BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

ACCELERATION OF NEUROSTIMULATION USING ARTIFICIAL INTELLIGENCE METHODS

AKCELERACE NEUROSTIMULACE POMOCÍ METOD UMĚLÉ INTELIGENCE

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR Bc. MARTIN GAŇO
AUTOR PRÁCE

SUPERVISOR doc. Ing. JIŘÍ JAROŠ, Ph.D.
VEDOUCÍ PRÁCE

BRNO 2022

Ústav počítačových systémů (UPSY) Akademický rok 2021/2022

Zadání diplomové práce

23927

Student:	**Gaňo Martin, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Počítačové vidění
Název:	**Akcelerace neurostimulace pomocí metod umělé inteligence** **Acceleration of Neurostimulation Using Artificial Intelligence Methods**
Kategorie:	Umělá inteligence

Zadání:

1. Seznamte se s problematikou šíření ultrazvuku v mozku popsanou Helmholtzovou rovnicí ve 3D prostoru.
2. Prostudujte metody umělé inteligence vhodné pro aproximaci Helmholtzovy rovnice.
3. Zvolte vhodnou metodu aproximace Helmholtzovy rovnice s ohledem na rychlost výpočtu.
4. Zvolenou metodu implementujte.
5. Iterativně vyhodnocujte a vylepšujte vyvíjené řešení na zadaných datových sadách.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Jaroš Jiří, doc. Ing., Ph.D.**
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1. listopadu 2021
Datum odevzdání:	18. května 2022
Datum schválení:	29. října 2021

Abstract

Treatment using transcranial ultrasound is a rapidly arising domain of medicine. This method brings options for non-invasive brain therapies, including ablation, neuromodulation, or potentially opening the blood-brain barrier for the following treatment. The health officer needs to constantly receive feedback on the ultrasound wavefield in the human skull in real-time to accomplish the cure using these techniques. The traditional methods for simulating monochromous ultrasound waves are computationally too expensive. That is why their usage would be infeasible for these purposes, and it brings the need for alternative methods. This work proposed and implemented a method to solve the Helmholtz equation in 3D space using a neural network achieving a faster convergence rate. The neural network design uses lightweight architecture based on *UNet*. The main interest of this work is neuromodulation because, in this application, it is possible to ignore several variables and phenomena that would not be negligible in other use cases. Omitting them from the calculations increased the chances of accomplishing computations in a reasonable time. The method is fully unsupervised and uses exclusively artificially generated spherical harmonics and physics-based loss for training, with no required ground truth labels. Results showed a faster calculation with acceptable error than other traditional methods.

Abstrakt

Léčba pomocí transkraniálního ultrazvuku je rychle se rozvíjející doménou medicíny. Tato metoda přináší možnosti neinvazivní mozkové terapie, včetně ablace, neuromodulace nebo potenciálního otevření hematoencefalické bariéry pro následující léčbu. Zdravotník potřebuje neustále dostávat zpětnou vazbu o ultrazvukovém vlnovém poli v lidské lebce v reálném čase, aby mohl pomocí těchto technik provést léčbu. Tradiční metody pro simulaci monochromních ultrazvukových vln jsou výpočetně příliš drahé. Jejich použití by proto bylo pro tyto účely neproveditelné a přináší to potřebu alternativních metod.Tato práce navrhla a implementovala metodu řešení Helmholtzovy rovnice ve 3D prostoru pomocí neuronové sítě dosahující vyšší rychlosti konvergence. Návrh neuronové sítě využívá odlehčenou architekturu založenou na UNet. Hlavním předmětem zájmu této práce je neuromodulace, protože v této aplikaci je možné ignorovat několik proměnných a jevů, které by v jiných případech nebyly zanedbatelné. Jejich vynecháním z výpočtů se zvýšila šance na provedení výpočtů v rozumném čase. Tato metoda je plně bez dozoru a používá výhradně uměle generované sférických harmonik a fyzikální ztráty pro trénink, bez nutnosti anotovaných dat. Výsledky ukázaly rychlejší výpočet s přijatelnou chybou než jiné tradiční metody.

Keywords

Helmholtz equation, learned optimizer, unsupervised learning, physics-based loss function, transcranial ultrasound.

Klíčová slova

Helmholtzova rovnica, naučený optimalizátor, učenie bez učiteľa, fyzikálna chybová funkcia, transkraniálny ultrazvuk.

Reference

GAŇO, Martin. *Acceleration of Neurostimulation Using Artificial Intelligence Methods*. Brno, 2022. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. JIŘÍ JAROŠ, Ph.D.

Rozšířený abstrakt

S výzkumem v oblasti neinvazivní léčby v oblasti mozku vzniká nový obor založený na dávkování ultrazvukového záření. Podstata léčby a dávkování závisí na frekvenci, intenzitě a dalších vlastnostech ultrazvuku. V současné době nejznámější a nejvíce sledované metody jsou ablace, neurostimulace a otevírání hematoencefalické bariéry pomocí ultrazvuku. Ablace je proces, při kterém cíleně pálíme části maligní tkáně v lidském mozku, které mohlo vzniknout například rakovinným bujením. Tato metoda vyžaduje vysoké intenzity a proto žádá značnou přesnost. Neurostimulace se využívá ke stimulaci zvolených regionů mozku za účelem prevence, resp. léčby neurodegenerativních onemocnění jako Alzheimerova, či Parkinosonova choroba. Dočasné otevření hematoencefalitické bariéry se provádí za účelem přenosu léčiv za oblast této bariéry (do mozku).

Všechny zmíněné metody mají společný požadavek – pro jejich provádění je potřeba znát relativně přesný model vlnového pole po zahájení působení ultrazvuku v lidském mozku. Hlavní výzvou pro určení vlnového pole je přítomnost lebeční kosti, která zkresluje, nebo v některých případech dokonce zcela zničí ohnisko ultrazvukových vln. Tento jev je přítomen i v případě nižších frekvencí. Rozdílné tvary, složení či tloušťka lebky v populaci zcela znemožnují předpočítat vlnové pole obecně a vyžaduje personalizovaný přístup ke každému pacientovi.

Zdravotníci tedy musí znát stav vlnového pole, především aktuální pozici ohniska ultrazvuku v reálném čase. Tradiční metody řešící tento problém mají vysokou výpočetní náročnost a pomalou rychlost konvergence. Výpočet vlnového pole v modelu lidské lebky může trvat desítky minut až několik hodin, což činí metodu prakticky nevykonatelnou. V současnosti je snaha najít metodu, která je vykonatelná na úkor přesnosti, nicméně musí poskytnout dostatek informace pro zdravotníka. Cílem naší práce je prozkoumat metody provádějící aproximaci vlnového pole v ustáleném stavu s monochromatickým zdrojem vlnění a nabídnout vhodnou alternativu pro stávající řešení.

Hlavním tématem našeho zájmu je neurostimulace, protože nám umožňuje zanedbat některé proměnné a jevy, což zjednodušuje výslednou rovnici a zvyšuje naše šance na poskutnutí podpory při dané terapii. Úřad pro kontrolu potravin a léčiv (FDA) dokonce schválila neuromodulaci pomocí ultrazvuku pro léčbu Parkinsonovy choroby. Na základě dostupné literatury jsme se rozhodli zanedbat v našem řešení různé jevy, jako například třecí efekt nebo nelineární efekty, a naším cílem je řešit homogenní Helmholtzovu rovnici ve 3D prostoru.

V rámci této práce byla navržena neuronová síť založená na architektuře UNet jako alternativa pro iterativní numerické solvery pro parciální diferenciální rovnice. Hlavním úkolem našeho řešení je iterativně snižovat chybu a přibližovat se ke správnému řešení podobně jako solvery pro parciální diferenciální rovnice. Design našeho řešení vychází ze sítě UNet a jde o relativně odlehčenou architekturu za účelem snížení času inference a trénování modelu.

Navržená metoda je plně naučená bez učitele, tedy bez znalosti referenčního řešení, díky využití fyzikální chybové funkce. Tato chybová funkce je reprezentována inverzním problémem vůči Helmholtzově rovnici. Díky této chybové funkci očekáváme dobrou generalizaci větších domén a zřejmě se vyhneme nutnosti generování velkého množství referenčních dat, což by představovalo výpočetně drahou úlohu.

Acceleration of Neurostimulation Using Artificial Intelligence Methods

Declaration

I hereby declare that this master's thesis was prepared as an original work by the author under the supervision of Mr Jiří Jaroš. The supplementary information was provided by Mr Antonio Stanziola. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

....
Martin Gaňo
May 24, 2022

Acknowledgements

I want to thank my supervisor Mr Jiří Jaroš along with external collaborator Mr Antonio Stanziola for their encouragement and continuous support throughout this project. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90140).

Contents

Chapter 1

Introduction

1.1 Problem description

With attempts to perform operations or treatments on the human brain comes the need for an effective non-invasive method because of its highly vulnerable character. One of the rapidly emerging methods is transcranial ultrasound therapy [5]. The essence of the treatment depends on the addition of exogenous microbubbles, frequency, intensity and other properties of the ultrasound. The most frequently researched types of treatment are ablation [44], neurostimulation [41, 39] and opening of the blood-brain barrier [1]. The ablation is a process of destroying the small, strictly selected part of the tissue, usually with some malignant character. Neuromodulation is applied to stimulate regions of the brain in order to prevent or treat, e.g. degenerative brain issues. Opening the blood-brain barrier enables the delivery of medications to the brain.

The methods have one common aspect: the requirement of a relatively precise wavefield model in the skull. The primary issue is the skull bone which causes distortions and shifting of the focus position [23], even destroying it [64], even for methods using low frequencies [29]. The population's skull shape and morphology differences prevent precomputing the general wavefield model [15].

Thus the health officers need constant real-time feedback on the state of the wavefield in the brain and mainly the focus of the wavefield [28]. However, commonly used methods require minutes to hours to calculate the wave propagation in the human brain model, making the treatment often infeasible in general practice [47, 49]. Eventually, the feasibly usable method needs to reasonably sacrifice accuracy in order to speed up but provide enough information about the wavefield [18].

This work explores existing methods for approximation wavefield in a steady-state with the monochromous radiation source and proposes a suitable solution for the problem. The topic of our interest is neuromodulation because it enables us to neglect some variables and simplify the target equation and increase our chances of solving the problem in an adequate time. Food and Drug Administration (FDA)[1] even approved neuromodulation for treating Parkinson's disease [60]. The solution in this work excludes effects like the shear wave effects or nonlinear effects, and our goal is to solve the homogeneous Helmholtz equation in 3D space [54].

This work proposes *UNet* based architecture as an alternative to iterative numerical partial differential equation solvers. The objective of the network is to iteratively decrease

[1] https://www.fda.gov/

the error of the wavefield and improve its accuracy, similarly to numerical PDE solvers. The design comes from *UNet* and relatively lightweight architecture in order to push down the inference and training time and memory. The method is fully unsupervised where *physics-based loss* [67] is represented by the inverse problem to our target equation [32]. We expect to achieve improved generalizability [53] to higher scale problems. Physics-based also enables us to avoid generating a massive amount of labelled data that would need to be generated by computationally expensive simulations, potentially in higher domains.

1.2 Key contributions

In this thesis, we considered various approaches to accelerating wave propagation computation and eventually developed a novel method for 3D space. The selected method was based on 2D Helmnet proposed by Stanziola et al. [63]. We implemented the Helmnet-based solution for 3D and experimentally demonstrated its speed and accuracy compared to other methods. The implemented framework provides the functionality of the trained model that predicts a 3D wavefield in the steady-state given the speed of sound distribution and source distribution. The functionality is shown in Figure 1.1.

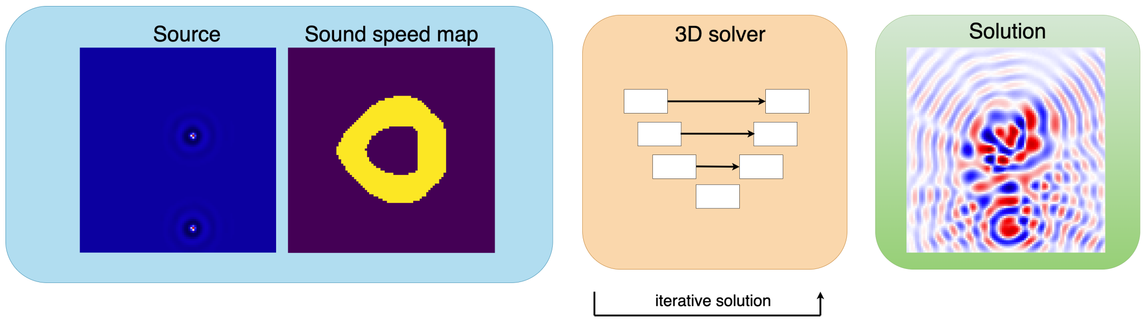

Figure 1.1: Our 3D neural network-based iterative solver takes as input source and sound speed distributions, and after performing a certain number of iterations outputs the target wavefield.

1.3 Structure of thesis

- Chapter 1 introduced the problem and motivation for our aim to replace traditional methods in some applications. The chapter described the most frequently studied ultrasound-based treatments for the brain. Then we presented a positive prospect of the method for future use not only restricted to wave equation.
- Chapter 2 explained the mathematical aspect of the problem. This chapter set the governing equation for our main problem and showed traditional ways of solving the wave equation. Except for traditional methods, we also showed a few alternative ways of solving the problem. Therefore we introduced physics-based machine learning and explained the motivation why we decided to apply it to solve the Helmholtz equation.
- Chapter 3 explained in detail the method proposed in this work. The chapter's primary goal is to describe the theoretical background for the Helmnet in 3D, the essential parts of the solution (e.g. replay buffer for TBPTT), motivation for using

it, and planned ways to evaluate the result. The part of the evaluation is also an artificial dataset described in this chapter.

- Chapter 4 focused on the technical aspects of our work connected to programming language, selected framework and computational resources. The most important part of this chapter is the recommendations for future work that should be considered. The recommendations included exploring different parameters, neural network frameworks etc.

