even lower quality (see appendix E). Note the sinc oscillations that appear even more clearly than before. Also, note the smoothing and disappearing of high frequency content.

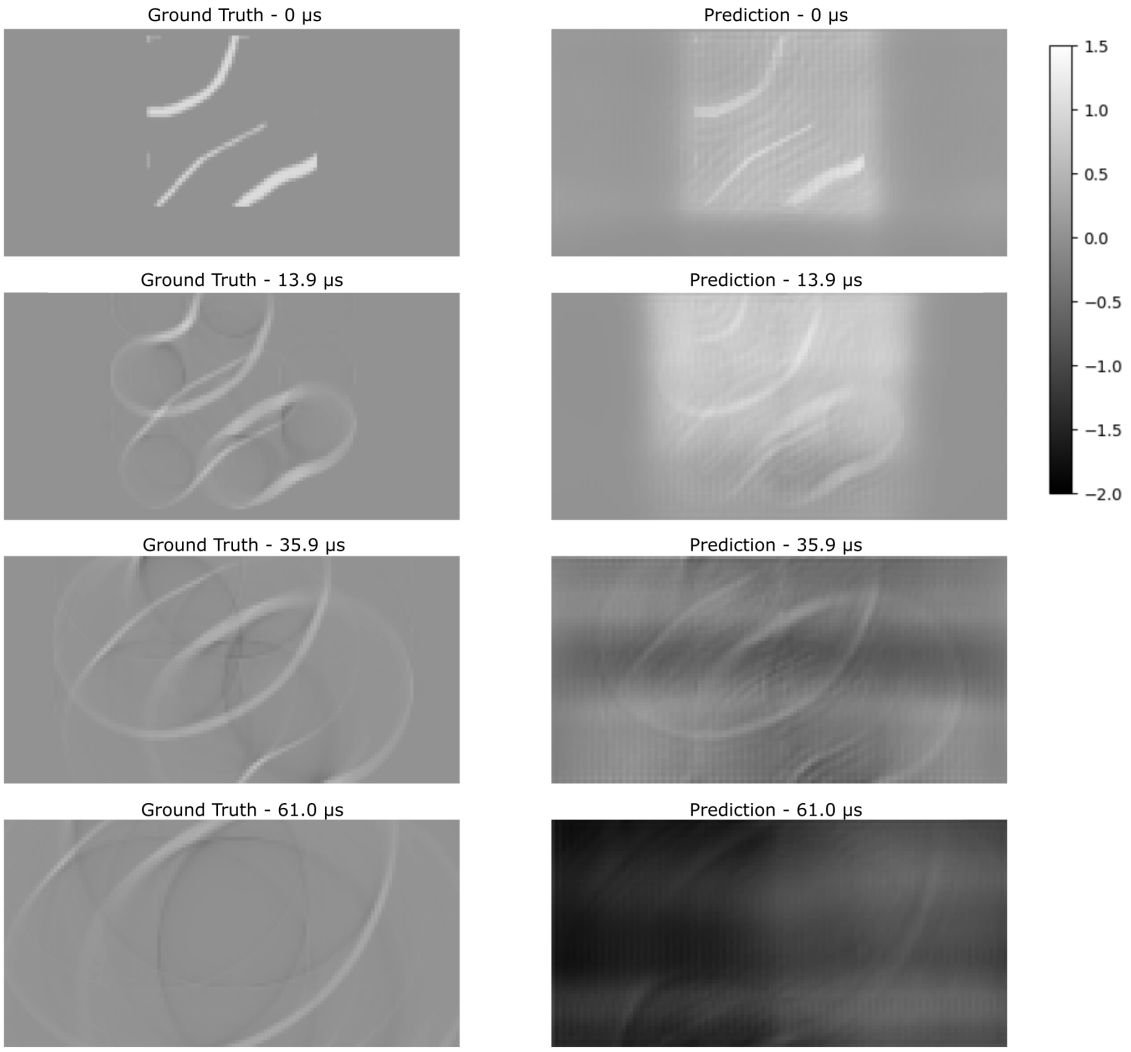


Figure 4.14: An image of size 128 by 256, containing higher frequency components than seen during training, is used as an input of the FNO, and the resulting prediction is compared to the ground truth. Note the sinc oscillations due to the rectangular k-space filtering applied by the FNO, and the removal of high frequency content.

4.6 Discussion and Conclusion

In this chapter, we have seen that FNOs can be used as accurate and fast surrogates of k-Wave as forward and time reversal wave solvers, and that they can generalize to out of distribution inputs. An increase in speed of about a factor of 13 was obtained. We must realize though, that k-Wave, even though being very popular and the standard for PA image reconstruction, is not necessarily the fastest classical method. This does not downgrade the result, since we have shown that FNOs can simplify calculations, by learning the distribution of images that can be encountered, instead of using a method that should work for all input, even though most of them are just random noise, and this could in theory reduce the computational complexity of any algorithm.

For the first time, it has been shown that FNOs can also accurately describe wave propagation in media with an inhomogeneous SOS. The quality of the results is lower than for homogeneous media, and oscillations appear, which is characteristic of FNOs, since they work in k-space and errors in k-space translate to oscillations in the spatio-temporal domain. The same network architecture was used for the inhomogeneous case as for the homogeneous case, which is not optimal considering the increase in complexity of the task. Increasing the network's complexity, will likely improve the results and reduce the artifacts, but further hyperparameter optimization is necessary.

The learned forward and time reversal operators have been implemented in an iterative time reversal framework. This approach is very general, and any other algorithm can be taken and have the wave propagations replaced with FNOs without retraining, provided the domain remains unchanged. We have seen that the results of iterative reconstruction are good for the first few iterations, but then degrade due to a build up of errors. The current approach is suboptimal, and methods like algorithm unrolling [50] can improve on these results, by preventing the error build up.

All experiments were performed on relatively small domains of just 64 by 128 pixels. Due to the requirement for FNOs to have the input and output domains to be the same, 2D images had to be represented as 3D arrays. This data-inefficient representation resulted in the need of 48GB GPUs for training the network. This is a big problem regarding scalability to the PAM3 data, which describes large 3D volumes that must be represented as 4D arrays. This leads to the same conclusion reached by Guan et al." [43] that this is far from current technological possibilities. An alternative approach to the one tried in this thesis, like a multi-grid approach [51], could solve this problem. It is difficult to say whether there is still an increase in computational speed using this method, warranting further investigation.

It is disappointing to see that the FNO is not capable of making predictions for different resolutions, as is claimed in [13]. We have observed that not only changing the grid resolution while keeping the frequency content the same, but also inputting smaller features corresponding to higher frequency components, results in very poor

performance. This is to be expected when considering the architecture of the FNO shown in figure 2.6. The FNO only learns a limited number of frequency components, and all frequencies beyond the maximum frequency present in the training data are set to zero, which explains the observed sinc oscillations. It is therefore also not accurate to claim that FNOs accept infinite-dimensional continuous objects as input, as suggested in [25]. Furthermore, examining the architecture in figure 2.6 and recognizing that multiplications in the Fourier domain correspond to convolutions in the spatio-temporal domain, we can see that the FNO architecture is similar to a CNN with skip connections. The important difference is that when an input is scaled to a higher grid resolution, the corresponding convolution kernel is scaled in the same way. This scaling is not done in standard CNNs, but a modified version called the convolutional neural operator (CNO) has been developed [49], which scales the convolution kernels appropriately. In the CNO, inputs with a different grid resolution than the one used during training are simply resampled to the training resolution using Fourier interpolation, and then later interpolated back to the desired resolution. This approach is very similar to what the FNO does.

The term neural operator for FNOs, along with all its claims about being an operator that maps between continuous function spaces instead of finite dimensional vector spaces dimensional spaces [25] is misleading. FNOs are not very different from standard CNNs, unlike what some people claim [13]. Using terms as representation equivalent neural operators [52] is probably more useful in classifying different types of neural networks, which relates a networks actions on the discrete function to the action on the underlying continuous representation.

Even if FNOs are not true operators between infinite-dimensional function spaces, this does not necessarily pose a problem for image reconstruction in the PAM3 system. Due to the limited detectable frequency of the transducers, the Nyquist sampling theorem ensures that a continuous function can be accurately represented with a finite number of coefficients. As a result, increasing the grid resolution beyond a certain point is unnecessary, since all information about the system is already captured within a finite set of values. However, accelerating the reconstruction process remains desirable, so a multi-grid approach using FNOs, or even standard CNNs or CNOs, could potentially address this challenge in the future.

Chapter 5

Neural Operators for Direct Inversion

Photoacoustic image reconstruction using iterative approaches is known to give better results than one-step approaches like time reversal in sparse- and limited-view imaging geometries [39], like for the PAM3 [3]. A downside of the iterative approaches is the high computational cost involved for the many wave simulation, as was discussed in chapter 4.

An alternative to learning parts of an iterative framework, and applying the learned operators multiple times, is using a fully learned approach where the inverse operator is learned from data. This approach has some disadvantages, like losing the explainability that we have in the iterative framework from chapter 4, since the operator is doing multiple things at the same time, i.e. performing inverse wave simulations, denoising, regularising etc. However, an advantage is having a fast one-step reconstruction method. This idea of learning the reconstruction operator from data is not new, see for example [53] for a more elaborate discussion on fully learned reconstruction operators. However, the use of neural operators for learning the photoacoustic reconstruction operator is new and will be investigated in this chapter.

There are several candidate neural operator architectures that can be used for learning the inversion operator. The operator architecture that will be studied, one of the most popular neural operators besides FNOs, is the deep operator network (DeepOnet). DeepOnets have been shown to be very successful in approximating operators all kinds of operators [28], but have never been used in the context of photoacoustic imaging.

This chapter consists of two sections. In the first section, a first attempt is made to perform photoacoustic inversion with DeepOnets. Hyperparameter optimization is performed for several DeepOnet architectures to find an optimal network that can be used for one-step reconstruction. The resulting network is tested on unseen data, the reconstruction speed will be determined and the super resolution capabilities of this so-called neural operator are tested in section 5.1. In the section 5.2, a connection will be made between DeepOnets and singular value decomposition (SVD), a connection also observed by others [54], which offers some new insights

and can help to understand what optimal reconstruction quality can be expected, helping to explain some of the results in section 5.1. Finally, the use of DeepOnets for direct inversion and what the results mean for the future of photoacoustic image reconstruction, will be discussed in section 5.3.

5.1 DeepOnets for Fully Learned Photoacoustic Inversion

The use of neural operators as photoacoustic reconstruction operators is new, so it is not clear what networks are the best choice. In chapter 4 we have already seen how one type of neural operators, the FNO, gives good results as a wave propagator, so one might question whether it will also perform well as a reconstruction operator. Besides the limitations of the FNO regarding inefficient data representation, there are some findings that suggest that FNOs, which describe problems in k-space, are not optimal as photoacoustic inverse operators, as is explained in appendix F. Therefore, FNOs will not be further investigated.

Another popular neural operator architecture is the DeepOnet. Compared to the FNO, DeepOnets do not have the disadvantage that they require the input and output domain to be the same. DeepOnets also work in the spatiotemporal domain instead of in k-space, and the results in appendix F show that this is beneficial. DeepOnets have the disadvantage that they only accept a fixed input size for the representation of the sensor data, but that is not problem in this case, since the sensor locations for the PAM3 are fixed[8]. DeepOnets also do not have the FNO's requirement that data is represented on a regular grid, but can be defined on an coordinate system. Another important difference with the FNO, is that the output of the DeepOnet can be queried at an infinite (grid) resolution, like for the neural functions in chapter 2, without the need for increasing the input size.