- Chapter 5 provided a complete evaluation of the method provided by this work. The three methods explored in this chapter are 3D Helmnet, GMRES and k-Wave simulation.

1.4 Summary

The first chapter introduced the motivation for bringing the new method to the computational ultrasound field of study. In this chapter, we also described this thesis's key contributions and structure.

Chapter 2

Wave propagation

Generally, computing the wave propagation in solid volumes is a remarkably complex task. The exact computation of wavefield in the human skull involves acoustic absorption, nonlinear effects or shear motion. Ultrasound with parameters for our problem enables us to omit absorption from the equation [69]. Moreover, the critical nonlinear effects are present only for higher intensities [58]. The shear motion effect does not concern significant issues since the radiation direction is perpendicular to the skull barrier. Its influence is negligible when the radiation is near normal incidence [56, 71, 13]. Thus these two variables are also neglected from our focus.

2.1 Wave equation

The objective of the work is to predict wavefield, which brings the need for a mathematical representation of wave propagation. The general form of the wave equation in the n-dimensional space:

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c(x)^2 \nabla^2 u(x,t) \tag{2.1}$$

where $u : \mathbb{R}^n \to \mathbb{C}$ represents the n-dimensional wavefield in given time, $x \in \mathbb{R}^n$ is the spatial coordinate and $t \in \mathbb{R}^+$ is a time variable. $c : \mathbb{R}^n \to \mathbb{R}^+$ is the speed of sound distribution and ∇^2 is the spatial *Laplacian operator*.

Nonetheless, the time variable is negligible for many medicinal applications of transcranial ultrasound therapy, including the method studied in this work. The time used for the treatment exceeds the time needed to achieve the steady-state. Thus, the wavefield does not need to be computed every time unit [39]. The time-independent character and the monochromatic source simplify the problem. The Helmholtz equation describes the simplified homogeneous time-independent wave propagation model subject to the Sommerfeld radiation condition.

2.2 Helmholtz equation

The Helmholtz equation is the eigenvalue problem for the Laplace operator corresponding to linear partial differential equations. It is based on Equation 2.1 by separation of variables and dropping time dependence:

$$\left[\nabla^2 + \left(\frac{\omega}{c(r)}\right)^2\right] u(r) = \rho(r) \tag{2.2}$$

where $u : \mathbb{R}^n \to \mathbb{C}$ is n-dimensional wavefield, r is n-dimensional point, $c : \mathbb{R}^n \to \mathbb{R}^+$ is the speed of sound distribution corresponding to the skull model, ω is the angular frequency, $\rho : \mathbb{R}^n \to \mathbb{C}$ is the source distribution. The source distribution is a n-dimensional tensor with source points equal to the source amplitude. ∇^2 is the spatial Laplace operator. In this work, we assume to have just two sound speed values in distribution c. One value is the background, and the other denotes sound speed in skull bone. The parameters are illustrated in Figure 2.1.

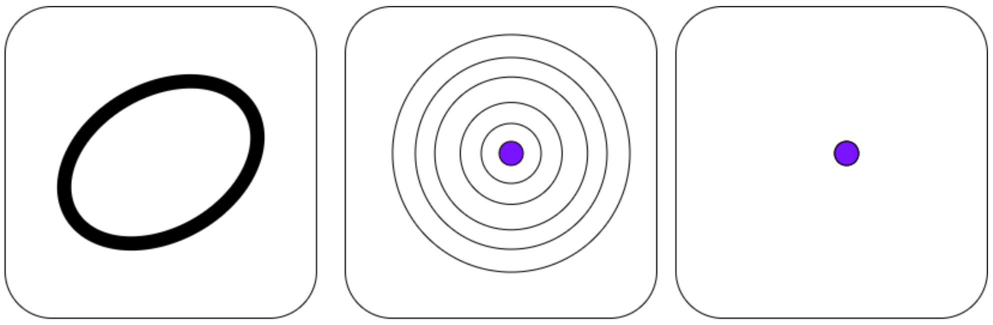

Figure 2.1: Figure shows the visualization of the three most important variables of the Helmholtz equation. The first picture shows the speed of sound (SOS) distribution c of the skull. The sound speed is derived and approximated using brain magnetic resonance imaging (MRI). The second picture shows the solution's expected result: the wavefield after applying ultrasound therapy in the steady-state. The third picture illustrates source distribution, one point (pixel) in the source tensor, with the value of source amplitude, for every source.

The Equation 2.2 is solved conditioned to Sommerfeld radiation condition:

$$\mathit{lim}_{|r|\to\infty} |r|^{\frac{n-1}{2}} \left(\frac{\partial}{\partial |r|} - i\frac{\omega}{c_0}\right) u(r) = 0 \tag{2.3}$$

where ω is the angular frequency of the source, $n \in \mathbb{R}^+$ is the number of spatial dimensions, c_0 is the sound speed of the background, r is the n-dimensional point and u is the wavefield. The problem is solved within the region $\Omega \subset \mathbb{R}^n$ as showed in Figure 2.2.

Equations were described by Stanziola et al. [63].

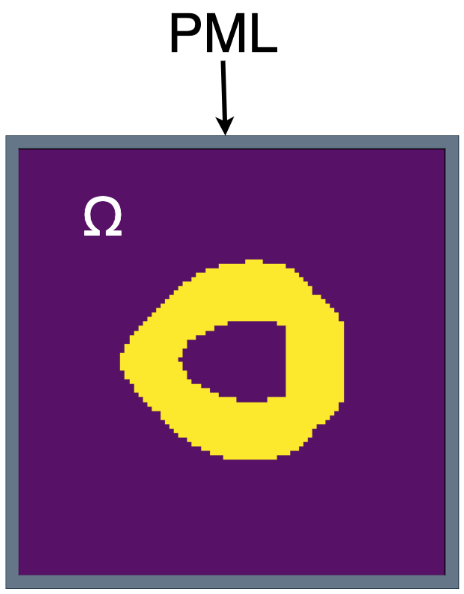

Figure 2.2: The perfectly matched layer is used to surround the domain of interest Ω. Figure shows one of the 2D training samples abstractly surrounded by the PML.

2.3 Numerical solutions

2.3.1 *Perfectly matched layer* (PML)

One way to remove the Sommerfeld radiation condition from the equation and get the solution within the domain is by applying the perfectly matched layer (PML). PML is the artificial absorption layer for wave equations for eliminating waves propagating from the inside used to truncate computational domains in numerical methods to simulate problems with open boundaries [65, 34]. In practice, it enables numerical methods to omit Equation 2.3 from our calculations. PML wraps up the domain as in Figure 2.2, and the system behaves as the waves are propagating into the infinite space [8].

Our implementation of the 3D perfectly matched layer is based on Bermudez et al. [9]. The domain wrapped up by the PML is a cube with sizes $2L_x, 2L_y, 2L_z$ such that:

$$L_x = L_y = L_z$$

and

$$2L_x * 2L_y * 2L_z = \Omega$$

where $\Omega \in \mathbb{R}$ is the cubical domain of interest. Then the domain wrapped up by PML is extended by PML so

$$\Omega \oplus PML = 2(L_x + \Delta L) * 2(L_y + \Delta L) * 2(L_z + \Delta L) \tag{2.4}$$

where ΔL is the thickness of the PML and $\oplus$ is operator of addition the PML to the domain of interest.

For bringing the absorption the derivative operators are transformed within the extended part ΔL of the domain of interest Ω:

$$\frac{\partial}{\partial \eta} \to \frac{1}{\gamma_\eta}\frac{\partial}{\partial \eta} \tag{2.5}$$

where $\eta = x$, y and z for 3D space and

$$\gamma_\eta = \begin{cases} 1, & \text{if } |\eta| < L_\eta \\ 1 + \frac{j}{\omega}\sigma(\eta), & \text{if } L_\eta \leq |\eta| < L_\eta + \Delta L \end{cases} \tag{2.6}$$

where ω is the angular frequency as described in Equation 2.2, area, such that $|\eta| < L_\eta$ is equal to the domain Ω without PML extension, $L_\eta \leq |\eta| < L_\eta + \Delta L$ denotes PML and σ is the absorption profile that grows quadratically within PML as defined by Bermudez et al. [9]:

$$\sigma(\eta) = \sigma_{max}\left(1 - \frac{\eta}{\Delta L}\right)^2, \tag{2.7}$$

where σ_{max} is the maximum absorption coefficient, ΔL is the thickness of the PML as in Equation 2.4.

Then the Laplace operator ∇^2 from Equation 2.2 must contain PML for 3D that has to be included in our equation and consequent implementation:

$$\hat{\nabla}^2 = \frac{1}{\gamma_\eta}\frac{\partial}{\partial\eta} = \frac{1}{\gamma_x}\frac{\partial}{\partial x} + \frac{1}{\gamma_y}\frac{\partial}{\partial y} + \frac{1}{\gamma_z}\frac{\partial}{\partial z} \tag{2.8}$$

that gives us modified main equation:

$$\left[\hat{\nabla}^2 + \left(\frac{\omega}{c(r)}\right)^2\right] u(r) = \rho(r) \tag{2.9}$$

The program realization of the thesis uses thickness of the PML layer $\Delta L = 8$ and maximal absorption coefficient $\sigma_{max} = 2$.

2.4 Partial derivative equations solvers

There are plenty of methods for solving partial derivative equations, including the Helmholtz equation. The methods vary by accuracy, speed and other related parameters. This work explored methods of solving PDEs based on differential operator, solution function, and forcing function (the same form as the Helmholtz equation). The equations with this form can be rewritten into a more general form:

$$A(c)u = \rho, \tag{2.10}$$

where $A(c)$ is a linear forward operator depending on sound speed distribution c, u is the solution function and ρ is the (known) forcing function. This equation is solvable by many techniques including boundary element methods [25], finite-element methods [9] or finite difference methods [70]. For our needs, it is not possible to inverse the forward operator (isolate solution) because of the size of the problem. That is why we need to employ iterative schemes. The solution of the partial derivative equation can be viewed as an optimization task for which we need to set the proper loss function and apply the optimization (minimization) algorithm.

To find a solution using an optimization algorithm, we must choose a loss function. The most meaningful candidate for the loss function is the squared norm of the residual e_k, equivalent to the mean square error. The loss function for this purpose was introduced by Raissi et al. [51] and is described by the following equation:

$$L_k(u_k, c, \rho) = ||e_k||^2 = \int_\Omega |e_k|^2 dr, \tag{2.11}$$

$$\text{s.t. } e_k = A(c)u_k - \rho \tag{2.12}$$

Other alternative approaches might be RMSE or MAE [14], using the physics-constrained loss [75] or Dirichlet energy [74].

With the selected loss (residual) function, it is now possible to apply a numeric solution. A suitable method for this purpose is *Generalized minimal residual method* (GMRES). GMRES is an iterative numerical method which approximates the solution by the vector in a Krylov subspace with minimal residual [59]. The Arnoldi iteration method is used for the construction of an l_2-orthogonal basis of the Krylov subspaces. [31].

Generally, solving the Helmholtz equation using iterative methods is computationally expensive because of a known slow convergence of methods using Krylov subspace [21]. The already mentioned motivation for this work is that the Krylov subspace methods are ineffective, especially when the number of waves in the Helmholtz operator becomes large. Also, standard algebraic preconditioners do not remedy the situation [21]. The issue is that every iteration update is local. However, the monochromatic wave problem is not local, and we know that, e.g. vital radiation source on one side of the domain of interest Ω influences the state of the wavefield in the point on the other side of the domain Ω.

The method used for generating the reference solution in this work is a part of *k-Wave* [68], also a conventional solver with the application of k-space pseudo spectral [11].

2.4.1 The generalized minimal residual (GMRES) algorithm

Since the GMRES is used as a benchmark for our solution (however just in small domain), we described it in more detail. The algorithm was proposed in 1986 by Saad and Schultz [59]. The method intended to improve methods like the ORDHODIR method from 1980 [73], Axellson's method from 1980 [3] or the generalized conjugate residual method [20].

As the name suggests, the method computes the solution with the minimal residual. A general algorithm for GMRES described by the authors is described by the Algorithm 1 [59].

2.4.2 Neural networks for PDEs

Neural networks are a promising alternative to classical methods for solving PDEs. This method brings a new point of view to the topic and remedies specific known issues. *Convolutional neural networks* (CNN) are capable of processing changes globally and speeding up the solution [22]. Their usage for solving physical equations is becoming more common [12]. Except for that, this field of study is still quickly developing, and we can expect new techniques and improvements, which will eventually improve the whole solution, including neural networks. Advantages are massive developers community improving the software or specialized hardware as *Tensor Processing Unit* (TPU) [35]. Moreover, using neural networks, it is possible to overcome some unpleasant phenomena related to the PDEs, like *curse of dimensionality* described by Bellman [7, 6, 48].

Their usage offers very promising approximation capabilities for nonlinear functions [10]. *Physics-informed neural networks* (PINN) [43] offers ability of unsupervised learning optimization without annotated data. Blechschmidt et al. [10] described a general approach for approximating the solution $u : (\mathcal{T}, \mathcal{D}) \to \mathbb{R}$ by PINN:

$$\partial_t u(t, x) + \mathcal{N}[u](t, x) = f, \tag{2.13}$$

Algorithm 1 GMRES

```
initialize $x_0$, $m$, $\epsilon$ ▷ $x_0$ is the initial value and $m$ number of iterations, $\epsilon$ is maximal error
$r_0 = f - Ax_0$                                    ▷ Algorithm starts
$v_1 = r_0/||r_0||$
for j < m do                                        ▷ Iterative solution
    $h_{i,j} = (A_{v_j}, v_i),\ i = 1, 2, ..., j$
    $\hat{v}_{j+i} = A_{v_j} - \sum_{i=1}^{j} h_{i,j} v_i$
    $h_{j+1,j} = ||\hat{v}_{j+1}||$
    $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$
end for
$x_m = x_0 + V_m y_m$, where $y_m$ minimizes $||\beta e_1 - \bar{H}||$, $y \in \mathbb{R}^m$
$r_m = f - Ax_m$
if $r_m > \epsilon$ then
    $x_0 = x_m$
    $v_1 = r_m/||r_m||$
    goto start
```

$$u(0, x) = u_0(x), \tag{2.14}$$

where $\mathcal{N}$, u is a solution and f is a forcing term, $x \in \Omega$. As discussed earlier, we neglect the time variable t from our computations. Then the trained neural network should approximate the solution function $u_\theta(x) \approx u(x)$ using optimized parameters θ. The physical information differs this method from other learning-based techniques trying to optimize the solution exclusively using data by the fitting network to labelled data $\{x_i, u(x_i)\}_{i=1}^{N}$. The PINN approach was already successfully applied in a wide range of applications, including fluid dynamics [52], inverse problems [45], or stochastic differential equations [72], which suggests us it may bring success also for the Helmholtz equation.

Stanziola et al. [63] proposed method employing architecture *UNet* [57] originally used for biomedical image segmentation. Architecture UNet is shaped as a letter „U“. It consists of a contracting path (the left side of Figure 2.3) and expanding path (the right side of Figure 2.3).

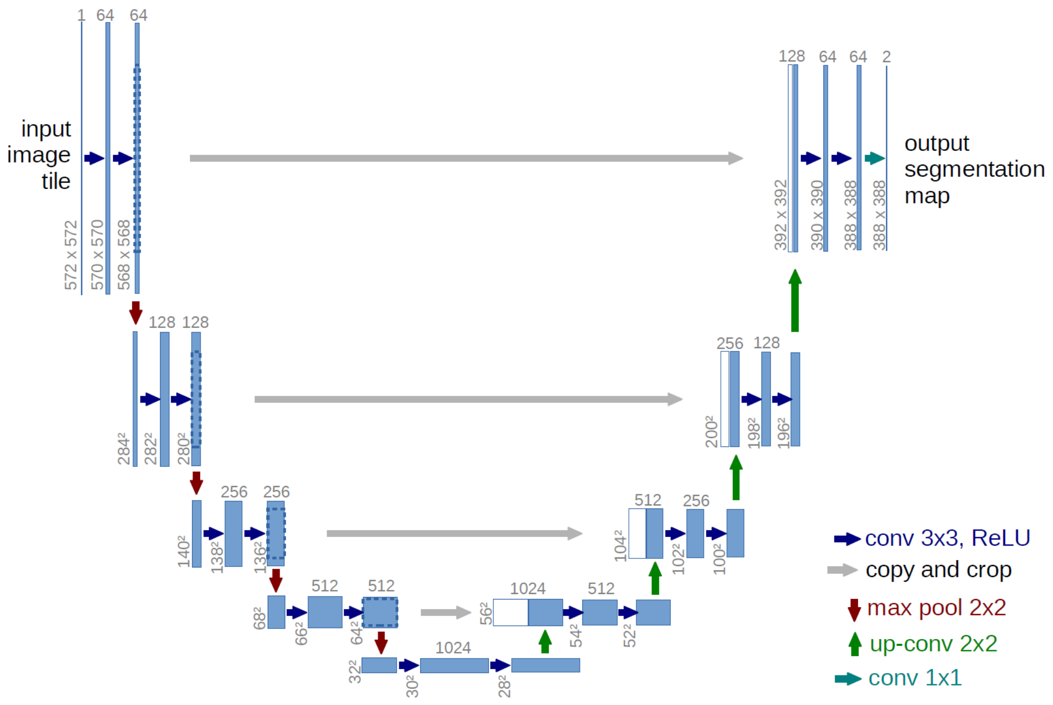

Figure 2.3: The original UNet architecture. Figure is adopted by Ronneberger et al. [57].

The work by Stanziola et al. [63] used UNet as an iterative 2D solver for boosting the current solution to the solution with a minor error. The application of the learned iterative solver for the Helmholtz equation was initially proposed by Rizzuti et al. [55] with the usage of UNet, using the finite-differences for the discretization. The UNet by Rizzuti et al. was trained using ground truths, but it is suggested that training in true fields might not be necessary for successful results. Stanziola et al. [63] also suggest using some Krylov iterations to prevent the solution from diverging as a regularization technique.

2.5 Summary

Wave propagation is a too complex topic to be described in one chapter of the thesis. Despite that, we tried to remind the most important aspects of the wave propagation in the application of transcranial ultrasound in section 2.1. That includes the general form of the wave equation that is generally time-dependent. The time-dependent equation was not solved in this work because, in general practice, the wavefield achieves steady-state within a short time, and the simulation wavefield before that happens is not essential.

We also explained the variables and phenomena we intentionally omitted from our focus. The main problem of this thesis is not the general wave equation, but the Helmholtz equation encapsulated into the perfectly matched layer to satisfy the Sommerfeld radiation condition. Section 2.2 shortly outlines the Helmholtz equation and visually demonstrates its most important variables of it.

The subsection 2.3.1 also reports about the perfectly matched layer and its essence for this work. We also shortly presented the mathematically-implementational aspect of the PML and prefigured the procedure of using PML for 3D adopted by Bermudez et al. [9]. A more detailed description of the implementation of the PML is included in Chapter 4. The thickness and absorption coefficient used in this work is adopted by Stanziola et al. [63] since the size of the problem is more or less the same. However, in a lower dimension than our work.

This chapter enumerates a few most common (numerical) techniques for solving partial derivative equations, specifically wave equations. These methods might serve as a benchmark for our newly emerging method and help us evaluate our results. The highlighted method of those is GMRES, which is a dedicated section 2.4.1. Since GMRES plays an essential role in our evaluation and is also applied in the 2D Helmnet [63] as a benchmark, we described it as slightly more profound than other methods, including the pseudocode of the algorithm.

The section 2.4 defines the physical loss function of our problem in general. This loss function does not need the reference value of the data, only the input data. The function was tested using the primitive optimizing algorithm. It showed that it yields visually meaningful data when optimized.

Last but not least, we briefly pointed out alternative approaches for solving physical problems (specifically PDEs) using machine learning. The approach is so-called *physics informed neural networks* (PINN). The section 2.4.2 mentioned a few of many methods for solving partial derivative equations using neural networks. The methods and possible use cases are a large amount. Thus we mentioned only a few most interesting for this work. The base for our solution is the architecture UNet for the image segmentation and is widely used in combination with a physics-based loss function. The section also described the motivation for solving this approach, mentioned the direct and indirect implications of using PINN, and eventually explained that our method, given its loss, is based on these approaches. A more detailed description of the UNet-based architecture used in our work is in the following Chapter 3.

Chapter 3

UNet for 3D Helmholtz equation

3.1 Helmnet

As we mentioned earlier, Stanziola et al. proposed *Helmnet*[1] [63] - UNet based neural network architecture for approximating iterative solver for 2D Helmholtz equation. The proposed method achieved impressive results compared to classical PDEs iterative solvers in terms of speed, accuracy, and trade-off. The accuracy of the Helmnet was computed using two loss functions - relative *Chebyshev distance* norm l_∞ and average RMSE error norm:

$$l_\infty = \frac{||\hat{u} - u||_\infty}{||u||_\infty} \tag{3.1}$$

$$RMSE = \sqrt{\frac{||\hat{u} - u||_2^2}{\mathrm{N}}} \tag{3.2}$$

where $\hat{u}$ is the solution provided by the trained iterative solver, u is the reference solution and N is the total number of pixels in the wavefield. The predicted and reference wavefields were normalized to a maximal amplitude of 1. The benchmark for the Helmnet was the GMRES method. As a result after 1000 iterations the l_∞ error achieved 0.36% and mean RMSE 4.6×10^{-4}. The $l_\infty = 1\%$ was even achieved after 250 iterations, and this error might be sufficient for medical usage. GMRES achieved an order of magnitude worse results after 1000 iterations.

The reference ground truth of the time-independent wave equation was generated using k-Wave. The time-domain solver *kspaceFirstOrder2DG* computed the solution in the steady-state. For the evaluation, the region of the perfectly matched layer was excluded because the PML is defined differently for k-Wave than for Helmnet, and this region is not interesting for practical applications.

Our reproduction of the Helmnet showed excellent results in terms of generalization to higher domains and other source positions, source amplitudes and frequencies. Hyperparameters required for reproducing Helmnet results the same way as in this work are shown in Table3.1. Except for the error function, it is also possible to track the error of the solver using the residual function (without knowledge of the reference solution) also defined in Equation 2.12 and used by Helmnet as a loss function. This feature enables us to find the optimal value of the residual (order of magnitude 10^{-4} with given data) and regulate the number of iterations accordingly.

[1] `https://github.com/ucl-bug/helmnet`

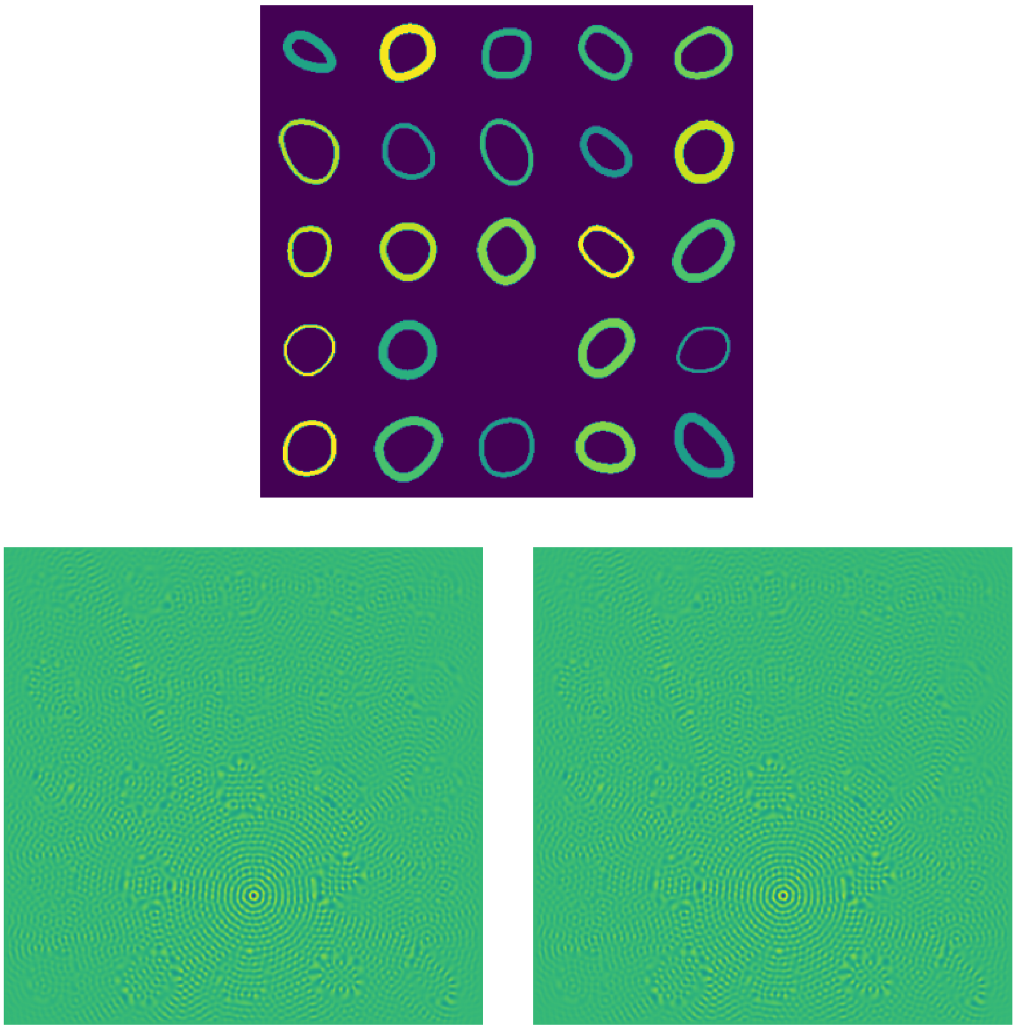

Figure 3.1: Figure shows the sound speed distribution of the domain with size 480x480 with 24 merged samples and the radiation source located in the „empty“ space of the grid. The left wavefield is the reference wavefield generated by k-Wave simulation, and 1000 iterations of the learned optimizer predict the right wavefield. The error after 1000 iterations is approximately 0.5%. This sample's inference (1000 iterations) took 9.29s using 1 GPU on the Karolina supercomputer. The colour of the sample denotes the speed of the sound in the given material.

Table 3.1: This table shows hyperparameters used for training 2D solver. The hyperparameters was used for our reproduction of Helmnet.

epochs	1000
buffer size	600
batch size	32
learning rate	0.0001
weight decay	0.000001
depth	6
state channels	2
interlayer channels	8
activation function	PReLU
TBPTT unrolling steps	10
gradient clipping	1

Table 3.2: This table shows the time needed for training the 2D solver on 9000 samples sized 96x96 using 8 GPUs NVIDIA A100 on supercomputer Karolina. Hyperparameters for training can be found in Table 3.1. Other values are the time needed for computation of wavefield in steady-state for one sample of the size 96x96 using 1 GPU NVIDIA A100 on Karolina using learned optimizer (neural network) and GMRES method. Both methods use 1000 iterations per sample (we even showed that GMRES have significantly higher error after the 1000th iteration).

Training	$\sim$ 9 hours
Inference (neural network)	$\sim$ 4 seconds
Computation (GMRES)	$\sim$ 8 seconds

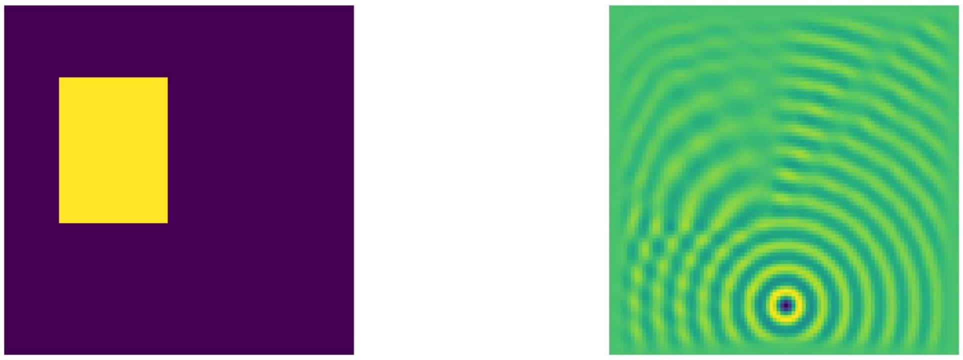

Figure 3.2: Figure shows the sound speed distribution of the domain with a size of 96x96. However, instead of an idealized skull model generated as a sum of several circular harmonics, this sample contains a filled rectangle and its predicted wavefield using a neural network. The ℓ_∞ error after 1000 iterations is below 0.45%. This sample's inference (1000 iterations) took 4.54s using 1 GPU on the Karolina supercomputer.