This chapter aims to find a suitable DeepONet architecture for rapidly solving the inverse photoacoustic problem and to evaluate the quality of the resulting reconstructions. To keep the architecture search manageable, it will be limited to the following four architectures:

	Branch	Trunk
Network 1	FCN	FCN
Network 2	FCN	SIREN
Network 3	CNN	FCN
Network 4	CNN	SIREN

The FCN-FCN and CNN-FCN were chosen because good results were obtained

[8]The whole bowl containing the sensors can rotate off course, but for a standard imaging protocol, measurements are performed for a fixed set of locations (see again section 2.1.2 for more details.)

with these network by others for solving PDEs [28], [55]. The decision to include architectures with SIRENs as trunk nets was made, because we know from section 2.2.1 that great results have been obtained with data representation using SIRENs thanks to their implicit regularization properties, and the fact that DeepOnets could be considered conditioned neural functions (see section 2.2.2.1) and might benefit as well from this. Only one other article was found that uses SIRENs for one of the networks in the DeepOnet architecture. In that article [56], DeepOnets are used for representing 3D shapes.

For the four different trunk-branch combinations, hyperparameter optimization was performed. Inspiration for approximately what network parameters should be considered came from [28]. The ranges of hyperparameters that were considered for each of the network combinations are given below:

1. **FCN-FCN:**For both FCNs, between 3 and 8 fully connected are used. All layers, except for the final layer, have the same number of nodes. The number of nodes for both the final layer, and the other layers, is taken from the set $\{32, 64, 128, 256, 512, 1024\}$. Note that the architectures for both FCNs can be different. ReLU activation functions are used.

2. **FCN-SIREN:** For the FCN and SIREN, the same architectures are used as in the FCN-FCN case. The frequency parameter of the SIREN, ω, is taken from the set $\{20, 40, 70, 100, 200\}$.

3. **CNN-FCN:** The FCN has between 3 and 5 layers and has 256, 512 or 1024 nodes for the last layer and a fixed number of 64, 128 or 256 nodes for the rest of the layers. The CNN has between 3 and 5 layers, a fixed kernel size of 3 or 5 for all layers and a stride of 2. The number of channels for the first layers is either 8 or 16, and double for each next layer, up to a maximum of 256 channels. ReLU activation functions are used after each convolutional layer. The output of the CNN is flattened, a small FCN is applied of either 2 or 3 layers. The last layer has 256, 512 or 1024 nodes, and the other layers have 64, 128 or 256 nodes.

4. **CNN-SIREN:** The same architectures were used as for the combination CNN-FCN, and the same frequency parameters, ω, were used as for the combination FCN-SIREN.

The number of unique networks that can be generated within the given parameter constraints is too large to feasibly train them all. For instance, almost 30,000 distinct CNN-SIREN network configurations are possible based on the defined parameter ranges. Therefore, for each of the 4 trunk-branch combinations, only 40 unique random models will be generated. The exact architectures for all the models can be found in appendix G.

The training data consists of sensor data as an input, and the object to be reconstructed, p_0, as an output. The sensor data was obtained using wave simulations,

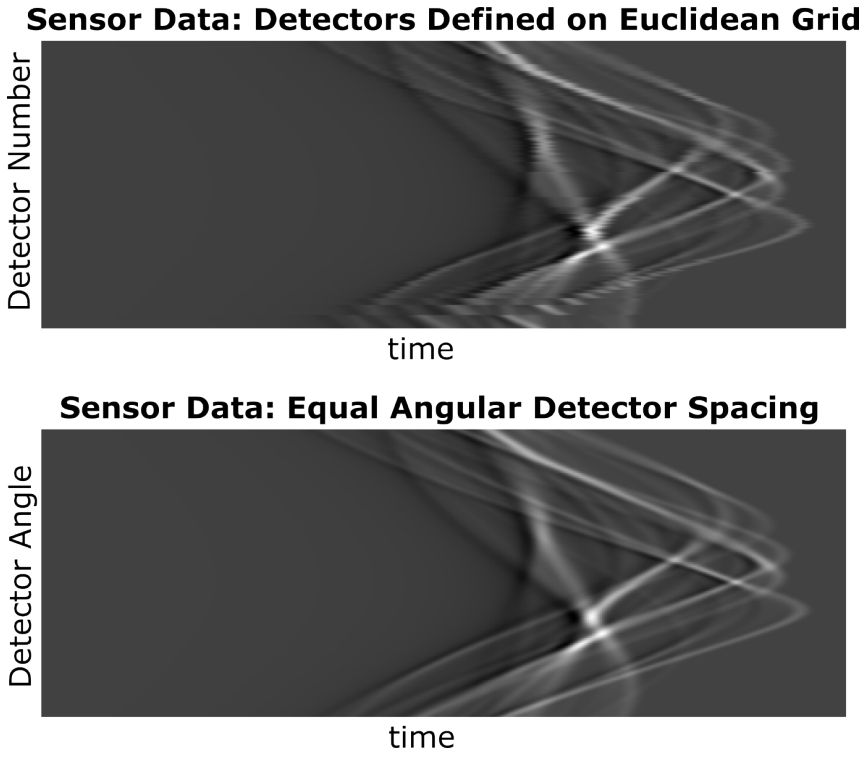


Figure 5.1: In this section, two different data representations are used. For the bottom figure, the detectors are placed along a circle with equal angular spacing, and ordered according to their angle. For the top figure, the detectors are placed approximately along a circle, such that the detector locations can be defined using integer grid coordinates of the grid on which the wave simulations was run, and the detector ordering is not perfectly according to the detector angle (note the staircase effect in the sensor data for the top figure, compared to the smooth appearance for the bottom figure.).

using the same method as in chapter 4. In chapter 4, the wave simulations were performed on a regular grid, thus we only have access to the pressures on this equally spaced grid. This is fine for networks like FCNs and SIRENs, but specifically for CNNs, additional structure could be helpful. Therefore, the simulated pressures are interpolated to a half ring consisting of 171 detectors with a constant radius and equal angular detector spacing (see figure 5.1). This is also closer to the situation of the PAM3 than making a restriction to have detectors located on a regular grid, but the difference is that the detectors are much closer, removing any sparsity in the measurements due to large detector spacings. Now having the correct data representation, the networks can be trained.

Each model is trained, using the training set, for 10000 epochs by optimizing the mean squared error loss function an Adam optimizer, an initial learning rate of 0.0001 and a cosine annealing scheduler that brings the learning rate down to 0. Based on the validation loss, the optimal network architecture can be chosen, and

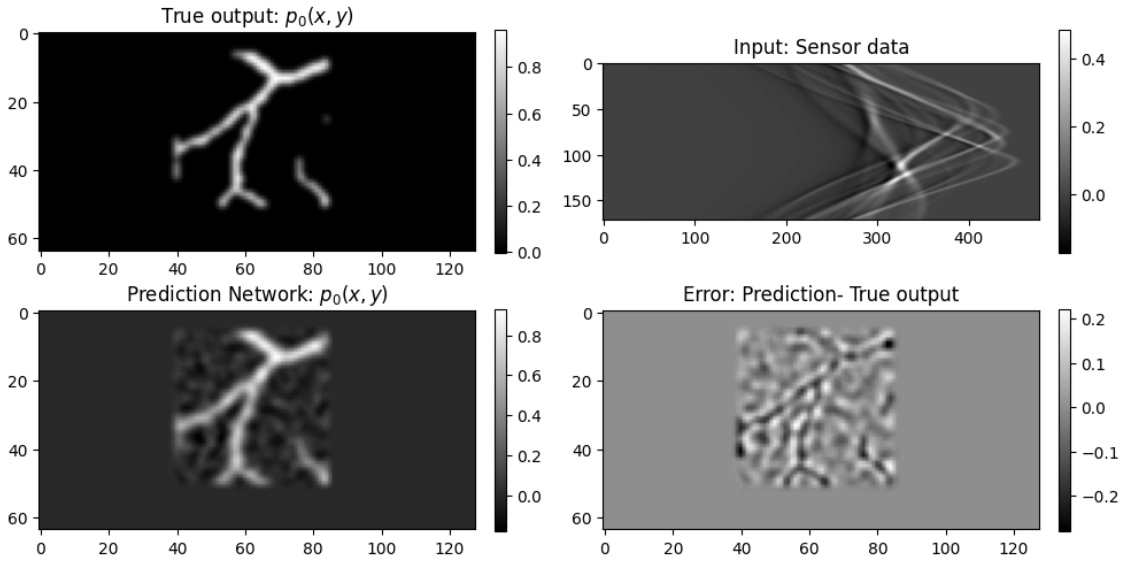
can be tested on the test set.

The validation losses after the final epoch can be found in appendix G. The network that performed the best used an FCN for the branch and a SIREN for the trunk. This output of this network for an image of the test set and an image of the training set is shown in figure 5.2. The average relative mean squared error for the complete test set was 0.3290. A lot of low resolution 'noise' can be observed in both the prediction for the training set as well as for the test set. For the training set, the absolute values of the prediction by the DeepOnet are close to the true output. A difference is that the prediction looks smoother than the ground truth. For the image of the test set, smoothing is even more pronounced. The average time it took to do 50 reconstruction was 0.0207 seconds. This is 8 times faster than the wave simulations of section 4.1, and 100 times faster that k-Wave's wave simulations.

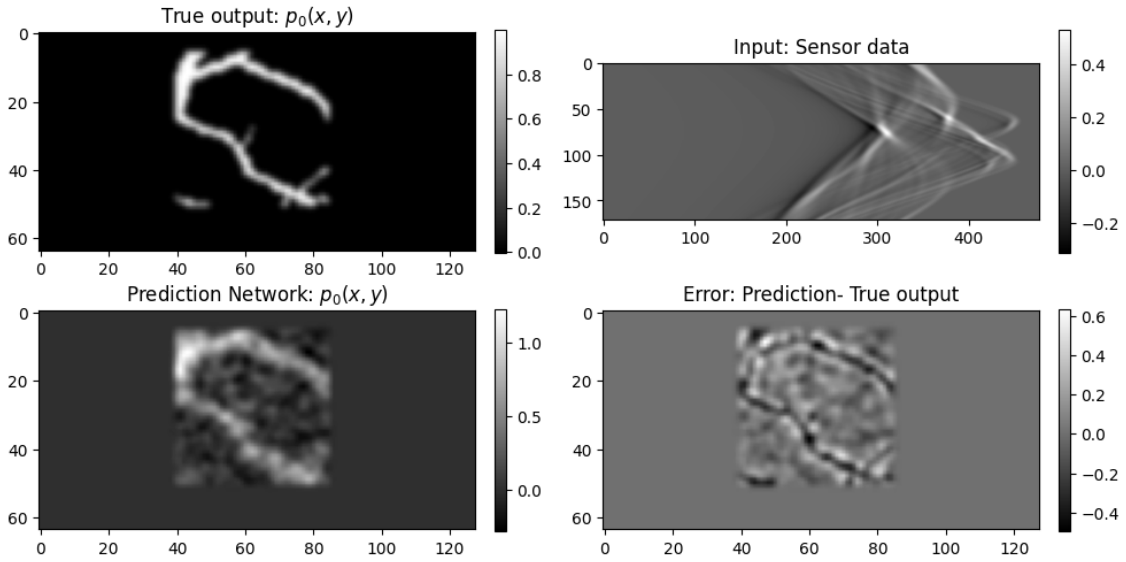
Generalizability to a completely different input, was again tested using the camera image. The result is shown in figure 5.3. The camera man is not clearly visible in the reconstruction, but the low resolution features are predicted correctly.

The super resolution capabilities of the DeepOnet were test by evaluating the DON on a finer grid than the training grid. The result is shown in figure 5.4.

In this section, a first attempt has been made to use DeepOnets as photoacoustic reconstruction operators. Several DeepOnet architectures have been trained, resulting in the optimal architecture that consists of an FCN for the branch and a SIREN for the trunk. The average relative mean squared test error amounted to 0.3290. Compared to the case of iterative time reversal in chapter 4, figure 4.11, this is comparable to the image quality after one iteration of iterative time reversal. However, the FNO-framework for iterative time reversal was able to reach lower errors after only 4 iterations. The biggest problem with the DeepOnet results, are the smoothing, and low frequency patterns in the background. The reason for the appearance of these patterns will be further investigated in the next section. Generalizability of the DeepOnet to the out of distribution image of a camera man was quite poor. Low resolutions features were correctly predicted, but any fine detail was lost. A big advantage of DeepOnets is the speed of the reconstructions. For the small domain of 64 by 128 pixels used in this chapter, the DeepOnet was almost 100 times faster than a single wave simulation in k-Wave. If this increase in speed would still hold for larger domains, and assuming that 10 iterations are necessary for classical photoacoustic reconstruction, which takes 3 hours and 20 minutes for the PAM3 geometry [3], the reconstruction time could be reduced to only a few seconds. Of course, this is just speculation and actual experiment would need to be performed to determine the actual increase in speed. Maybe for reasons, like the need to use multiple GPU's due to the size of the data, or a multi-grid approach, the results of this section do not hold for very large domains and smaller speed ups will be obtained. At least, the results are promising. We have also seen that DeepOnets can be queried at a finer grid than the grid at which the training data was defined. This does not mean finer features are visible, because only the grid resolution is increased. Due to the implicit regularization of the trunks, which are just neural functions, this results in



(a) Training set



(b) Test set

Figure 5.2: Comparison of the DeepONet predictions and ground truth solutions, given sensor data as input, illustrated for examples from both the training and test datasets.

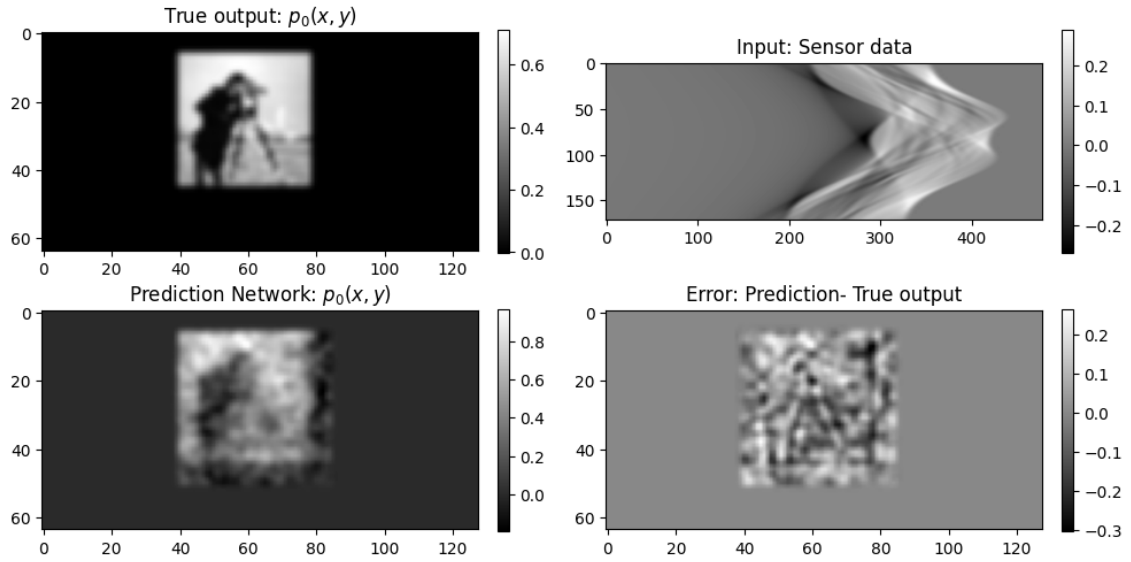


Figure 5.3: The generalizability of the DeepOnet is tested on an image of a cameraman.

Figure 5.4: Given the same sensor data as in figure 5.2, the DeepOnet was evaluated on a fine grid of 256 by 512 pixels.

the same interpolation as was seen in chapter 3.

5.2 A Connection Between DeepOnets and SVD

The photoacoustic reconstructions given by the DeepOnet in section 5.1 are not as good as was hoped. Especially when compared to the FNOs of chapter 4, whose results were close to perfect, the results of the DeepOnet showed unwanted low resolution background patterns, both for prediction on the training data as well as for the unseen test data.

In this section, a connection will be drawn between DeepOnets and SVD, which will be used to get a better understanding of DeepOnets. This will also explain why the previously mentioned artifacts are visible in the photoacoustic reconstructions and it will become clear that the amount of training data that was used, was the limiting factor.

Neural networks make use of nonlinear activation functions. This nonlinearity makes artificial neural networks extremely powerful, but also difficult to analyse. For DeepOnets, this is a bit different. Even though DeepOnets consist of two nonlinear networks, the outputs of both networks, which are vectors of the same length, are combined in a linear way by calculating their dot product, as given by equation 2.6, repeated here:

$$\mathcal{D}\{f\}(x) = \sum_{k=1}^{R} \mathcal{B}_k\{u\}\,\mathcal{T}_k(x)$$

When training a DeepOnet, we are trying to find the optimal basis functions, $\mathcal{T}_k$, and basis coefficients $\mathcal{B}_k$, that best describe the data. After training, the basis functions are fixed, but the coefficients vary based on the input, which is the sensor data. Even though both the trunk and the branch network can be highly nonlinear, they are in the end combined in a linear fashion, which is fundamental to the following argumentation for the maximum accuracy that can be obtained with a DeepOnet.

In section 5.1 the following optimization problem was solved:

$$\underset{\xi}{\operatorname{argmin}} \frac{1}{528} \sum_{i=1}^{528} \|\mathcal{D}_\xi\{f_i\} - p_i\|_2^2 \tag{5.1}$$

where $\mathcal{D}_\xi\{f_i\}$ is the network prediction, ξ are the network parameters and p_i is the true output that needs to be approximated. Define the matrix Π such that each row is the vectorized form of p. The first row is p_1, the second row is p_2, and so on until p_{528}:

$$\Pi = \begin{bmatrix} — \ p_1 \ — \\ — \ p_2 \ — \\ \vdots \\ — \ p_{528} \ — \end{bmatrix} \tag{5.2}$$

Similarly, define the matrix Δ such that each row corresponds to the vectorized form of $\mathcal{D}\{f_i\}$:

$$\Delta = \begin{bmatrix} — \ \mathcal{D}\{f_1\} \ — \\ — \ \mathcal{D}\{f_2\} \ — \\ \vdots \\ — \ \mathcal{D}\{f_{528}\} \ — \end{bmatrix} \tag{5.3}$$

Using equation 2.6, we can write Δ as

$$\Delta = \begin{bmatrix} — \ \mathcal{D}\{f_1\} \ — \\ — \ \mathcal{D}\{f_2\} \ — \\ \vdots \\ — \ \mathcal{D}\{f_{528}\} \ — \end{bmatrix} = \begin{bmatrix} \mathcal{B}_1\{u_1\} & \mathcal{B}_2\{u_1\} & \cdots & \mathcal{B}_R\{u_1\} \\ \mathcal{B}_1\{u_2\} & \mathcal{B}_2\{u_2\} & \cdots & \mathcal{B}_R\{u_2\} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{B}_1\{u_{528}\} & \mathcal{B}_2\{u_{528}\} & \cdots & \mathcal{B}_R\{u_{528}\} \end{bmatrix} \cdot \begin{bmatrix} \mathcal{T}_1 \\ \mathcal{T}_2 \\ \vdots \\ \mathcal{T}_R \end{bmatrix}, \tag{5.4}$$

We can now see that the optimization problem in 5.1 is the same as the following optimization problem:

$$\underset{\xi}{\operatorname{argmin}} \|\Delta_\xi - \Pi\|_F^2, \tag{5.5}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and the ξ dependence of Δ is specifically denoted.

I will now introduce a theorem [57], after which it becomes clear why I have rewritten equation 5.1 to the form of equation 5.5.

Theorem 5.1 (Eckhart-Young Theorem)**.** *Let $A \in^{m\times n}$ be a real matrix with $m \leq n$. The SVD of A is given by $A = U\Sigma V^T$. Let $B \in \mathbb{R}^{m\times n}$ be another real matrix.. The singular values of A are denoted by $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Assume that $rank(B) \leq k$. Then the solution to the minimization problem*

$$\begin{aligned} \min_{B\in\mathbb{R}^{m\times n}} \quad & \|A - B\|_F \\ \textit{subject to} \quad & \operatorname{rank}(B) \leq k \end{aligned}$$

has as a solution the truncated SVD of A, denoted by A_k, i.e.

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = U_k \Sigma_k V_k^T,$$

where U_k, Σ_k, V_k are the matrices formed by taking the first k columns of U and V and the top-left $k \times k$ block of Σ, respectively. The optimal value is given by

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^{m} \sigma_i^2.$$

Proof. See [57]. ■

Seeing the correspondence between equation 5.5 and theorem 5.1, we can explain the limited reconstruction quality in section 5.1:

Given the circumstances in section 5.1, so having a training set consisting of 528 images, and having a DeepOnet architecture for which R=128 (see equation 2.6), and for a moment assuming no limits on $\mathcal{B}$ and $\mathcal{T}$, except for the length of the output vector, then theorem 5.1 can be applied directly. Thus, the optimal performance of the chosen DeepOnet architecture in section 5.1 on the training set is limited to the truncated SVD of equation 5.2. There are several reasons why the optimal accuracy is not reached, like getting stuck in a local minimum and not reaching a global minimum or using a network that is not expressive enough. So when training a DeepOnet, the trunks are trying to approximate the principal components, and the branch is a nonlinear network that is trying to determine the basis coefficients from the sensor data.

The theoretical insight gained, will now be used to see how good or bad the results in section 5.1 actually were. The result of projecting an instance of the training set and an instance of the test set onto the first 128 principal components is shown in figure 5.5. When compared, this figure looks very similar to the output of the DeepOnet in figure 5.2. The theoretical arguments given above thus tell us that the DeepOnet performed really well, and that a better result, given the network architecture, was not possible, and the low resolution background patterns cannot be removed without changing the network architecture. The theoretical discussion tells us exactly how the results can be improved, namely by increasing the number of nodes in the final layer of both networks of the DeepOnet, which means increasing the rank of the matrix to be approximated in theorem 5.1. If all 528 principal components of the training set are used, and an image of the test set is projected onto the space spanned by these vectors, the best possible result that could be obtained is the result shown in figure 5.6. The results do not say that this accuracy will actually be reached. Maybe the branch cannot predict the basis coefficients correctly because there is not enough information in the input data for example.