3.2 2D training dataset

The significant advantage of the PINN is their ability to train without annotated data. As we already mentioned, generating ground truths for the Helmholtz equation is computationally intensive (even for two dimensions). Generation of the large dataset for supervised training would consume a massive amount of time and computational resources. The Helmnet took advantage of PINN and used a physics-based loss function instead of labelled data.

Although there is no need for annotated data, the method still needs raw inputs. The required input data are as illustrated in Figure 1.1, source distribution and sound speed distribution. Creating source distribution is a trivial task (placing source amplitude value in the corresponding pixel position for every source), and the generation of the artificial skull model is a little more complicated. The speed sounds dataset needs to resemble a human skull scan and cover its differences in population. For this purpose, the authors summed up a few 2D circular harmonics with random amplitude and phase and created around them barriers with uniformly random thickness between 2 and 10 pixels. The value filling the barrier is homogeneous within the artificial skull bone and denotes sound propagation speed. The value, in reality, is usually 1.5 to 2 times faster than in the background (soft tissues) [63]. This is preserved in the dataset, with a background sound speed value of 1 and skull bone speed sound of 1.5 to 2.

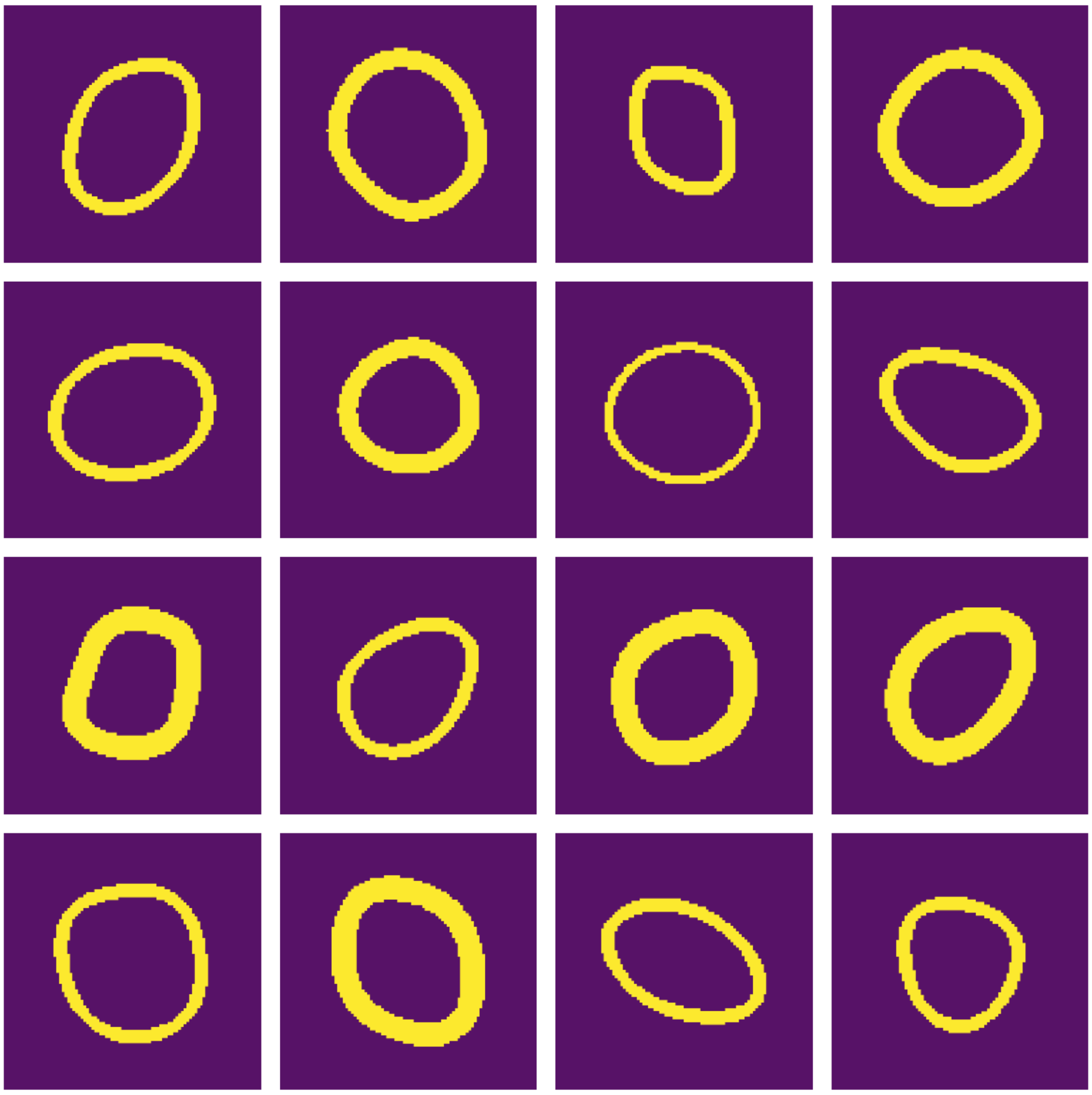

Figure 3.3: Samples from the training dataset [63]. Since the physics-based loss is used, we do not need any target wavefields.

3.3 Iterative solution using trained solver

As we mentioned in section 2.4.2, we aim to use a learned optimizer instead of solving Equation 2.2 directly. Thus the iterative optimization equation can be written as follows:

$$u_{k+1} = u_k + f_\theta(u_k, c, \rho), \tag{3.3}$$

where θ is the vector containing the model parameters of the learned optimizer, f_θ is the learned optimizer with following parameters - u_k denoting the previous solution, c denoting the sound speed distribution and the source distribution ρ. Source frequency ω is omitted here because it is the constant value.

This specific approach is, from some perspectives, quite problematic. Inputs to the optimizer f_θ are values that belong to very different domains, making the training process much harder and requiring a more significant amount of training data.

Even though the general idea will be preserved, instead of using the optimizer with inputs of c and ρ, we will apply knowledge of the forward operator $A(c)$ that we have already introduced in the Equation 2.2, and we will use residual e_k as follows:

$$u_{k+1} = u_k + f_\theta(u_k, e_k), \tag{3.4}$$

where residual e_k might be viewed as a physics loss:

$$e_k = A(c)u_k - \rho, \tag{3.5}$$

and $A(c)$ is defined by the following relation:

$$A(c) = \left[\widehat{\nabla}^2 + \left(\frac{\omega}{c(r)}\right)^2\right] \tag{3.6}$$

where members of equation corresponds to the Equation 2.2.

The final form of the iterative function contains a connection to the standard iterative solver, that is, the memory h_k [50]. From the lenses of optimization, we could also view this memory as a momentum which is commonly used in many similar optimization problems [19]. This the updated iteration is defined by this relation (see Figure 3.4 for schematic expression):

$$\begin{aligned}(\Delta u_{k+1}, h_{k+1}) &= f_\theta(u_k, e_k, h_k)\\ u_{k+1} &= u_k + \Delta u_{k+1}\end{aligned} \tag{3.7}$$

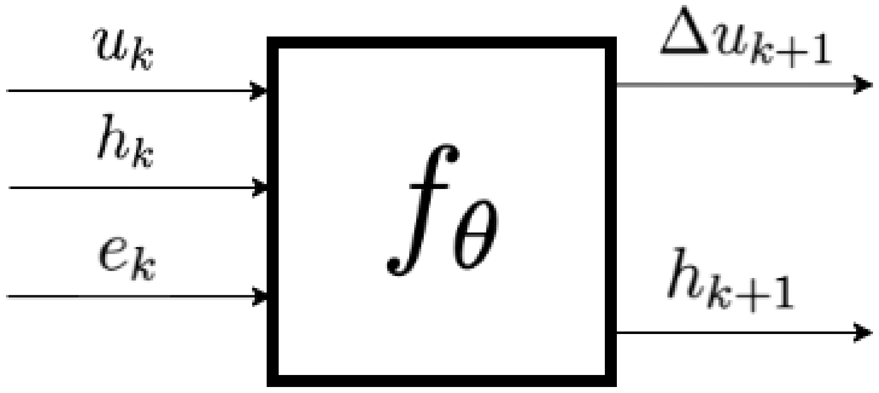

Figure 3.4: Figure shows the inputs and output of the learned optimizer. Residual e_k is computed using ρ, c, u_k with application of perfectly matched layer.

To sum it up, the function f_θ is considered an optimizer with memory h_k, enabling many strategies used in this work. The learned optimizer used in this work was first introduced for various image reconstruction problems. The residual was given by the gradient of the log-likelihood loss function for a Gaussian prior.

As we have already mentioned, function f_θ is a trained neural network with model parameters of θ, and e_k is a residual computed using a homogeneous Helmholtz equation with the application of PML. The network uses auxiliary input of the variable absorption coefficients that characterize the PML to cover the Sommerfeld radiation condition by simulating open boundaries even the domain of interest is enclosed.

The solution updates used in this work are similar to the Euler schema described by Lu et al. [42]:

$$X_{n+1} = X_n + f_n(Y_n) \tag{3.8}$$

The memory recurrent belief state h_k allows this work to use various optimizers. The optimizer is given by the information stored in the memory. Storing the second-order information (gradient and its magnitude) from the preceding step would make an optimizer similar to the method proposed by Andrychowicz et al.[2]. Instead of pure residual, they suggest to *Learning to learn by gradient descent by gradient descent*. This method is not used in this work, but we suggest exploring this option in the future.

Since our method requires a certain amount of iterations, we have to find a global loss function for our solution because loss defined in Equation 2.11 is usable for one iteration. If the number of iterations is constant, one approach can compute the loss function after performing all the iterations, but performing backpropagation over many iterations would be infeasible. Because of that, we decided to compute loss as a sum of all partial losses for every iteration:

$$R = \sum_{k=0}^{T} w_k L_k(u_k, c, \rho) \tag{3.9}$$

where R is the total loss, T is the total number of iterations, k is the iteration number, w_k is the weight of the significance of the loss in the k-th iteration, L_k is already mentioned physics-based loss (residual) and other symbols are knows from previous sections.

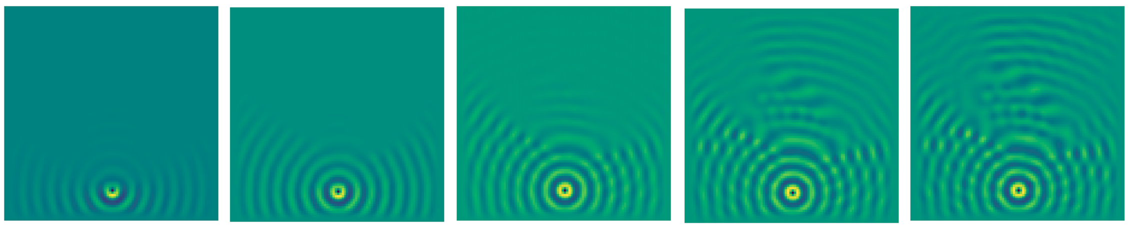

Figure 3.5: This figure shows the convergence progress to the solution by the 2D learned optimizer proposed in Helmnet framework. Images show iterations with numbers 1, 3, 10, 25 and 1000. We can see that the partial solution is very similar to the final result after the small number of iterations. This leads us to decrease the number of iterations to save computational time.

3.4 Extension of the Helmnet to 3D

Extending the Helmnet to 3D is suggested by Stanziola et al. [63] in the original paper. This is also necessary for therapeutic usage to provide information about wavefield in the whole brain model. As mentioned in the earlier sections, the 3D wave prediction is a time-critical issue, and it needs to be solved within seconds with adequate accuracy for neuromodulation. Indeed, we have to avoid very deep architectures to decrease time and memory limits.

Because of that reason, we kept architecture similar to the original Helmnet but adapted it for 3D inputs. The input layer of the neural network has the spatial dimensions to match the input data, that is, in this case, tensors with size 96x96x96. The input layer contains six channels, which consist of two channels for the real and imaginary parts of the complex wavefield u_k, another two channels for the real and imaginary parts of the complex residual e_k representing the physics-based loss, and three channels for the variable absorption coefficients σ_x, σ_y and σ_z, required for the Perfectly Matched Layer in every coordinate.

An important and frequently used block for this network is the double-3D-convolutional layer (DC3D). This layer consists of two convolutions with 3x3x3 kernels with non-linear activation of parametrized-RELU [27]. Their job is to compute the output passed to subsequent layers and compute the hidden state. See the illustration in Figure 3.6.

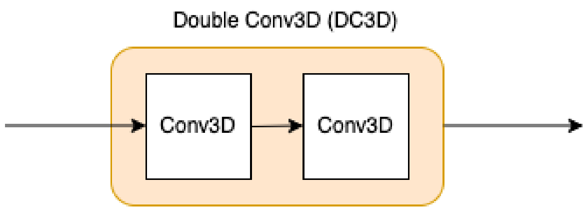

Figure 3.6: Two consequent Conv3D layers with kernel 3x3x3.

The encoding block is another entity (block) present in the neural network. Every encoding block contains two double-3D-convolutional layers. The first one accepts two inputs: the hidden state and the current input representation, and the second layer accepts output from the first one and recomputes the hidden state. The output from the first layer then splits into the second layer, the decoding block and follows to the downsampler and the encoding block at the lower levels, etc. The EB is illustrated in Figure 3.7. The size of the hidden state is the same as the size of the input and has two channels for real and imaginary parts of the complex number.