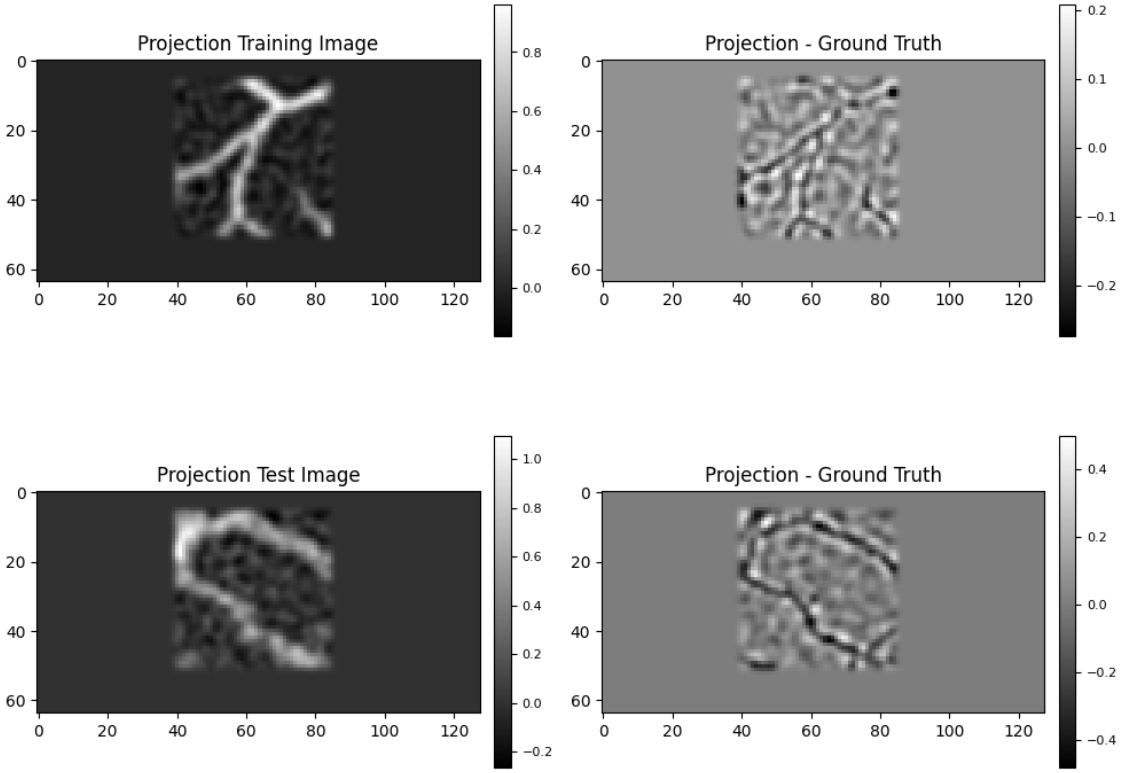


Figure 5.5: Shown here are an image from the training set and an image from the test set that are both projected onto the space spanned by the first 128 principal components of the training set. Also shown, is the difference between the projection and the full image.

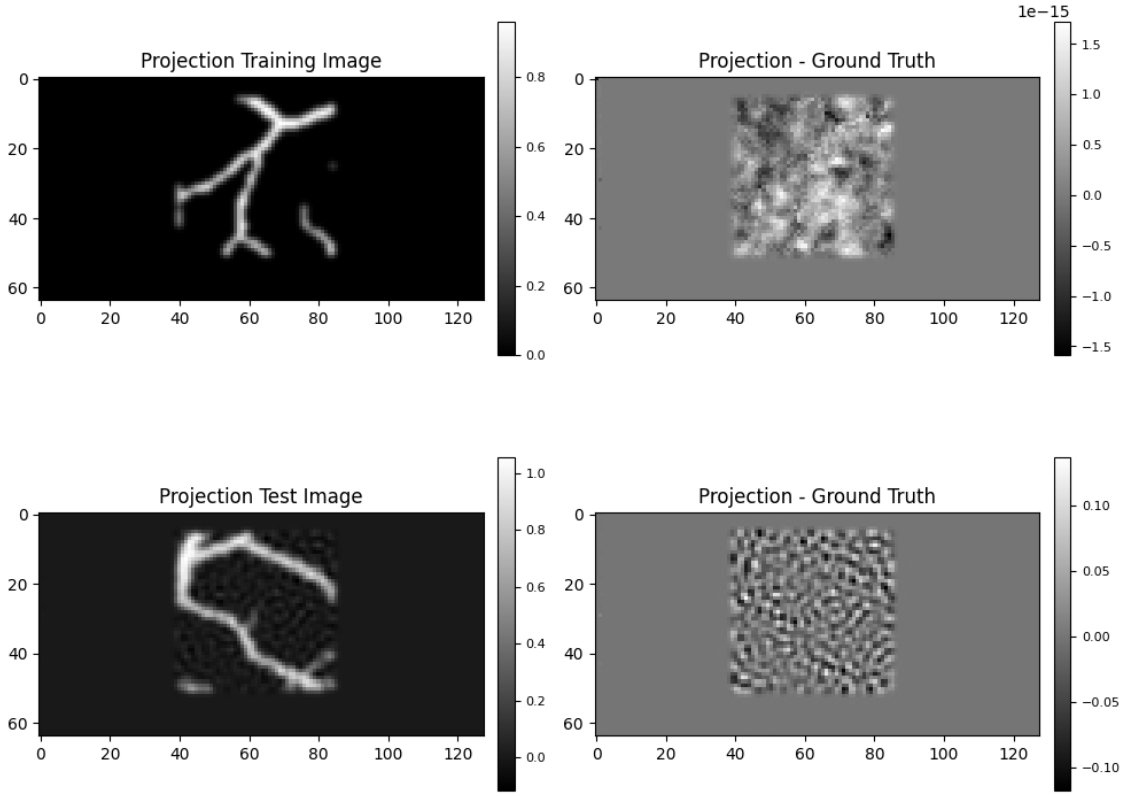


Figure 5.6: The same images as in figure 5.5 are now projected onto the space spanned by the images in the training set. Off course, the error of the image in the training set is equal to zero, up to floating point noise. There is still an error in the image of the test set, since it is not in the span formed by the images in the training set.

5.3 Discussion and Conclusion

In this section, we have seen how DeepOnets can be used for direct inversion. After hyperparameter optimization for 4 different combinations of trunk-branch networks, a DeepOnet architecture made from an FCN and a SIREN was found to perform the best as a photoacoustic reconstruction operator. The speed up compared to a single wave simulation in k-Wave of a factor 100 was obtained. A downside was the relatively large error. Especially when compared to the iterative results of chapter 4, where a better result was obtained after only 2 iterations.

A theoretical argument has been given to predict the optimal network output. This insight of a connection between DeepOnets and SVD also shows us how, in theory, a better result can be obtained. Increasing the number of trunk and branch outputs, corresponding to an increase in the rank of the matrix to be approximated, can improve the results but more data might be required to do this to reach an acceptable accuracy. From the connection with SVD, it is clear that DeepOnets are best suited for approximating sparse operators, which is also reported in the literature [25]. The photoacoustic reconstruction operator is not sparse, as is also suggested by the experimental results in this section, which means that DeepOnets are probably not the most suitable type of networks for direct inversion.

DeepOnets can be evaluated at any grid coordinate after training. This is similar to the neural functions from 3. True super resolution, in the sense of being able to see finer details, is not obtained, only the same interpolation resulting from the implicit regularization of the SIREN architecture. Maybe, there is an alternative use for DeepOnets in the context of photoacoustic imaging though. Just like neural functions, of which PINNs are one example, derivatives with respect to input coordinates can be easily calculated. Maybe, DeepOnets can be used for the task of solving the wave equation, as was done in chapter 4, followed by optimizing the resulting coordinate based network like a PINN. The physics based loss term that is then added, and can give information for non-integer grid coordinates, can be a first step to true super resolution outputs.

Chapter 6

Discussion

Many different results have been obtained in this thesis. Most of the results were already discussed at the end of each chapter. Here I will only discuss the most significant results.

First of all, the use of NFs for data interpolation has been investigated. The use of NFs did not improve the reconstruction quality significantly. Self-supervised techniques do not 'know' what the usual data looks like, and cannot use any information to base its interpolation properties on, other than using some smooth interpolation. The use of NFs for data inpainting in the context of photoacoustic imaging is therefore very limited. Learned techniques are probably a better alternative for data inpainting, although these learned methods are difficult to realize in practice for the computational volumes considered for the PAM3.

The use of FNOs showed some impressive speed ups for wave simulations compared to k-Wave and the quality and generalizability were excellent. Unfortunately, only small domains of 64 by 128 pixels were considered in this thesis. Even these small domains required the use of large 48 GB GPUs for training. This has to do with the data inefficient representation that is required for FNOs. This makes FNOs, as used in the report, unsuitable for larger PAM3 geometries. Perhaps multigrid approaches [51] could offer a solution and might be worth investigating. It is not known wether the results regarding speed ups are still true for larger geometries, methods like the multigrid approach work very differently than the methods explored in this thesis.

For the first time, FNOs were used to simulate waves on domains with an inhomogeneous speed of sound. Good results were obtained, but there is still room for improvement. Increasing the complexity of the FNO will probably provide excellent results. Iterative reconstruction using FNOs also showed some good results, but again, there is room for improvement. Error build up after a few iterations, made results unusable. Methods like algorithm unrolling [50], can probably solve these problems.

It was disappointing to see that FNOs are not actually discretization invariant, since this is the reason why FNOs would be special and belong to the class of

neural operators. The FNO architecture is not very different from a standard CNN architecture, but with skip connections. It would be interesting to compare the two in future work. The claims that FNOs are very different from CNNs [13] are therefore not fair. Maybe, instead of using the term 'neural operator' it is better to classify networks in the future based on whether they are 'representation equivalent' a term introduced by Bartolucci et al. [52], meaning that operations performed on discrete data have the same effect as on continuous data.

Approximating the photoacoustic reconstruction operator with DeepOnets gave results of poor quality on first sight. A theoretical analysis showed that the results were close to being as good as possible, given the network architecture. DeepOnets are known to perform well for sparse problems, and photoacoustic inversion is definitely not sparse, as was seen by the low quality reconstruction results. Thanks to the theoretical results, it is known how to improve the results, namely increasing the network size, and using more data. For the PAM3 system, this might mean that huge amounts of data are necessary, making the usability of DeepOnets limited. Even though the quality was not great, the speed up of a factor 100 was very impressive. Perhaps even lower quality results are useful as a better initialize for iterative algorithms, so that they require fewer iterations. So, the use of DeepOnets as photoacoustic reconstruction operators seems limited, but there might be some applications for which they are useful.

Chapter 7

Conclusion and Outlook

The central question of this thesis was whether continuous neural representations can be used to accelerate photoacoustic image reconstruction and improve reconstruction quality in the case of sparse measurements. DeepONets and FNOs have been shown to be significantly faster than k-Wave; however, these methods currently have several limitations. In their present form, they are not yet suitable for practical use in photoacoustic image reconstruction. Further research into applying deep learning techniques to large computational domains may lead to meaningful advancements, as the current results for small domains, while promising, have limited practical value. Moreover, despite claims that these techniques provide continuous representations of objects rather than relying on discretization, this thesis finds that such continuity does not hold under the conditions investigated.

Appendices

Appendix A

AI Statement

During the preparation of this work, I used ChatGPT-3.5 and ChatGPT-4 to write and improve Python and Matlab code and to rewrite parts of this thesis. After using this tool, I thoroughly reviewed and edited the content as needed, taking full responsibility for the final outcome.

Appendix B

Reducing Data Sparsity by a Factor of 2

Theorem B.1 (Projection-Slice Theorem)**.** *The 1-dimensional Fourier transform of the projection data for a certain angle θ, with respect to the variable l, is equal to the 2-dimensional Fourier transform of the image along the line with that same angle θ, i.e.*

$$\mathcal{F}_{1D}\{g\}(\rho,\theta) = \mathcal{F}_{2D}\{f\}(\rho\cos(\theta), \rho\sin(\theta))$$

Proof. See [35]. ■

With the help of theorem B.1, we can proof the following proposition:

Proposition B.2. *Let's say that we have used filtered back projection using the sinogram data, $g(l,\theta)$, for $l \in \mathbb{R}$, and discrete angles $\theta \in [0, \Delta\theta, 2\Delta\theta, \ldots, \pi - \Delta\theta]$, where the angular spacing is given by $\Delta\theta = \frac{\pi}{N_\theta}$, with $N_\theta \in \mathbb{N}$ being the number of detector angles for which a measurement is performed, to obtain an approximation $f^*(x,y)$, of the object $f(x,y)$. A reconstruction that is obtained after artificially increasing the number of measurements by a factor of 2, using linear interpolation as given by equation 3.2, denoted by $f_{intp,2}$ is given by:*

$$f_{intp,2}(x,y) = \frac{1}{2}f^*(x,y) + \frac{1}{4}\mathcal{R}_{\frac{\Delta\theta}{2}}\{f^*(x,y)\} + \frac{1}{4}\mathcal{R}_{-\frac{\Delta\theta}{2}}\{f^*(x,y)\} \tag{B.1}$$

Proof. Using equation 3.2 and the linearity of the Fourier transform and theorem B.1, we know that

$$\mathcal{F}_{1D}\left\{g_{intp}\right\}\left(\rho, \frac{2n-1}{2}\Delta\theta\right) = \frac{1}{2}\mathcal{F}_{1D}\left\{g\right\}\left(\rho, (n-1)\Delta\theta\right) + \frac{1}{2}\mathcal{F}_{1D}\left\{g\right\}\left(\rho, n\Delta\theta\right),$$
$$\text{for} \quad n = 1, \ldots, N_\theta$$
$$\begin{aligned}
&\mathcal{F}_{2D}\left\{f_{intp}\right\}\left(\rho\cos\left(\frac{2n-1}{2}\Delta\theta\right), \rho\sin\left(\frac{2n-1}{2}\Delta\theta\right)\right) \\
&= \frac{1}{2}\mathcal{F}_{2D}\left\{f\right\}\left(\rho\cos\left((n-1)\Delta\theta\right), \rho\sin\left((n-1)\Delta\theta\right)\right) \\
&\quad + \frac{1}{2}\mathcal{F}_{2D}\left\{f\right\}\left(\rho\cos\left(n\Delta\theta\right), \rho\sin\left(n\Delta\theta\right)\right) \\
&= \frac{1}{2}\mathcal{R}_{-\frac{\Delta\theta}{2}}\left\{\mathcal{F}_{2D}\left\{f\right\}\right\}\left(\rho\cos\left(\frac{2n-1}{2}\Delta\theta\right), \rho\sin\left(\frac{2n-1}{2}\Delta\theta\right)\right) \\
&\quad + \frac{1}{2}\mathcal{R}_{\frac{\Delta\theta}{2}}\left\{\mathcal{F}_{2D}\left\{f\right\}\right\}\left(\rho\cos\left(\frac{2n-1}{2}\Delta\theta\right), \rho\sin\left(\frac{2n-1}{2}\Delta\theta\right)\right)
\end{aligned}$$

Applying the frequency domain filter as given in definition 3.4 and then applying the inverse Fourier transform and using proposition 3.10 gives, in polar coordinates,

$$f_{intp}(r,\theta) = \frac{1}{2}\mathcal{R}_{-\frac{\Delta\theta}{2}}\left\{f^*\right\}(r,\theta) + \frac{1}{2}\mathcal{R}_{\frac{\Delta\theta}{2}}\left\{f^*\right\}(r,\theta)$$

The last step to get to the final result is, knowing that have obtained a sinogram with twice the number of samples according to equation 3.2, and realizing that

$$\mathcal{F}\left\{f_{intp,2}\right\}(\rho,\theta) = \frac{1}{2}\mathcal{F}\left\{f^*\right\}(\rho,\theta) + \frac{1}{2}\mathcal{F}\left\{f_{intp}\right\}(\rho,\theta),$$

which directly gives us the desired result.

■

A quick experiment is performed to show the effect of linear interpolation as predicted by proposition B.2. The radon transform for 30 equally spaced angles from 0 to 180 degrees is applied to a phantom consisting of points placed along circles of different radii. The phantom, the filtered back-projected reconstruction and the filtered back-projected results with interpolated measurements are shown in figure B.1.

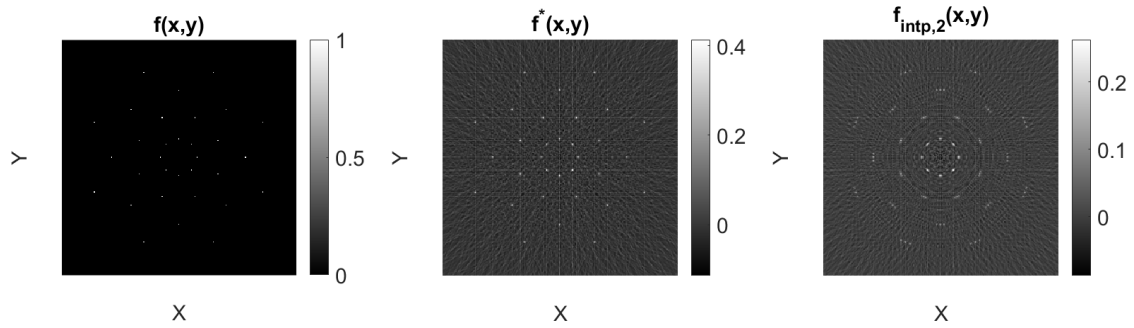


Figure B.1: This figure shows, from left to right, a phantom consisting of multiple points, its reconstruction obtained with filtered backprojection and the reconstruction obtained with filtered backprojection after increasing the number of measurements by a factor of 2 using linear interpolation.

Figure B.1 shows us how interpolation gives a reconstruction that contains two copies of the reconstruction without interpolation, but then rotated by a small amount clockwise and counter-clockwise.

Appendix C

Generation of Training Data

GIGO: Garbage In, Garbage Out

Attributed to George Fuechsel, an American IBM technical instructor.

The use of high-quality training data is of utmost importance when training a neural network to achieve accurate predictions on unseen data. In this appendix I will explain how the data, used for training the neural networks that act as surrogate models for classical wave propagation solvers, was generated.

C.1 Initial Pressure Distribution

The first thing we need if we want to solve the acoustic part of the photoacoustic forward problem, as depicted in figure 2.1, is an initial pressure distribution. To stay close to the application of interest, i.e. PAM, pressure distributions with a blood vessel-like shape will be used. The Retinal Vascular Tree Analsysis (RETA) database [58] will be used for this. The RETA database contains 54 segmentations of retinal blood vessels with a size of 1024 by 1024 pixels. One of the images in the dataset is shown in figure C.1. These images are too large for the experiments performed, therefore patches with a size of 128 by 128 pixels were extracted randomly from the images, where the outer 100 pixels of the original images were removed, because they contained no blood vessels. For the first 44 images, 12 patches were extracted per image, resulting in a total of 540 images in the training set. Another 3 patches were extracted per image to generate a validation set containing 132 images. For each of the 10 remaining images, 12 patches were also extracted to generate a test set containing 120 images. The patches were later reduced in size to 64 by 64 pixels using nearest-neighbor interpolation. Some of the patches are shown in figure C.2. The generated patches can be used as initial pressure distributions and propagated using an acoustic solver as will be explained in the next section.

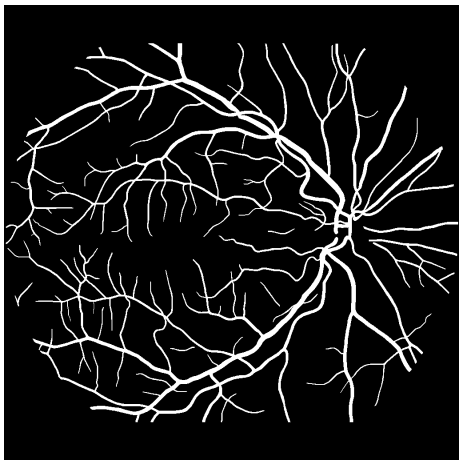


Figure C.1: An example of one of the images in the RETA dataset, showing a mask of retinal blood vessels.

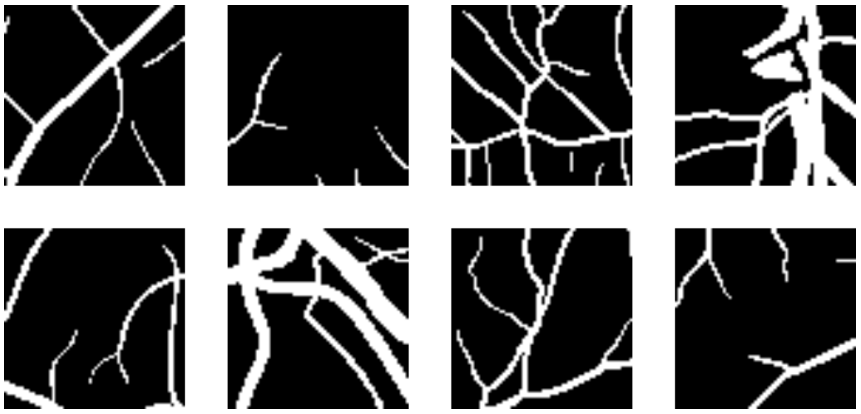


Figure C.2: Eight examples of patches with a size of 64 by 64 pixels, which will be used as initial pressure distributions for the wave propagation simulations.

C.2 Wave Propagation

In order to simulate the acoustic part of the photoacoustic forward problem, we need to be able to solve the wave equation. A popular toolbox for acoustic wave simulations is k-wave [38], an open source Matlab wave propagation tool. It takes too much space in this report to explain exactly how this toolbox works, and what the advantages and disadvantages are, but the interested reader can find detailed explanations in the k-wave user manual [59]. There are a few things written in the user manual which are important to mention though.

The first thing is that that the solutions given by the discrete k-space pseudospectral method implemented in k-wave converges to the analytical continuous solutions as the grid spacings go to zero. An easy way to test the accuracy of the solution is therefore to test how the solution changes for finer discretizations.

Secondly, k-wave assumes that solutions are periodic, i.e. the domain is a hypertorus, $\mathbb{T}^n = \underbrace{\mathbb{S}^1 \times \cdots \times \mathbb{S}^1}_{n}$, with n being the dimension of the domain. The result is that waves that leave the bottom of the domain will reappear at the top, if no action is taken to compensate for this effect. One way of solving this problem is expanding the domain, and limiting the time of the simulations to prevent this wrap-around artifact. A downside is the increased computational effort. Another method, implemented by k-wave, is adding a perfectly matched layer (PML) layer to the boundary of the domain. The PML layer is an absorbing layer whose properties are described by a set of non physical equations, and has as a goal absorbing all the incident waves and preventing reflections. It is important to note that PML layers are not perfect and they are a major course of limiting the accuracy to approximately 10^{-4} or 10^{-5} at best.

The third and last thing to consider is the maximum spatial frequency that can be represented on the computational grid and how this relates to the continuous function represented in a discretized manner. According to the Nyquist sampling theorem, the sampling frequency needs to be at least twice the maximum frequency present in the signal. K-wave makes the assumption that the continuous function, $\hat{p}(x)$, that gave rise to the discretized values, is given by the band-limited interpolant, given by

$$\hat{p}(x) = \frac{1}{N_x} \sum_{m=-N_x/2}^{N_x/2-1} P(k_m) e^{-\frac{2\pi i}{N_x}\frac{mx}{\Delta x}} \tag{C.1}$$

where $P(k_m)$ are the discrete Fourier coefficients. A result of assuming that the continuous function is given by equation C.1 is that unexpected oscillations can appear in the numerical solution (think of convolution with a sinc function)[9]. It is preferred to have smooth solutions without the oscillations, therefore a Blackman

[9]It is important to realize that these oscillations do not mean that the results of the wave simulations are wrong or unstable!

window is used to filter the initial pressure distribution in the spatial frequency domain. This prevents the oscillations from appearing.

I have just summarized the most important properties of k-wave and the challenges it comes with. Having this knowledge in the back of our mind, we can now generate the training data necessary to train a neural network that acts as a wave solver. In summary, the following steps are taken to generate the training data[10]:

1. A computational domain is created.
2. A 64 by 64 sized patch, filled with blood vessels, is placed in the computational domain and will act as the initial pressure distribution.
3. The initial pressure distribution is smoothened using a Blackman window.
4. Either a PML layer is placed outside the computational domain or the computational domain is expanded to twice the original size by expanding it in all directions followed by placing an PML layer outside the boundary of the domain.
5. K-wave is used to solve the wave-propagation problem until a time equal to the maximum time it takes to travel across the original computational domain.