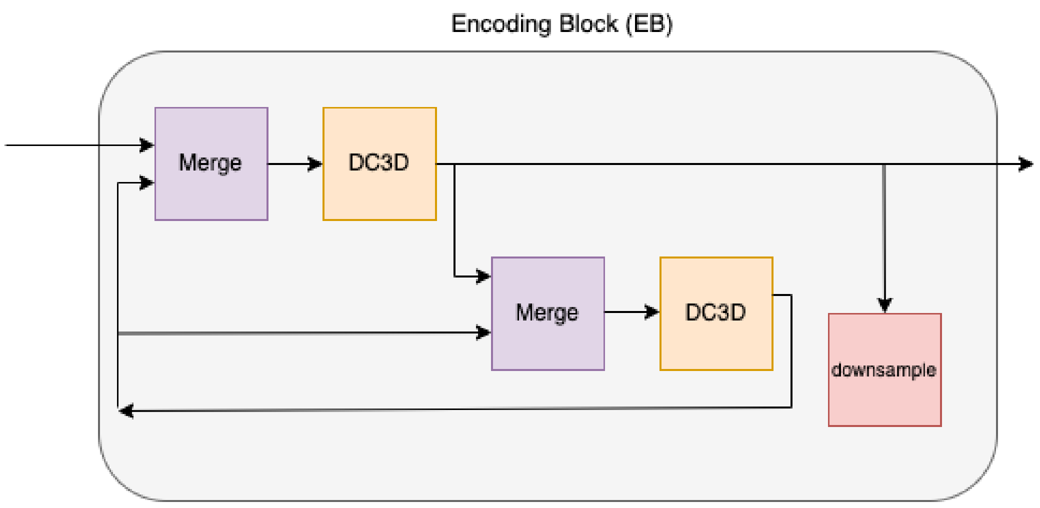

Figure 3.7: The encoding block consists of two *DC3D* such that first one sends its output to the EBs at the lower layers, to decoding blocks (out of EB) and to another *DC3D* to recompute hidden state h.

Encoding blocks are also connected to another type of block called the Decoding block (Figure 3.8). This block upsamples the output from the decoding block at the lower level and merges it with the output of the encoding block at the corresponding level.

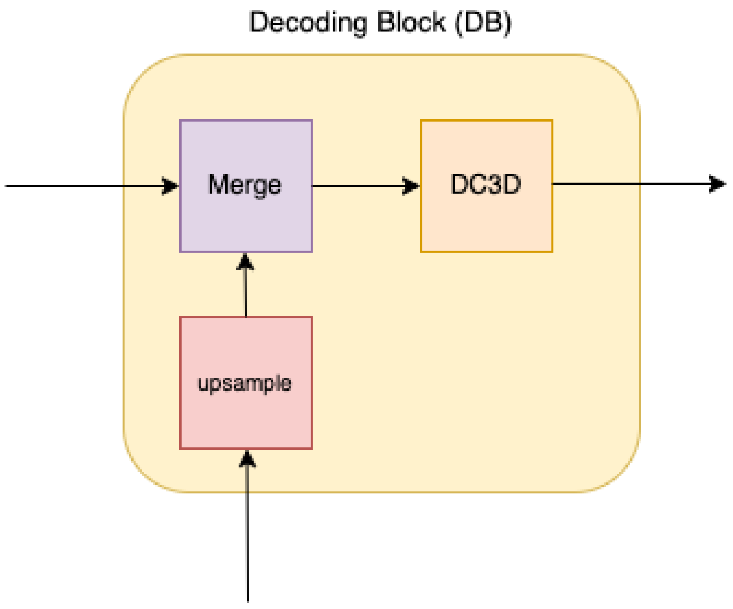

Figure 3.8: Encoding block merges upsampled output from lower-level DB and EB.

The general intuition behind this network is some level of the translation invariance of the iterative solver prescribed by the fully convolutional architecture. Another expectation from the network is the possibility of the local updates (as *GMRES*) with the combination of long-range handling dependencies and global updates. This property should be achieved by multiscale cascade structure (encoder-decoder structure) [26]. The proposed network (Figure 3.9) has 48160 parameters, but there is a chance that we will need a more heavyweight network for the effective 3D solver. Initial measurements of this architecture showed that inference by the model lasts about three times more than in the case of a 2D network. Assuming this (or similar) architecture will work and converge similarly to a 2D solution, we can expect that the complete prediction of the wavefield for a sample with a size 96^3 would take about 12 seconds. Thus we will have to think about accelerating this computation.

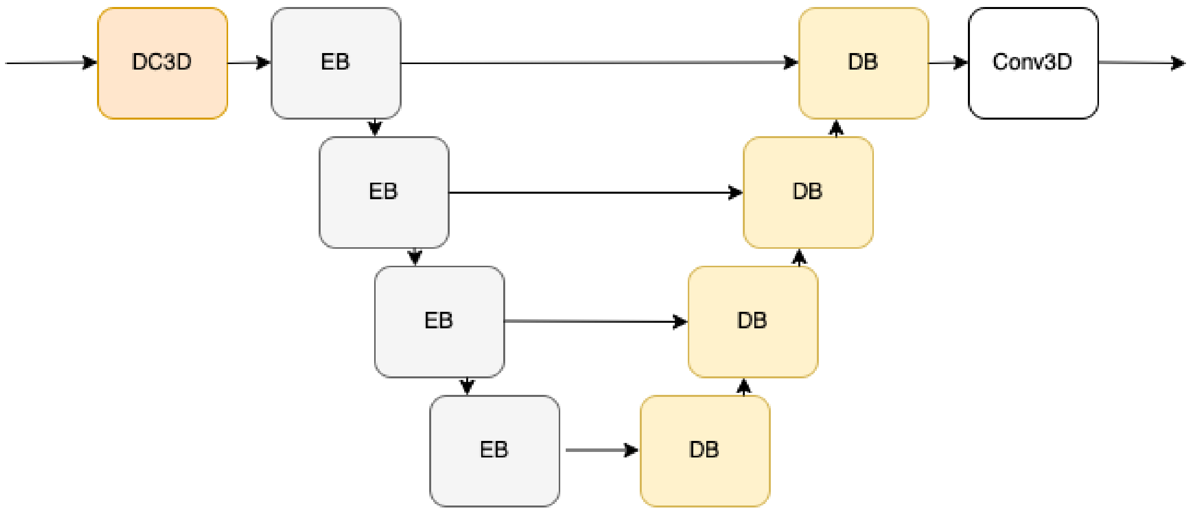

Figure 3.9: This is the first proposal of the neural network for a 3D optimizer. The network is very lightweight because it contains only 48160 parameters. This number of parameters might be changed in the future to better generalizability.

3.4.1 Training of the 3D Helmnet

Since the beginning of the project, it was expected to be very computationally expensive to train the neural network for the 3D data, even though the number of parameters is not much higher than in the case of the original 2D Helmnet. As shown in Table 3.2, the inference of the 2D solver took approximately 4 seconds and training about 9 hours. This suggests that the training of the one dimension higher model could last days on the same device. Moreover, the memory requirements could be an order of magnitude higher, and there was a concern that we would need to adjust parameters to require even more resources.

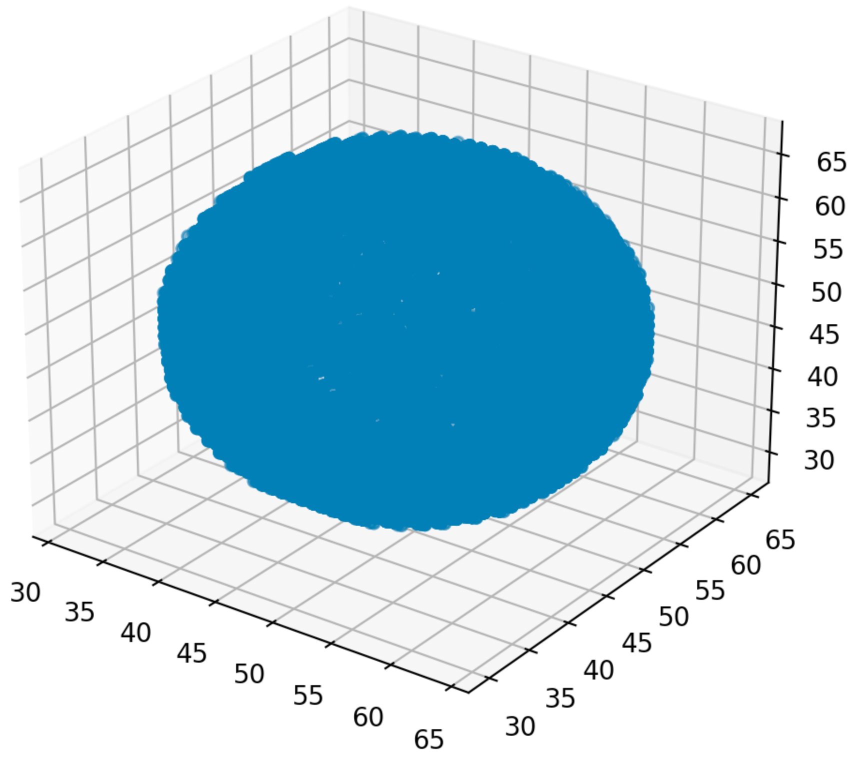

Figure 3.10: Figure shows the artificially generated sphere representing the shape of the human skull.

The dataset for 3D Helmnet consisted of 9 000 3D sound speed maps. 8 000 samples were used for training, 500 for validation and 500 for final evaluation. The evaluation data has generated also reference solutions using k-Wave. The reference solution is unnecessary during the training, thanks to the physics-based loss function. The thickness of the skulls in the dataset is between 10 to 15 pixels, the background sound speed is $1m/s$, and the source frequency is $1Hz$. During the training was source position selected randomly. It is suggested to restrict source position only to meaningful ones (near the outer side of the skull). Placing a radiation source inside the skull might help train the model in general, but it will probably lose some of its accuracy for real-life usage.

The size of the training data was 96^3 (see Figures 3.11 and 3.10). 3D data are generated by summing up several spherical harmonics with random parameters to simulate differences of the skull in population [24].

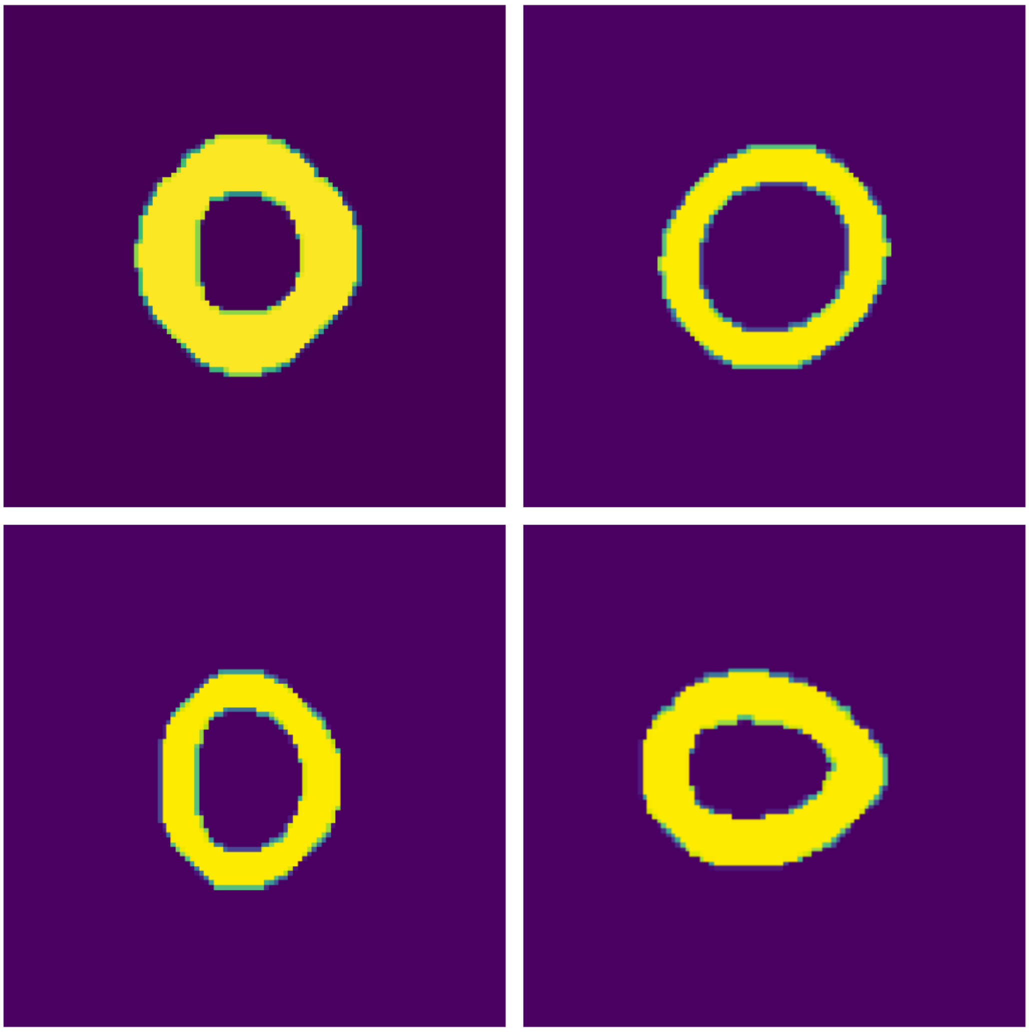

Figure 3.11: Figure shows the artificially generated 3D data cut on one of the axis to visualize better.

This method will require tens to hundreds of iterations to achieve an acceptable solution. From the memory and computational power perspective of view, it is not feasible to perform backpropagation updates through all iterations, so we are using *truncated backpropagation through time* (TBPTT) [66] in combination with replay buffer [30]. The number of truncated backpropagation steps is in our work 10, but it is suggested to try a higher number of TBPTT steps in the future work, according to computational abilities. The replay buffer contains the experience with samples in various stages of evolution. That is for enabling:

- fast improvement in early stages
- precise improvements in late stages
- preserve the solution when improvement is not possible

The replay buffer contains triplets T_b:

$$T_b = (c, u_k, h_k) \tag{3.10}$$

where c is the speed of sound distribution, u_k is the wavefield in k-th iteration and h_k is the memory in the k-th iteration.

The samples in the replay buffer are chosen randomly to preserve the uniform distribution of states in all iterations from zero to maximal limit. In our implementation is the number of replay buffers 100. The buffer size of 100 was just an unresolved workaround because the larger samples in the buffer would not fit into the RAM with unchanged other parameters and hardware resources. It is strongly suggested for future work to explore ways to expand replay buffer size. One of the possible solutions to decrease memory requirements would be gradient checkpointing. To achieve better results, we suggest using replay buffer size B_s:

$$B_t \approx \frac{10k_{max}}{\alpha} \tag{3.11}$$

where B_t is the replay buffer size, k_{max} is the maximal number of iterations and α is the number of TBPTT steps. The longer the TBPTT chain is, the less amount of samples we need, and the buffer size should also reflect the iteration amount. We need to increase chances that almost for every iteration there is at least one sample in the buffer.

Then the loss of one batch is based on loss defined in Equation3.9 with uniform weight (all weights $w_k = 1$):

$$R_{batch} = \frac{1}{N}\sum_{n}^{N}\sum_{k}^{\alpha} L_k(u_{k,n}, c_n, \rho) \tag{3.12}$$

where R_{batch} is the loss of the batch with size N, k is the iteration index in TBPTT steps, α is the number of TBPTT steps, L_k is the loss, $u_{k,n}$ is the predicted solution on given indexes, c_n is the corresponding sound speed distribution (same for whole TBPTT), ρ is the source distribution.

The loss calculation also uses Laplacian ∇^2 to include the impact of PML to approximately satisfy Sommerfeld radiation condition [76].

Every training step performs the fluctuation in the replay buffer. The randomly selected sample in the buffer is replaced with a new, suitable one. In this work, we used the maximal iteration number of 500. Using a higher amount of the maximal iteration is suggested. In this work, we were restricted by the memory limits.

Since we only used TBPTT steps $\alpha = 10$, which is a relatively small number compared to required iterations, there was a concern that the model would learn short temporal dependency and would not be able to improve for a longer time. It is suggested to explore the possibility of unbiasing the TBPTT [66], but the phenomenon does not significantly influence this work, and we can perform hundreds of iterations successfully. Because of that, we assume that pushing the required iteration number down has a better perspective for future work. Related work proposes also using a so-called experience buffer for storing memory h_k instead of generating it as our work does [37].

The optimizer used in our work is Adam [38] with a batch size of only 4 (!). Our memory-expensive implementation conditions the number of samples in one batch, and it is strongly suggested to use higher values in future work. This work uses a learning rate of 10^{-4} and Xavier weight initialization [40].

The condition for storing experience in the replay buffer is that the iteration of experience k is lower than maximal iteration $k < k_{max}$ and that the residual L of the sample is below L_{max}. The L_{max} used in this work is 0.7. The description of the implementation is in the Chapter 4

Algorithm 2 Training of the 3D helmnet

```
Require: Training data $\mathcal{T}$, Validation data $\mathcal{V}$, source distribution $\rho$, buffer size $B_t$, maximal iteration $k_{max}$, TBPTT steps $\alpha$, batch size $N$
  $i \leftarrow 0$
  while $i < |B_t|$ do                                  ▷ initial filling the replay buffer
      $i \leftarrow i + 1$
      $k \leftarrow \mathcal{U}(0, k_{max})$
      $c \leftarrow \mathcal{T}_{i_c}$
      Random init $u_k$, $h_k$
      Store ($c$, $u_k$, $h_k$) in the replay buffer
  end while
  $\hat{V} \leftarrow \infty$                                       ▷ Best model loss
  for $epoch \in N_{epochs}$ do
      Randomly sample batch with batch size $N$ out of $B_t$
      for $e \in N_{batch}$ do
          for $i \in \{0, ..., \alpha\}$ do
              $res_{i-1} \leftarrow A(c)u_{i-1} - \rho$
              $u_i \leftarrow u_{i-1} + f_\theta(u_{i-1}, res_{i-1}, h_{i-1})$
          end for
          $L_n = \sum_i L_{i,n}$
      end for
      $i \leftarrow \mathcal{U}(k + 1, k + t)$
      if $i < k_{max}$ & $res_i < 0.7$ then
          Store experience ($c$, $u_i$, $h_i$) into the replay buffer $B_t$ instead of experience $e$.
      else
          $c \leftarrow \mathcal{T}_{i_c}$
          Random init $u_k$, $h_k$
          Store ($c$, $u_k$, $h_k$) in the replay buffer
      end if
      $R \leftarrow \frac{1}{N_{batch}} \sum_n L_n$
      $\mathcal{G} \leftarrow \nabla_\theta R$
      $\theta \leftarrow SGD(\theta, \mathcal{G})$
      if $epoch\%10 = 0$ then
          $L_{val} \leftarrow 0$
          for $d \in \mathcal{V}$ do
              Random init $u_k$, $h_k$
              Random init $\rho_v$
              for $i \in k_{max}$ do
                  $r_{i-1} \leftarrow A(c)u_{i-1}\rho_v$
                  $u_i \leftarrow u_{i-1} + f_\theta(u_{i-1}, e_{i-1}, h_{i-1})$
              end for
              $L \leftarrow L_{k_{max},n}$
              $L_{val} \leftarrow L_{val} + L$
          end for
          if $L_{val} < \hat{V}$ then
              $\hat{V} \leftarrow L_{val}$
              Save $\theta$
          end if
      end if
```

3.5 Summary

The chapter provided insight into the topic of the physics-informed neural networks described in 2D Helmnet and mainly introduced our proposed solution for solving the Helmholtz equation using a neural network.

The section 3.1 reminds the 2D Helmnet proposed by Stanziola et al. [63]. The section contains experiments we performed by reproducing the solution and describes the evaluation methods used in the paper. We pointed few most impressive achievements and summarized the required resources for training and inference using the model. We also showed hyperparameters and other configurations required to reproduce our results with 2D Helmnet.

In section 3.2 we described the artificially generated dataset for 2D model training and also the generation of the reference data. The data generation process is simple and uses only circular harmonics to model a skull sample. We adopted this approach for 3D.

The vital section of this chapter was section 3.3 that described the concept of wavefield inference using a learned optimizer. It was justified within this section the input and output shape of the neural network architecture and explained important terms like *belief state* or *residual*. The introduced loss function referenced the physics informed neural networks mentioned earlier in this work. At the end of the section, we briefly demonstrated the process of wavefield prediction using this method.

The section 3.4 showed the entire design of the 3D solution. The section closely explains all neural network architecture building blocks, including Double Convolutional BLock, Encoding Block, and Decoding Block. It is essential to remind that the total size of the neural network is only slightly larger than for 2D. The reason is that a much larger model would extend the time required to accomplish wavefield prediction. The input and output sizes and channels (residual, absorption coefficients) are also mentioned in this section. This section also includes a few proposals for future work.

Section 3.4.1 described the whole process of training 3D Helmnet, differences from the 2D original version of Helmnet and explained essential concepts (like the *truncated backpropagation through time* (TBPTT)). Except for that, the data generation process using spherical harmonics is described here.

The data are generated as simple as in the case of 2D using the automated script. The reference solutions are calculated using software k-Wave and its `kspaceFirstOrder3DG` function. In this section are a few examples of the training dataset, but since the visualization of 3D data is a challenging task, the samples are usually cut by one axis to achieve a 2D image. The cut point is selected to show some contours of the skull model. If the cut were at the index 0, only the background would be visualized.

The following part of the section explains the replay buffer for TBPTT. This buffer is responsible for the learning chain of the wavefield improvements. The section also described that the data in the replay buffer must have a uniform distribution in terms of iteration number, sure accuracy (residual), and the size of the buffer must be sufficient. We also pointed out that the replay buffer size and content should be the objective of future work because different replay buffers could yield a model with different (potentially useful) properties.

Furthermore, last but not least, the section reminds some downsides of our training, like using the batch size of only 4. This was not selected by careful selection but because of the lack of memory.

Chapter 4

Implementation

This work's main objective and achievement is the proposal and evaluation of a novel method for transcranial ultrasound therapy. Implementing the method is not trivial, and it is essential to use external dependencies and well-known frameworks for its successful realisation. The current implementation uses the ML framework *Pytorch*[1]. The critical aspects of the solution to our governing problem are speed and accuracy. Many of our recommendations for future work consisted of improvements conditioned by better computational and memory resources. However, plenty of the improvements would be directly caused and enabled by the better implementation. Because of that, the code provided by this work is only considered experimental and requires refactoring to be used in real industry.

This chapter describes the technical aspects of the algorithms used in this work and their realisation. The main goal of this chapter is to reveal weak and strong spots of our program to enable improvement in future work. As we have already shown, our solution successfully competes with classical methods with finely optimised implementation. Nevertheless, it is still expected that only by improving the implementation we could achieve even better results and provide solutions usable in practical applications. It is strongly recommended for future work to employ a faster framework, e.g. *Jax*[2] to improve the performance of the program.

4.1 Structure of the project

This section describes the modules of the Python part of the implementation. Generally, the following modules are responsible for training the model and its inference. Code for experiments and data generation is not essential for understanding the 3D Helmnet or future work. This is omitted from the explanation.

- **derivative.py** This module contains simple structure holding the FFT-based derivative [33]. This is used by the PML.
- **double_conv.py** The file contains frequently used layer in our architecture, called Double Convolutional (DC) Layer. The layer simply consist of two instances of `Conv3d` by `torch.nn` interleaved by parameterized rectified linear unit (PReLU) activation function. The kernel size used for 3D convolutional layers is 3 and the padding 1.

[1] https://pytorch.org/

[2] https://github.com/google/jax

- **encoder_block.py** The module encoder_block contains class implementing functionality of the encoder. The encoder is the left part of the UNet (see Figure 3.9). The encoder consists of double convolutional layer and 3D convolution with kernel size 8, padding 3 and stride 2. It is responsible for downscaling the input in order to achieve multiscale solver characteristics.

- **experience.py** The file holds structure containing information about experience in the replay buffer. The experience consists of the predicted wavefield, hidden belief state, speed of sound distribution residual, source distribution and number of the iteration.

- **fast_laplacian_pml.py** The Laplacian is used for calculating the physics-based loss. This module provides Laplacian operator with included PML to enable computing the residual.

- **hparams.json** Configuration file, contains not only hyperparameters, but general config of the program. This file simplifies experiments with various source position, replay buffer size, number of unrolling steps, frequency, etc.

- **output_later.py** The class defines module of the neural network responsible for the last layer. The number of ouput channels is 2 (real and imaginary part of the wavefield). The used kernel size is 1 and the last layer is also 3D convolutional layer.

- **replay_buffer.py** The file holds the class responsible for modeling the replay buffer. The important functionality is behaviour similar to list and option to return random sample from the buffer.

- **solver.py** This script is the runnable core of the whole implementation. The script contains the class `IterativeSolver3D`, that is a module from GPU-accelerated *pytorch-lightning*[3]. The module performs training, validation and eventually inference on new samples. After finishing the training is only important function for inference method *forward*, however to calculate residual and overall error is also important module with Laplacian. By running the file is started the training with the parameters defined in *hparams.json*. Training is performed by the class *Trainer* provided by interface of *Pytorch Lightning* and during the training is saved 10 models with the best validation loss. The validation is performed using random source position, using validation dataset (500 samples). The training is performed similarly using random source positions, using training dataset (9000 samples). We suggest exploring possibility of validation using similarity metric and comparison to reference data instead of using residual.

- **source_module.py** This file is responsible for suitable representation of the source distribution and providing random source distributions.

- **unet_3d.py** Architecture of the neural network based on UNet. Uses encoder for encoding and downsampling the data, and encoder combined with the layer `ConvTranspose3d` for encoding and upsamplig the result. The depth of the UNet is given by the expected size of the training samples of 96, thus the depth is 4. Using larger domains of the training samples could be infeasible, so the depth and size of the training samples is still a stable part of the implementation, not the configuration.

[3]https://www.pytorchlightning.ai/

- **utils.py** provides heler functions, including loading the configuration, or the pre-trained model.

4.2 Proposals for the future work

As we already mentioned in this and other previous chapters, this work omitted many potential improvements. This section summarized them.

- **higher training domain**; despite the fact that this work showed that our solution successfully generalizes to higher domains, it is recommended to explore options of using larger data for training. This could lead to increasing the size of the network, which would prolong the inference of the one iteration. However, this could decrease the number of the required iterations.
- **replay buffer content**; we recommend for the future work to analyze the content of the replay buffer during the training. This work is not checking the distribution of i for wavefield in given iteration w_i, thus we cannot show what iterations is model learning in given stage of the training. Controlling it in can reveal other lacks in this work. It would be also useful to explore ways how to specialize the model only for the brutal changes in the beginning stages of the wavefield optimization, or for fine tuning at the end. Controlling the samples in the buffer can even bring various models for every stage of the optimization.
- **batch size**; the significant restriction of this work is model trained only with batch size 4. This is much lower than commonly evaluated or used batch size [36]. The reason for this is lack of memory. We strongly recommend evaluate models trained with greater batch sizes and compare the training process.
- **gradient checkpointing**; It is expectable that even with the machines with larger memory resources, it wouldn't be possible to study all possible parameters and options. We recommend to use the *Gradient Checkpointing* introduced by Chen et al. [16] in 2016. This method would enable training larger-than-memory models. Common libraries like PyTorch provide this method as a part of the framework.
- **TBPTT steps, buffer size, maximal iteration**; These parameters are binded and should be modified while the Equation 3.11 is satisfied. If it is possible from the memory point of view, the increase of the unrolling steps could improve the way of converging to the solution. On the other hand it might be desired to decrease the number of iterations at all and try to specialize the model.
- **Single source position training**; Stanziola et al. [63] used one source position for the whole training process in the original 2D Helmnet. In case of 2D, the training using one source position generalized perfectly to any other source position (located in the place similar to real ultrasound). Our experiments showed that the model trained on the single source position overfits to the position. This does not necessarily mean it is a useless model. Eventually, the trained model provided as a result of this work is trained by random source positions. We strongly suggest restrict the possible source position only to positions that are meaningful for the therapy. Another suggestion is to explore the performance of single source position. If the model would perform

much better, there is a way to use it and rotate the skull accordingly to achieve the result.