The figure below shows a graphical depiction of the 5 steps:

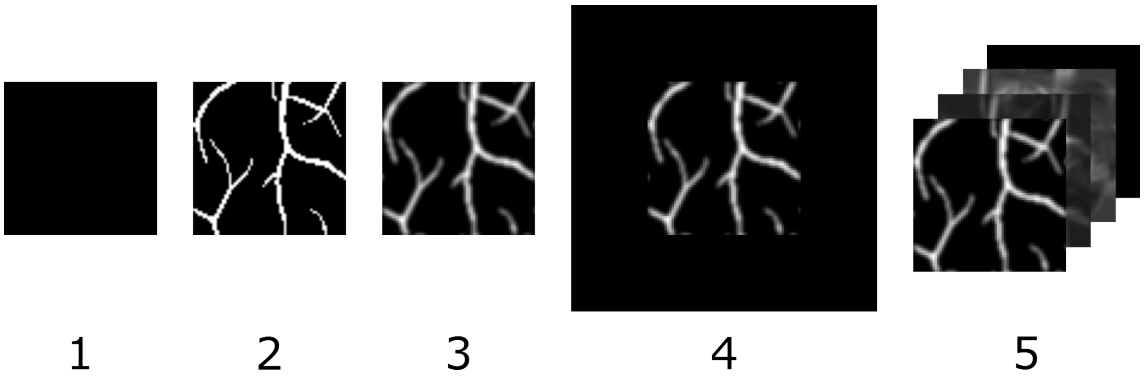


Figure C.3: The data used for training is generated by (1) creating a computational domain, (2) inserting an initial pressure distribution in the shape of blood vessels, (3) smoothing using a Blackman window, (4) expanding the domain/adding a PML layer and (5) solving the wave propagation problem using k-wave.

Before the pipeline was used to generate all the training data, the accuracy of the solution was tested on one instance by increasing the size of the computational domain by a factor of 2 and 4 and using Fourier interpolation to increase the size

[10]For some experiments in this report, the steps may vary slightly, and any differences will be noted in the relevant sections.

of the initial pressure distribution and running k-wave. The difference between the original results and the results using a factor 2 increase of domain size were less then the floating point accuracy. The same result was obtained by comparing the results for the factor 2 increase with the results for the factor 4 increase, and the results for the factor 4 increase with the results for the factor 8 increase, allowing us to assume that the results are accurate.

Appendix D

Extra iterations for figure 4.11

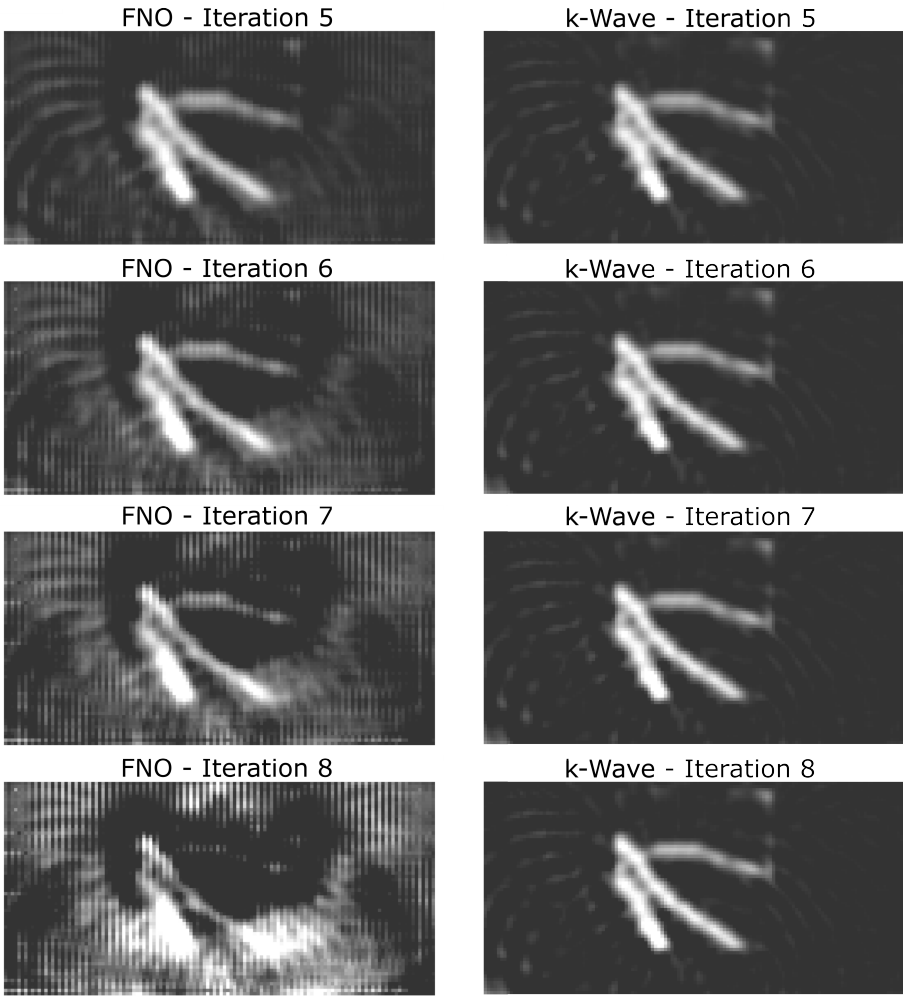


Figure D.1: This figure shows that the error build up of the FNO produces results that are unusable for the 5th iteration and beyond.

Appendix E

Full output of figure 4.14

The full resolution output related to figure 4.14 is shown below:

Figure E.1: This figure shows the output of an 128 by 256 input image for an FNO trained on 64 by 128 sized images

Appendix F

FNO as a Direct Inversion Operator

Several experiments were performed for the use of FNOs as direct inversion operators. None of the experiments resulted in good outcomes. Shown below is one of the results, which is a good example, and all other results were very similar. An FNO was trained using sensor-data as an input, and a stack of initial pressures as an output. The trained network would give the same output, no matter the input. The output is shown below. It was almost as if the network was not sensitive to the input data, even though tiny variations based on sensor data were present. For a training set of just 4 images and a network with millions of parameters, the result was similar. The network would always get stuck in a local minimum. It is known that FNOs work well for problems that can be described easily in k-space, and this is probably not the case for photoacoustic inversion, which could explain the poor results

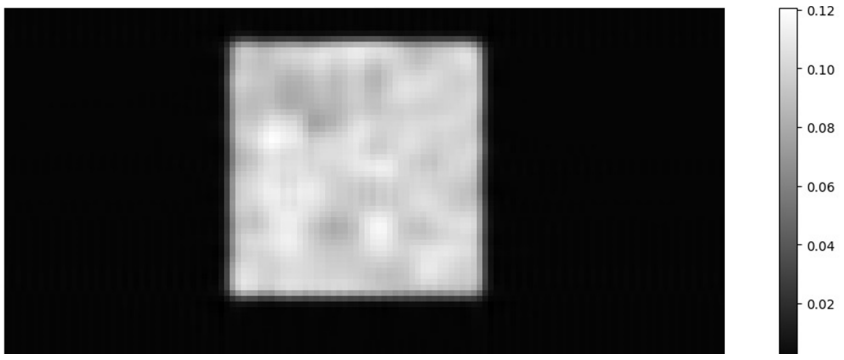


Figure F.1: The output of the FNO trained for direct inversion gave the average of all inputs, shown as the white square within which all the blood vessel images were defined.

Appendix G

DeepOnet Architectures

In section G.1, G.2, G.3, and G.4, the hyperparameters are given that define the DeepOnet architectures. Note that to each DeepOnet, a single trainable bias term is added after the dot product. The best performing network architectures for each of the four trunk-branch combinations, is made bold. The validations losses are given in section G.5.

G.1 FCN-FCN

Net #	Branch Layers	Branch Hidden Nodes	Branch Nodes Last Layer	Trunk Layers	Trunk Hidden Nodes	Trunk Nodes Last Layer
1	2	256	128	7	1024	128
2	2	32	128	2	256	128
3	5	128	64	7	1024	64
4	6	64	512	4	256	512
5	3	256	128	2	1024	128
6	5	512	64	6	256	64
7	3	1024	64	3	512	64
8	5	128	64	3	32	64
9	3	128	128	2	512	128
10	5	32	128	5	1024	128
11	6	512	512	7	1024	512
12	7	256	32	4	256	32
13	5	256	32	3	512	32
14	4	1024	512	3	512	512
15	6	1024	64	5	1024	64
16	6	128	512	3	32	512

Net #	Branch Layers	Branch Hidden Nodes	Branch Nodes Last Layer	Trunk Layers	Trunk Hidden Nodes	Trunk Nodes Last Layer
17	6	512	128	7	64	128
18	2	512	256	2	1024	256
19	5	512	128	3	1024	128
20	4	128	32	5	1024	32
21	2	32	128	4	256	128
22	3	32	32	3	32	32
23	4	512	256	7	64	256
24	5	32	512	3	32	512
25	**5**	**1024**	**256**	**6**	**256**	**256**
26	5	32	32	6	256	32
27	4	128	1024	3	1024	1024
28	7	64	64	4	512	64
29	7	1024	32	5	32	32
30	3	512	64	7	512	64
31	4	32	1024	6	64	1024
32	7	32	32	7	128	32
33	2	64	512	2	64	512
34	6	512	32	6	256	32
35	4	1024	512	4	32	512
36	4	128	64	7	128	64
37	2	1024	32	6	64	32
38	5	64	128	2	64	128
39	7	64	1024	2	256	1024
40	6	32	32	6	64	32

G.2 FCN-SIREN

Net #	Branch Layers	Branch Hidden Nodes	Branch Nodes Last Layer	Trunk Layers	Trunk Hidden Nodes	Trunk Nodes Last Layer	Omega
1	5	1024	32	4	64	32	200
2	6	1024	128	7	32	128	40
3	4	64	256	7	256	256	40
4	**3**	**1024**	**128**	**5**	**1024**	**128**	**20**
5	7	128	32	2	128	32	70
6	5	128	128	5	1024	128	70

Net #	Branch Layers	Branch Hidden Nodes	Branch Nodes Last Layer	Trunk Layers	Trunk Hidden Nodes	Trunk Nodes Last Layer	Omega
7	5	128	32	7	128	32	100
8	5	256	512	6	32	512	200
9	5	128	512	1	512	512	100
10	4	512	256	2	128	256	70
11	4	128	128	7	256	128	70
12	4	128	1024	3	1024	1024	100
13	6	256	128	6	32	128	20
14	2	64	512	7	1024	512	70
15	4	128	256	6	256	256	200
16	6	1024	512	2	128	512	70
17	3	32	32	3	1024	32	200
18	2	1024	64	7	32	64	200
19	7	64	256	2	32	256	200
20	7	32	32	3	128	32	40
21	6	256	32	4	128	32	70
22	3	1024	64	4	512	64	100
23	6	1024	128	3	512	128	20
24	3	256	64	5	1024	64	70
25	6	128	128	4	128	128	70
26	3	256	128	2	1024	128	40
27	4	512	64	2	32	64	20
28	7	32	1024	7	128	1024	40
29	2	128	512	4	1024	512	20
30	6	1024	128	3	64	128	70
31	7	1024	64	3	256	64	200
32	2	32	512	7	32	512	200
33	2	512	1024	2	32	1024	100
34	7	64	64	5	64	64	70
35	5	128	32	5	256	32	40
36	5	1024	512	6	32	512	200
37	3	1024	32	2	512	32	100
38	2	64	256	4	512	256	20
39	7	64	1024	4	128	1024	40
40	7	512	256	5	1024	256	40

G.3 CNN-FCN

Branch:

Net #	CNN Layers	Kernel Size	Channels 8/16	Linear Layers	Nodes	Nodes Last Layer
1	5	3x3	16	2	64	256
2	3	5x5	8	3	128	256
3	3	3x3	8	2	128	256
4	4	3x3	8	2	256	1024
5	3	5x5	8	3	64	1024
6	4	3x3	8	2	256	256
7	5	5x5	16	2	128	1024
8	5	5x5	8	2	64	512
9	4	5x5	16	3	64	512
10	5	5x5	8	2	128	512
11	5	3x3	16	3	128	512
12	3	3x3	16	2	64	512
13	3	3x3	16	3	128	256
14	4	3x3	16	2	256	1024
15	5	3x3	8	2	64	1024
16	5	5x5	8	2	64	512
17	**5**	**3x3**	**16**	**3**	**256**	**1024**
18	3	3x3	16	3	256	1024
19	3	5x5	8	2	256	512
20	3	3x3	16	2	128	512
21	4	5x5	8	3	64	1024
22	5	3x3	8	3	64	1024
23	4	5x5	16	3	256	512
24	3	3x3	8	2	128	256
25	3	5x5	8	3	256	1024
26	4	3x3	8	3	128	256
27	4	3x3	16	2	128	256
28	3	5x5	8	3	128	256
29	4	3x3	8	2	128	256
30	3	5x5	8	2	256	512
31	4	3x3	8	3	256	512
32	3	3x3	8	2	64	512
33	5	3x3	16	3	256	1024
34	4	3x3	16	3	64	512
35	3	3x3	16	3	64	512
36	4	3x3	16	3	128	1024
37	3	5x5	8	2	128	256
38	4	5x5	8	3	128	1024
39	4	3x3	16	2	64	256
40	3	3x3	16	2	256	1024

Trunk:

Net #	Number Layers	Nodes Hidden Layers	Nodes Last Layer
1	3	256	256
2	3	128	256
3	2	128	256
4	3	64	1024
5	4	64	1024
6	2	64	256
7	3	256	1024
8	3	256	512
9	3	256	512
10	2	128	512
11	3	128	512
12	2	128	512
13	2	64	256
14	2	128	1024
15	4	64	1024
16	4	128	512
17	**4**	**256**	**1024**
18	4	256	1024
19	4	64	512
20	4	128	512
21	3	128	1024
22	4	64	1024
23	2	256	512
24	4	128	256
25	3	64	1024
26	3	64	256
27	4	64	256
28	2	256	256
29	3	64	256
30	3	64	512
31	3	128	512
32	3	256	512
33	4	128	1024
34	4	64	512
35	2	128	512
36	2	128	1024
37	2	256	256
38	3	128	1024
39	3	256	256

Net #	Number Layers	Nodes Hidden Layers	Nodes Last Layer
40	4	64	1024

G.4 CNN-SIREN

Branch:

Net #	CNN Layers	Kernel Size	Channels 8/16	Linear Layers	Nodes	Nodes Last Layer
1	4	5x5	8	2	64	512
2	4	5x5	16	3	128	1024
3	3	3x3	16	3	256	512
4	5	3x3	8	3	128	256
5	4	5x5	16	3	64	512
6	4	5x5	16	2	256	512
7	5	5x5	16	3	256	1024
8	5	5x5	8	3	64	1024
9	5	3x3	8	2	64	256
10	4	5x5	16	3	256	256
11	5	5x5	8	3	64	1024
12	3	3x3	16	3	64	256
13	4	5x5	16	2	128	512
14	3	3x3	16	3	128	512
15	5	3x3	8	2	128	256
16	4	5x5	8	3	256	512
17	5	5x5	16	2	256	1024
18	5	3x3	16	2	64	1024
19	4	5x5	16	3	256	512
20	4	5x5	8	3	64	512
21	5	3x3	16	3	64	256
22	5	3x3	16	2	128	256
23	**5**	**3x3**	**16**	**2**	**256**	**512**
24	3	3x3	8	3	256	512
25	5	5x5	16	3	256	1024
26	3	5x5	16	2	128	1024
27	4	5x5	8	3	256	256
28	5	5x5	16	2	128	1024
29	3	3x3	8	3	256	256
30	4	3x3	16	3	256	256
31	4	3x3	8	2	256	256
32	3	3x3	8	2	256	256

Net #	CNN Layers	Kernel Size	Channels 8/16	Linear Layers	Nodes	Nodes Last Layer
33	3	5x5	8	3	64	512
34	5	3x3	8	2	256	1024
35	5	3x3	8	3	256	1024
36	4	5x5	8	2	64	512
37	5	3x3	8	3	128	1024
38	4	3x3	8	2	64	512
39	4	5x5	8	2	64	512
40	4	5x5	16	2	128	1024

Trunk:

Net #	Number Layers	Nodes Hidden Layers	Nodes Last Layer	Omega
1	2	128	512	40
2	3	128	1024	200
3	3	256	512	100
4	4	64	256	70
5	4	64	512	100
6	4	256	512	100
7	4	256	1024	40
8	4	128	1024	40
9	4	128	256	40
10	4	256	256	70
11	4	256	1024	20
12	4	64	256	70
13	2	128	512	20
14	3	64	512	200
15	4	256	256	70
16	4	64	512	200
17	4	256	1024	70
18	4	64	1024	70
19	2	128	512	20
20	4	256	512	70
21	2	64	256	40
22	2	256	256	40
23	**4**	**64**	**512**	**40**
24	3	128	512	20
25	2	128	1024	200
26	2	64	1024	20
27	4	64	256	200
28	3	128	1024	40

Net #	Number Layers	Nodes Hidden Layers	Nodes Last Layer	Omega
29	4	256	256	70
30	2	256	256	20
31	2	64	256	20
32	2	128	256	20
33	3	64	512	70
34	3	256	1024	70
35	2	256	1024	70
36	4	128	512	200
37	2	256	1024	70
38	4	64	512	200
39	4	256	512	100
40	3	64	1024	100

G.5 Validation Losses

Below, the validation losses for all network architectures are presented. The best performing network (in bold and with an underscore) is network 4 for the FCN-SIREN combination. For the other branch-trunk combinations, the best validation losses are highlighted in bold.

#Net	FCN-FCN	FCN-SIREN	CNN-FCN	CNN-SIREN
1	0.00466	0.00867	0.00858	0.00780
2	0.01024	0.00613	0.00784	0.00994
3	0.00725	0.01512	0.00957	0.01428
4	0.00733	**0.00418**	0.00690	0.00553
5	0.00876	0.00800	0.00848	0.00868
6	0.00502	0.01051	0.00898	0.00525
7	0.00551	0.00821	0.00641	0.01105
8	0.01125	0.01029	0.00781	0.00901
9	0.00881	0.01024	0.00746	0.00766
10	0.01007	0.00872	0.00821	0.01012
11	0.01564	0.01100	0.00797	0.00828
12	0.00712	0.01798	0.00962	0.00854
13	0.00706	0.00846	0.00974	0.00611
14	0.00555	0.01006	0.00848	0.00872
15	0.00841	0.01153	0.00950	0.00616
16	0.00907	0.01453	0.00776	0.00498
17	0.00683	0.01043	**0.00521**	0.00593
18	0.00781	0.01084	0.00556	0.00925

#Net	FCN-FCN	FCN-SIREN	CNN-FCN	CNN-SIREN
19	0.00480	0.01140	0.00688	0.00586
20	0.00736	0.01073	0.00910	0.01037
21	0.00934	0.00856	0.00743	0.00875
22	0.01024	0.00570	0.00795	0.00634
23	0.00723	0.00676	0.01647	**0.00430**
24	0.00977	0.00766	0.00851	0.00925
25	**0.00444**	0.00821	0.00682	0.00944
26	0.00884	0.00846	0.00878	0.00776
27	0.00561	0.01018	0.00779	0.00503
28	0.00785	0.01505	0.00874	0.00564
29	0.00932	0.00764	0.00917	0.01188
30	0.00524	0.00483	0.00817	0.00526
31	0.01132	0.00709	0.00728	0.00666
32	0.01015	0.06069	0.00995	0.00847
33	0.00976	0.00705	0.00600	0.01188
34	0.00730	0.00852	0.00852	0.00440
35	0.00828	0.00827	0.00936	0.00487
36	0.00773	0.00756	0.00793	0.00964
37	0.00739	0.00758	0.00930	0.00689
38	0.01057	0.00944	0.00731	0.00879
39	0.00832	0.01323	0.00864	0.00756
40	0.01149	0.01286	0.00882	0.00595

Abbreviations

CNN	Convolutional Neural Netwok
DeepOnet	Deep Operator Network
FCN	Fully Connected Network
FNO	Fourier Neural Operator
FOV	Field of View
INR	Implicit neural representation
NF	Neural Function
PA	Photoacoustic
PAI	Photoacoustic Imaging
PAM	Photoacoustic mammography
PAM3	3rd generation photoacoustic mammoscope
PAT	Photoacoustic Tomography
SIREN	Sinusoidal Representation Networks
SOS	Speed of sound
US	Ultrasound

Symbols and Units

$\lvert\cdot\rvert$	Absolute value or magnitude	
$\lVert\cdot\rVert$	Standard l^2-norm	
$*$	Convolution operator	
C	Space of continuous functions	
$\delta(\cdot)$	Dirac delta function	
$\mathcal{F}\{\cdot\}$	Fourier transform operator	
$\mathcal{F}^{-1}\{\cdot\}$	Inverse Fourier transform operator	
Γ	Grüneisen coefficient	[unitless]
$H(\cdot)$	Heating function	[J/m^3]
$\mathbb{N}$	Natural numbers/Non-negative integers	
$p(\cdot)$	Pressure	[Pa]
$p_0(\cdot)$	Initial Pressure	[Pa]
$\mathbb{R}$	Real numbers	
$\mathbb{R}^+$	Non-negative real numbers	
$\mathcal{R}\{\cdot\}$	Rotation Operator	
$\mathbb{S}^1$	$[0, 2\pi]/0 \sim 2\pi$	
θ	Angle	
$\mathbb{T}^n$	n-dimensional torus ($\mathbb{T}^n = \underbrace{\mathbb{S}^1 \times \cdots \times \mathbb{S}^1}_{n}$)	
$\mathbb{Z}$	Integers	
$\mathbb{Z}_+$	Non-negative integers	

Bibliography

[1] RIVM, *Kans op borstkanker*, Jun. 2023. [Online]. Available: https://www.rivm.nl/bevolkingsonderzoek-borstkanker/over-borstkanker/kans-op-borstkanker.

[2] RIVM, *Bevolkingsonderzoek borstkanker — rivm*, Feb. 2025. [Online]. Available: https://www.rivm.nl/bevolkingsonderzoek-borstkanker.

[3] M. Dantuma, F. Lucka, S. C. Kruitwagen, *et al.*, *Fully three-dimensional sound speed-corrected multi-wavelength photoacoustic breast tomography*, 2023. arXiv: 2308.06754 [physics.med-ph].

[4] S. R. Arridge and O. Scherzer, "Imaging from coupled physics," *Inverse Problems*, vol. 28, p. 080 201, 8 Aug. 2012. DOI: 10.1088/0266-5611/28/8/080201.

[5] P. Beard, "Biomedical photoacoustic imaging," *Interface Focus*, vol. 1, no. 4, pp. 602–631, DOI: 10.1098/rsfs.2011.0028.

[6] K. Wang and M. A. Anastasio, "Photoacoustic and thermoacoustic tomography: Image formation principles," in *Handbook of Mathematical Methods in Imaging*, O. Scherzer, Ed. New York: Springer, 2015, pp. 1081–1116, ISBN: 978-1-4939-0790-8. DOI: 10.1007/978-1-4939-0790-8_50.

[7] B. T. Cox, J. G. Laufer, P. C. Beard, and S. R. Arridge, "Quantitative spectroscopic photoacoustic imaging: a review," *Journal of Biomedical Optics*, vol. 17, no. 6, p. 061 202, 2012. DOI: 10.1117/1.JBO.17.6.061202.

[8] S. R. Arridge, M. M. Betcke, B. T. Cox, F. Lucka, and B. E. Treeby, "On the adjoint operator in photoacoustic tomography," *Inverse Problems*, vol. 32, no. 11, p. 115 012, Oct. 2016. DOI: 10.1088/0266-5611/32/11/115012.