- **Gradient clipping**; Stanziola et al. [63] used gradient clipping [17] for Helmnet.
- **Absorption**; another remedy might be changing the absorption coefficients of the PML.

4.3 Summary

This chapter briefly described the state of the current PyTorch-based implementation. The main contribution is the proof of concept, but we still need improvements and optimization for industrial usage. There are still many potential parameters and modifications that could bring us better performance and overall results. It is strongly recommended to study these improvements in the future.

Chapter 5

Experiments

This work aimed to achieve similar success as 2D Helmnet. The first step towards 3D Helmnet was reproducing end evaluating 2D results. All experiments within this work was performed using GPUs NVIDIA A100[1] on supercomputer Karolina[2]. This chapter shows the brief evaluation of the 2D Helmnet proposed by Stanziola et al. [63] and our results for 3D in comparison to GMRES [59] and k-Wave [68]. Training processes are performed using parallel 8 GPUs, the maximal number of GPUs in one allocated node by Karolina.

5.1 2D results

As we showed in Table 3.2, the time required for training 2D Helmnet using Karolina was about 9 hours per 1000 epochs. The validation loss is illustrated in Figure 5.1.

[1]https://www.nvidia.com/en-us/data-center/a100/

[2]https://www.it4i.cz/en/infrastructure/karolina

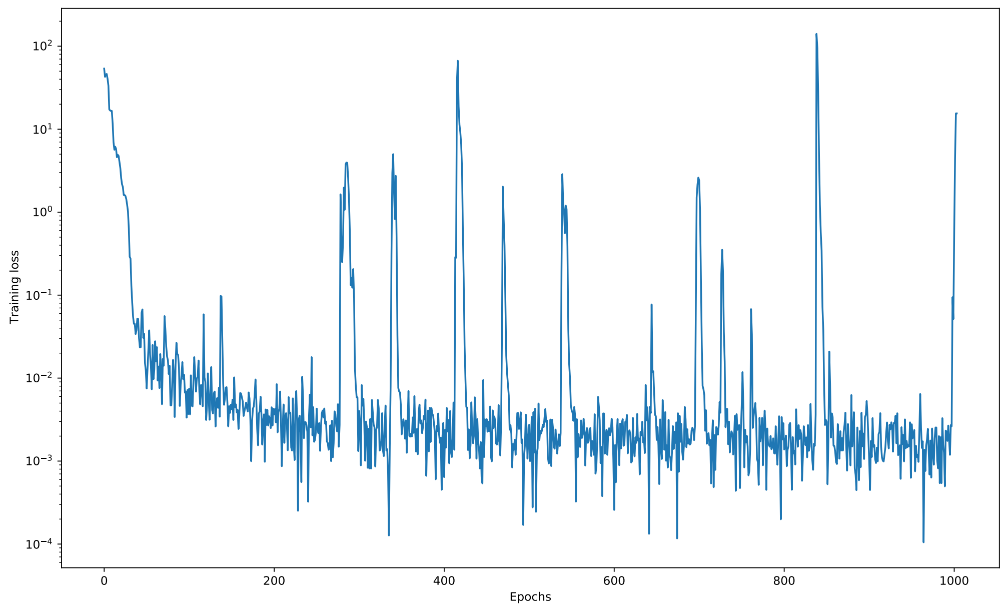

Figure 5.1: This chart shows the evolution of ℓ_∞ error with training epochs. This suggests decreasing the iteration number for training because the error is close to its minimum after less than a hundred iterations. However, it is essential to say that the loss decreases almost until the end of the training with 1000 iterations. The training took about 9 hours using 8 GPUs on the reference machine.

The evaluation confirmed the successful results described in the original paper [63]. The inference time with 1000 iterations is 4 seconds using GPU on Karolina.

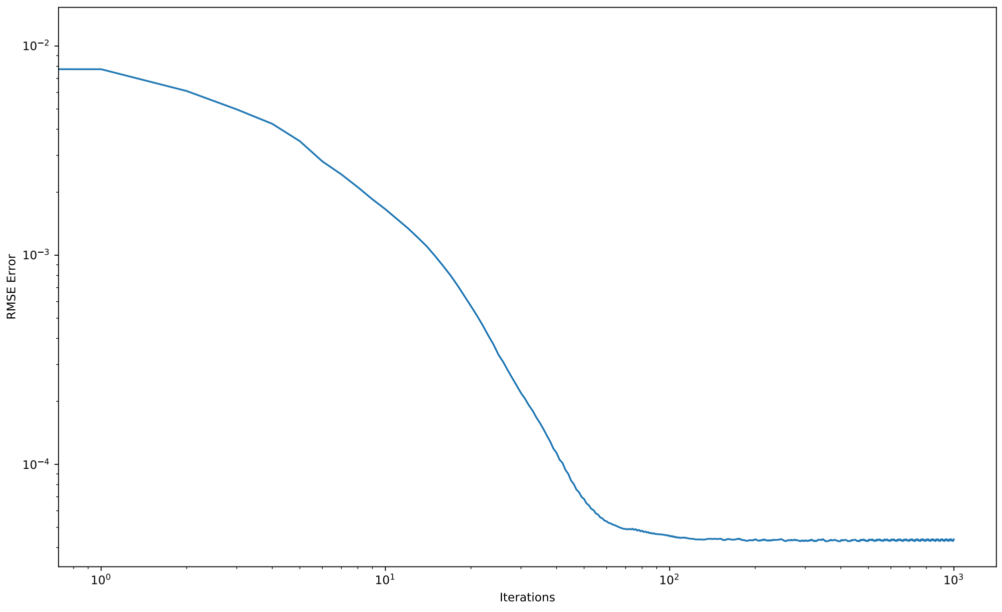

Figure 5.2: This chart shows the evolution of RMSE error according to the iteration number while predicting the wavefield. We may observe that the method converges quickly to values close to the minimal error. The minimal RMSE error is $4.285 * 10^-4$. However, the error of $9.40896 * 10^-4$ is achieved already after 44 iterations. The progress of the solution is illustrated in Figure 3.5.

5.2 3D Helmnet experiments

The previous section showed that we could reproduce results for 2D Helmnet proposed in the original paper. In this section, we present the results achieved in this work and also highlight potential improvements. Our experiments used the reference machine Karolina, as in previous experiments. The GPU used for experiments is also the same as in the previous experiments (NVIDIA A100[3]).

The base of our experiments contained our learned optimizer under test using other methods for comparison. The methods selected for comparison are function `kspaceFirstOrder3DG` by the k-Wave solver and GMRES method. The k-Wave software was used to generate reference solutions, and GMRES was used to show how fast and close the result to the reference solution can our method achieve compared to GMRES. Implementation of GMRES used in this work is provided by *Matlab* [4, 59].

Figure 5.3 showed the progress of training and validation loss while training the model. Our model was trained using 70 epochs. We suggest explore options to train using more epochs (e.g., model by Stanziola et al [63] was trained using 1000 epochs). The training took approximately 70 hours using 8 GPUs on a reference machine.

[3]https://www.nvidia.com/en-us/data-center/a100/

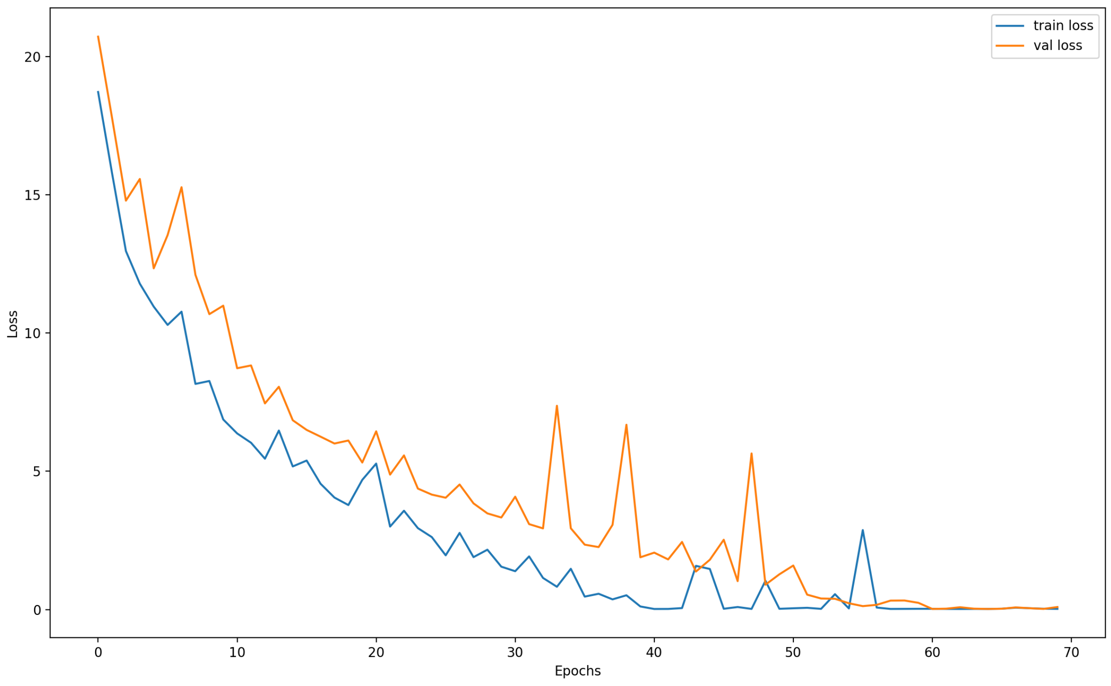

Figure 5.3: The chart shows the evolution of the training and validation loss of the learned optimizer. The model was trained using 9000 training samples with 70 epochs. The validation dataset consisted of 500 samples, and the model was trained and evaluated with a random source position.

5.2.1 Residual convergence

After training the model, the first step towards providing a learned optimizer is to verify the residual convergence to zero during the iterative solution. The chart in Figure 5.4 shows the 300 iterations by the 3D learned solver. The solved wavefield with a low residual is shown in Figure 5.5.

The chart in Figure 5.6 shows the oscillation of the residual between the 300th and 1300th iteration. Keeping in the replay buffer only samples with low quality caused faster convergence, but residual exploding from some iterations caused the model not to learn to handle optimized samples. This feature might be useless because we can quite accurately detect that the optimization is finished using the residual. Accelerated residual convergence using less optimized buffer content (with exclusively low-quality samples) should be the objective of future work.

These few experiments showed the general capability of our solver to optimize residuals and yield meaningful results. The following steps compare our results with traditional solvers, specifically k-Wave.

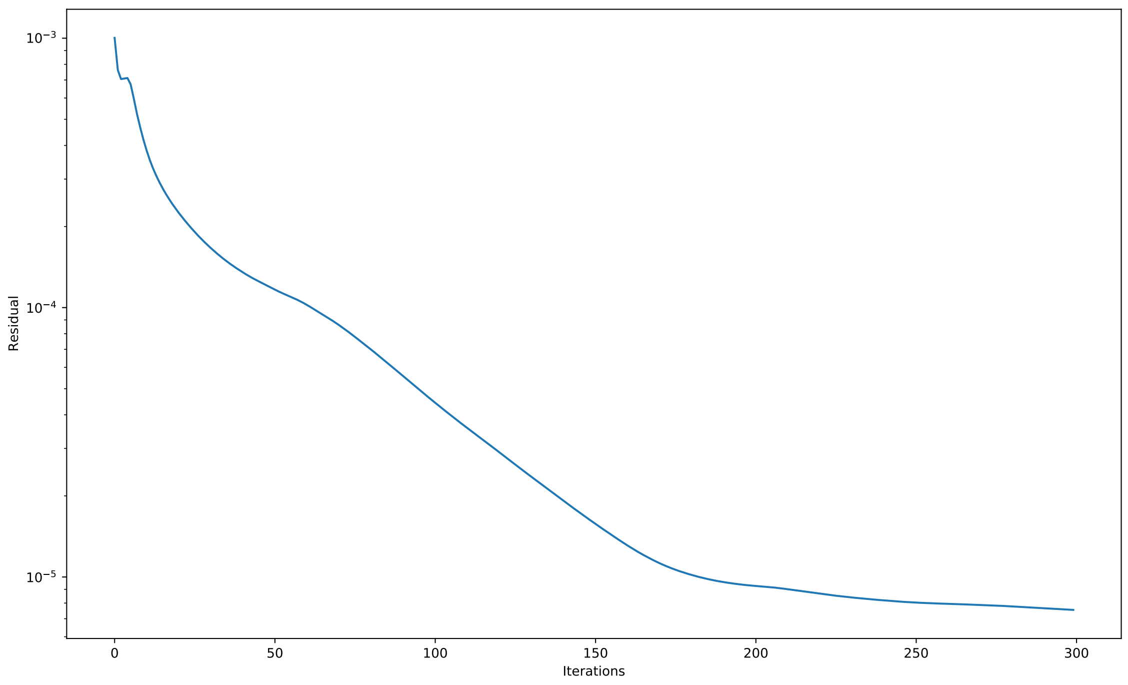

Figure 5.4: This experiment shows the iterative solution of the single sample with size 96^3 with the source at the position $(82, 48, 54)$ and with perfectly homogeneous sound speed distribution with value 1. The residual starts approximately at a value 0.001, and the best value $7.559 * 10^{-6}$ if achieved after 300 iterations

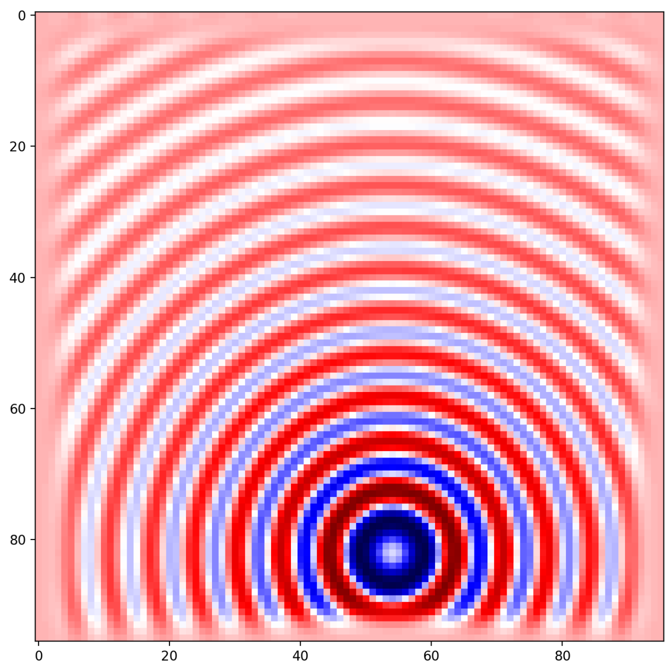

Figure 5.5: Figure illustrates the result of the 300 iterations of the 3D learned solver (Figure 5.4). Different structure at the edges is a sign of the PML. The source is placed at the position $(82, 48, 54)$ and this 2D image is cut on the index 40 by the y-axis. The whole process of inference took 8.5s using the reference machine Karolina.