[9] F. Lucka, N. Huynh, M. Betcke, *et al.*, "Enhancing compressed sensing 4d photoacoustic tomography by simultaneous motion estimation," *SIAM Journal on Imaging Sciences*, vol. 11, no. 4, pp. 2224–2253, 2018. DOI: 10.1137/18M1170066.

[10] P. Kuchment and L. Kunyansky, "Mathematics of photoacoustic and thermoacoustic tomography," in *Handbook of Mathematical Methods in Imaging*, O. Scherzer, Ed. New York: Springer, 2015, pp. 1117–1167, ISBN: 978-1-4939-0790-8. DOI: 10.1007/978-1-4939-0790-8_51.

[11] T. T. Khoei, H. O. Slimane, and N. Kaabouch, "Deep learning: Systematic review, models, challenges, and research directions," *Neural Computing and Applications*, vol. 35, pp. 23 103–23 124, 31 2023. DOI: 10.1007/s00521-023-08957-4.

[12] A. Toosi, A. G. Bottino, B. Saboury, E. Siegel, and A. Rahmim, "A brief history of ai: How to prevent another winter (a critical review)," *PET Clinics*, vol. 16, pp. 449–469, 4 2021. DOI: 10.1016/j.cpet.2021.07.001.

[13] K. Azizzadenesheli, N. Kovachki, Z. Li, M. Liu-Schiaffini, J. Kossaifi, and A. Anandkumar, "Neural operators for accelerating scientific simulations and design," *Nature Reviews Physics*, vol. 6, pp. 320–328, 5 2024. DOI: 10.1038/s42254-024-00712-5.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[15] "Approximation theory and fourier analysis," in *Functional Analysis: An Elementary Introduction*, ser. Graduate Studies in Mathematics, vol. 156, American Mathematical Society, 2014, ch. Chapter 9, pp. 147–176, ISBN: 978-0821891711.

[16] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 5 1989. DOI: 10.1016/0893-6080(89)90020-8.

[17] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions*, Jun. 17, 2020. DOI: 10.48550/arXiv.2006.09661.

[18] Y. Sun, J. Liu, M. Xie, B. Wohlberg, and U. S. Kamilov, "Coil: Coordinate-based internal learning for tomographic imaging," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 1400–1412, 2021. DOI: 10.1109/TCI.2021.3125564.

[19] Y. Xie, T. Takikawa, S. Saito, *et al.*, "Neural fields in visual computing and beyond," *Computer Graphics Forum*, vol. 41, no. 2, pp. 641–676, 2022. DOI: 10.1111/cgf.14505.

[20] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *Nerf: Representing scenes as neural radiance fields for view synthesis*, 2020. arXiv: 2003.08934 [cs.CV].

[21] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions*, 2020. arXiv: 2006.09661 [cs.CV].

[22] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019. DOI: 10.1016/j.jcp.2018.10.045.

[23] Z. Li, N. Kovachki, K. Azizzadenesheli, *et al.*, *Fourier neural operator for parametric partial differential equations*, 2021. arXiv: 2010.08895 [cs.LG].

[24] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," *Nature Machine Intelligence*, vol. 3, pp. 218–229, 3 2021. DOI: 10.1038/s42256-021-00302-5.

[25] N. Boullé and A. Townsend, "A mathematical guide to operator learning," in *Numerical Analysis Meets Machine Learning*, ser. Handbook of Numerical Analysis, S. Mishra and A. Townsend, Eds., vol. 25, Elsevier, 2024, ch. 3, pp. 83–125. DOI: 10.1016/bs.hna.2024.05.003.

[26] Y. Chen, M. Staring, O. M. Neve, *et al.*, "Cones: Conditional neural fields with shift modulation for multi-sequence mri translation," *Machine Learning for Biomedical Imaging*, vol. 2, pp. 657–685, Special Issue for Generative Models 2024, ISSN: 2766-905X. DOI: 10.59275/j.melba.2024-d61g.

[27] D. Ha, A. Dai, and Q. V. Le, *Hypernetworks*, 2016. arXiv: 1609.09106 [cs.LG].

[28] L. Lu, X. Meng, S. Cai, *et al.*, "A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data," *Computer Methods in Applied Mechanics and Engineering*, vol. 393, p. 114 778, 2022. DOI: 10.1016/j.cma.2022.114778.

[29] N. Kovachki, Z. Li, B. Liu, *et al.*, "Neural operator: Learning maps between function spaces with applications to pdes," *Journal of Machine Learning Research*, vol. 24, no. 89, pp. 1–97, 2023. [Online]. Available: http://jmlr.org/papers/v24/21-1524.html.

[30] Y. Wang, Y. Chen, Y. Zhao, and S. Liu, "Compressed sensing for biomedical photoacoustic imaging: A review," *Sensors*, vol. 24, no. 9, p. 2670, Jan. 2024. DOI: 10.3390/s24092670.

[31] N. Awasthi, R. Pardasani, S. K. Kalva, M. Pramanik, and P. K. Yalavarthy, *Sinogram super-resolution and denoising convolutional neural network (SRCN) for limited data photoacoustic tomography*, Jan. 17, 2020. arXiv: 2001.06434[eess].

[32] F. Natterer and F. Wübbeling, "The radon transform," in *Mathematical Methods in Image Reconstruction*, Society for Industrial and Applied Mathematics (SIAM), 2001, ISBN: 9780898716221.

[33] B. Rubin, *Introduction to Radon Transforms (With Elements of Fractional Calculus and Harmonic Analysis)*, 1st ed., B. Rubin, Ed. New York: Cambridge University Press, 2015, ISBN: 9780521854597.

[34] E. M. Stein and R. Shakarchi, "The Radon transform and some of its applications," in *Fourier Analysis: An Introduction*, 1st ed., Princeton University Press, 2003, ch. 6.5, pp. 198–207, ISBN: 9780691113845.

[35] J. L. Prince and J. M. Links, "Computed Tomography," in *Medical imaging signals and systems*, 2nd ed., Upper Saddle River, NJ: Pearson Education, 2015, ch. 6, pp. 186–234, ISBN: 9780132145183.

[36] E. M. Stein and R. Shakarchi, "The Radon transform and some of its applications," in *Fourier Analysis: An Introduction*, 1st ed., Princeton University Press, 2003, ch. 6.2, pp. 180–184, ISBN: 9780691113845.

[37] A. Wirgin, *The inverse crime*, 2004. arXiv: math-ph/0401050 [math-ph].

[38] B. E. Treeby and B. T. Cox, "k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields," *Journal of Biomedical Optics*, vol. 15, no. 2, p. 021 314, 2010. DOI: 10.1117/1.3360308.

[39] C. Tian, K. Shen, W. Dong, *et al.*, "Image reconstruction from photoacoustic projections," *Photonics Insights*, vol. 3, no. 3, R06, Sep. 2024, Publisher: SPIE. DOI: 10.3788/PI.2024.R06.

[40] Y. Shen, J. Zhang, D. Jiang, *et al.*, "S-wave accelerates optimization-based photoacoustic image reconstruction in vivo," *Ultrasound in Medicine & Biology*, vol. 50, no. 1, pp. 18–27, Jan. 1, 2024. DOI: 10.1016/j.ultrasmedbio.2023.07.014.

[41] G. Osnabrugge, S. Leedumrongwatthanakun, and I. M. Vellekoop, "A convergent born series for solving the inhomogeneous helmholtz equation in arbitrarily large media," *Journal of Computational Physics*, vol. 322, pp. 113–124, Oct. 1, 2016. DOI: 10.1016/j.jcp.2016.06.034.

[42] Z. Tian, Y. Jing, and A. Han, "An open-source GPU-based acoustic simulator for fast and accurate modeling of acoustic scattering," in *2024 IEEE Ultrasonics, Ferroelectrics, and Frequency Control Joint Symposium (UFFC-JS)*, Sep. 2024, pp. 1–4. DOI: 10.1109/UFFC-JS60046.2024.10793878.

[43] S. Guan, K.-T. Hsu, and P. V. Chitnis, "Fourier neural operator network for fast photoacoustic wave simulations," *Algorithms*, vol. 16, no. 2, 2023. DOI: 10.3390/a16020124.

[44] B. E. A. Saleh and M. C. Teich, *Fundamentals of Photonics*, 3rd edition. Hoboken, NJ: John Wiley & Sons, 2019, ISBN: 9781119506874.

[45] Z. Li, M. Liu-Schiaffini, N. Kovachki, *et al.*, *Learning dissipative dynamics in chaotic systems*, 2022. arXiv: 2106.06898 [cs.LG].

[46] "Cameraman," 1978. [Online]. Available: https://dome.mit.edu/handle/1721.3/195767.

[47] A. Javaherian, F. Lucka, and B. T. Cox, "Refraction-corrected ray-based inversion for three-dimensional ultrasound tomography of the breast," *Inverse Problems*, vol. 36, no. 12, p. 125 010, Dec. 2020. DOI: 10.1088/1361-6420/abc0fc.

[48] A. Petschke and P. J. L. Rivière, "Photoacoustic image reconstruction using the pseudoinverse of the system matrix with the potential for real time imaging," in *Photons Plus Ultrasound: Imaging and Sensing 2012*, vol. 8223, SPIE, Feb. 23, 2012, pp. 676–681. DOI: 10.1117/12.909145.

[49] B. Raonić, R. Molinaro, T. D. Ryck, *et al.*, *Convolutional neural operators for robust and accurate learning of PDEs*, Dec. 1, 2023. DOI: 10.48550/arXiv.2302.01178. arXiv: 2302.01178[cs].

[50] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, Mar. 2021. DOI: 10.1109/MSP.2020.3016905.

[51] J. Kossaifi, N. Kovachki, K. Azizzadenesheli, and A. Anandkumar, *Multi-Grid Tensorized Fourier Neural Operator for High-Resolution PDEs*, Sep. 2023. arXiv: 2310.00120 [cs.LG].

[52] F. Bartolucci, E. d. Bézenac, B. Raonić, R. Molinaro, S. Mishra, and R. Alaifari, *Representation equivalent neural operators: A framework for alias-free operator learning*, Nov. 2, 2023. arXiv: 2305.19913[cs].

[53] A. Hauptmann and B. T. Cox, "Deep learning in photoacoustic tomography: Current approaches and future directions," *Journal of Biomedical Optics*, vol. 25, no. 11, p. 112 903, Oct. 2020, Publisher: SPIE. DOI: 10.1117/1.JBO.25.11.112903.

[54] S. Venturi and T. Casey, "SVD perspectives for augmenting DeepONet flexibility and interpretability," *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115 718, Jan. 1, 2023. DOI: 10.1016/j.cma.2022.115718.

[55] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 218–229, 2021. DOI: 10.1038/s42256-021-00302-5.

[56] J. He, S. Koric, D. Abueidda, A. Najafi, and I. Jasiuk, "Geom-DeepONet: A point-cloud-based deep operator network for field predictions on 3d parameterized geometries," *Computer Methods in Applied Mechanics and Engineering*, vol. 429, p. 117 130, Sep. 1, 2024. DOI: 10.1016/j.cma.2024.117130.

[57] A. Dax, "From eigenvalues to singular values: A review," *Advances in Pure Mathematics*, vol. 3, no. 9, pp. 8–24, Dec. 4, 2013. DOI: 10.4236/apm.2013.39A2002.

[58] X. Lyu, L. Cheng, and S. Zhang, "The reta benchmark for retinal vascular tree analysis," *Scientific Data*, vol. 9, p. 397, 1 2022. DOI: 10.1038/s41597-022-01507-y.

[59] B. Treeby, B. Cox, and J. Jaros, *K-wave a matlab toolbox for the time domain simulation of acoustic wave fields user manualk-wave a matlab toolbox for the time domain simulation of acoustic wave fields user manual, version 1.1*, Aug. 2016.