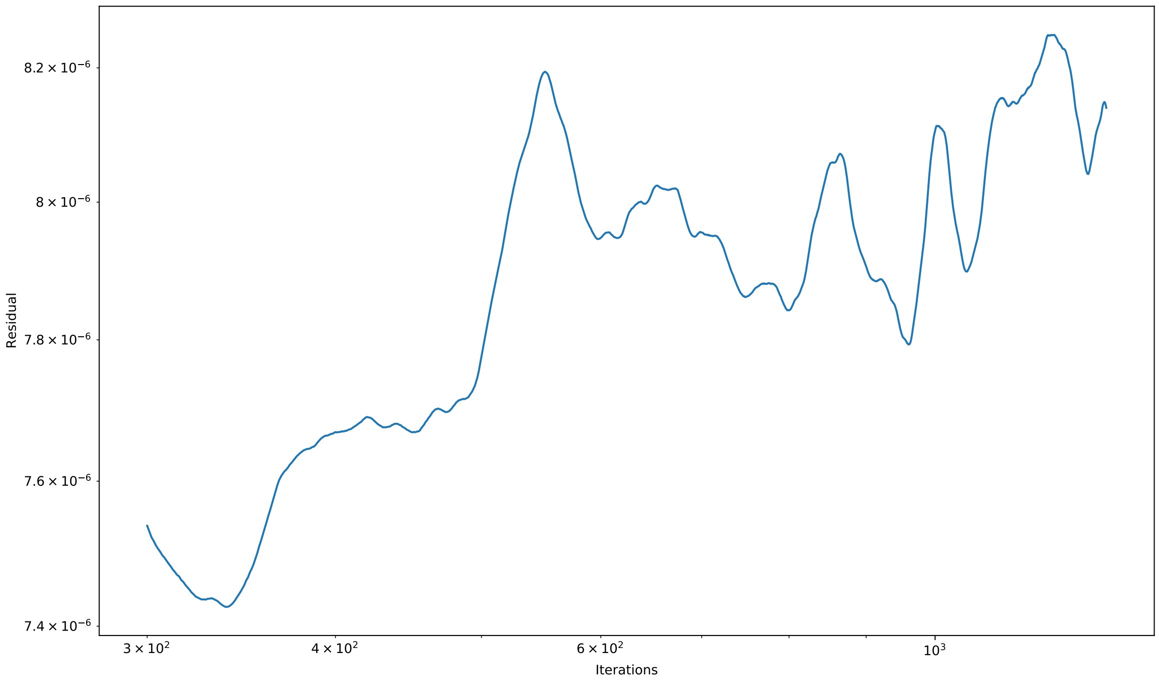

Figure 5.6: This chart shows the progress of the residual after the 300th iteration. The result is indistinguishable by the human eye from the result in Figure 5.5. Generally, after achieving some iteration, the residual is no longer improving but is oscillating in order of magnitude 10^{-6} up and down. This phenomenon is learned by keeping optimized samples in the replay buffer and performing only tiny modifications if the improvement is impossible.

5.2.2 k-Wave reference solution

The reference solution for this work was generated using the software k-Wave. Using the memory of our GPU (40960MiB) was the maximal possible domain of size $416 \times 416 \times 416$, which was sufficient for evaluation.

The first simple experiment demonstrates that the predicted solution achieves similarity to the reference solution by k-Wave. The comparison between the reference solution and the predicted one is in Figure 5.7. The prediction of the homogeneous environment is not helpful for industrial or medical usage. However, this simple example shows the ability to achieve the same results for the elementary problem as the classical method.

The downside is that 300 iterations took 8.5s to yield the given result, while k-Wave finished in 1.81s. Even though it seems that for the domain $96 \times 96 \times 96$ is the learned optimizer useless compared to k-Wave, the 3D Helmnet might still bring an advantage. Figure 5.8 shows an impressively fast convergence rate where the model is able after 25 iterations to achieve a visually very similar result with an error of only 1.02%. This is performed within 0.71s, which is two times faster than the k-Wave.

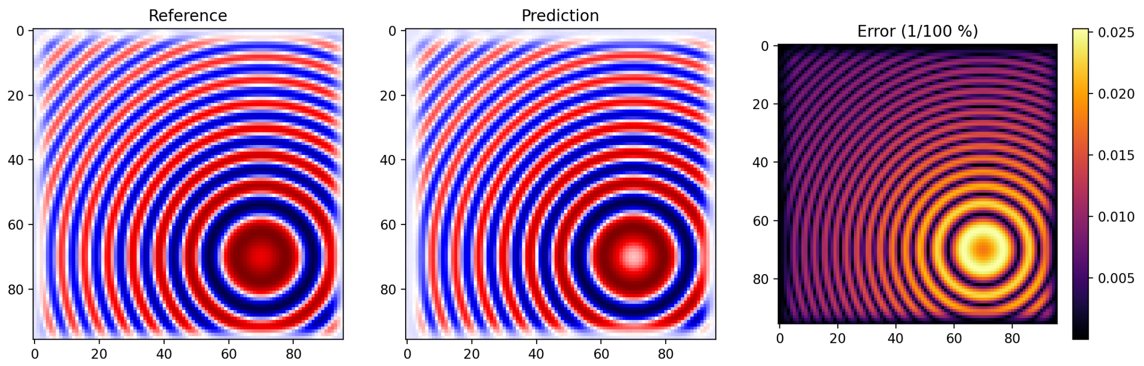

Figure 5.7: The comparison of reference solution by k-Wave and the prediction after 300 iterations. An average error is 0.927%. The source is located on position $(70, 70, 70)$. This is the cut on the y-axis (index 40). However, the cut by every axis should be the same.

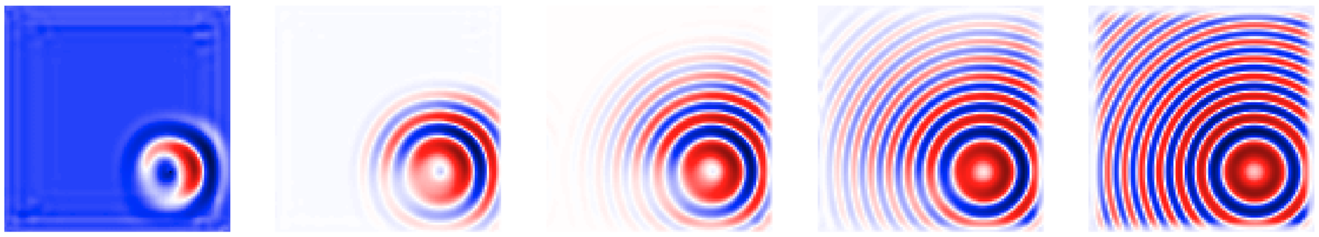

Figure 5.8: This series of wavefield images show relatively fast progress of the computation. In this order after the second, third, fifth, eighth and 25th iterations. The time required to accomplish 25 iterations is 0.71s, and the error compared to the k-Wave solution is 1.02%. The settings are the same as in Figure 5.7.

5.2.3 Multiple sources

Since the model was trained using one source, it is essential to verify its generalizability to domains with multiple sources. The usage in the therapy is not that common, yet still, there might be applied even for multiple beam sources [46, 61, 62]. In this work, we trained the model using one training source (at random locations) to apply to multiple sources. Figure 5.9 shows an accurate approximation with two sources applied, and Figure 5.10 shows three sources. The time required to compute wavefield for multiple sources is equivalent to one source in both methods. Notice that the error is slightly larger for more sources.

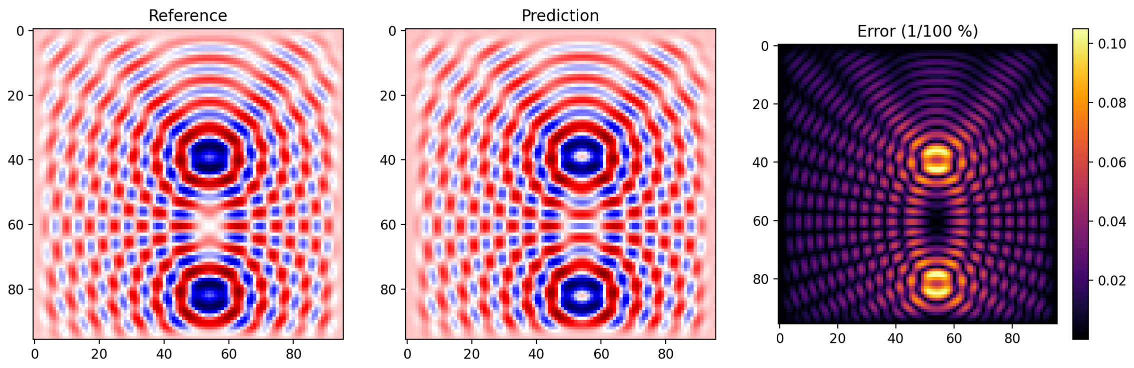

Figure 5.9: Predicted solution is computed using 50 iterations ($\approx$ 1.4s). The sources are at positions $(82, 48, 54)$ and $(39, 48, 54)$. The image is cut by axis y at index 40.

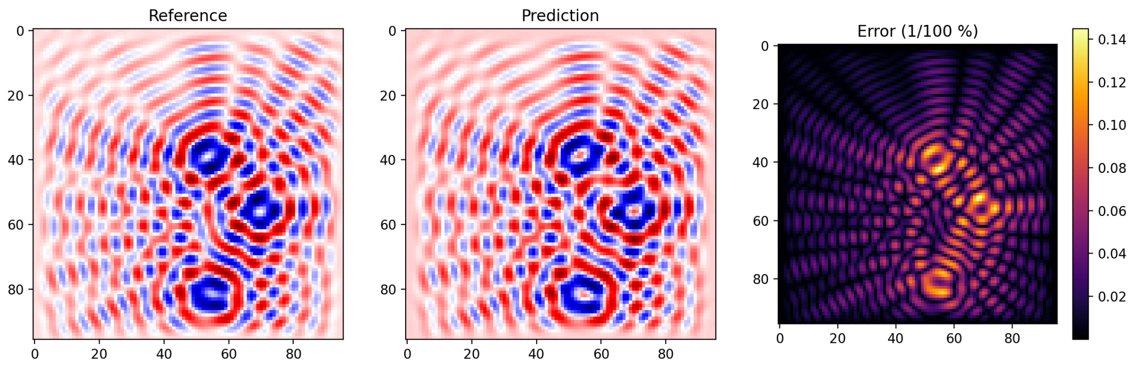

Figure 5.10: The predicted solution is computed using 50 iterations ($\approx$ 1.4s). The two sources are at the same positions as in Figure 5.9, thus positions $(82, 48, 54)$ and $(3948, 54)$. One more source is added at the position $(55, 48, 70)$. The image is cut by y-axis at index 40.

5.2.4 Large domains

Our solution accelerates the wave prediction, especially in the larger domain, compared to k-Wave. The larger domain is, the higher acceleration we achieve (the maximal comparable domain is 416^3 since our resources do not allow higher domains in k-Wave).

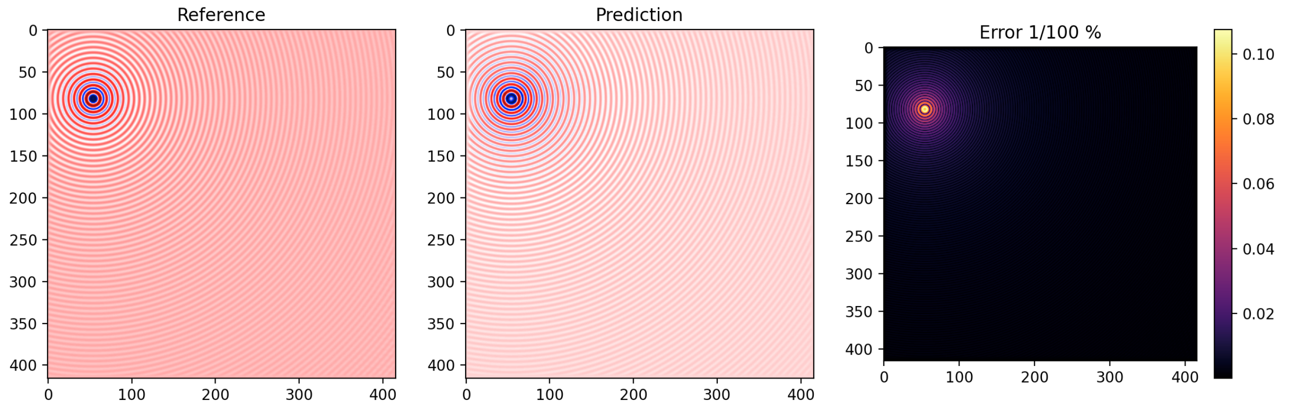

Figure 5.11: Our solution achieved a mean error of 1.05% after 32 iterations. The domain is 416^3, the largest domain we can compute using k-Wave. The inference by 3D Helmnet took $\approx$ 61 seconds using one GPU on Karolina, and the k-Wave simulation took $\approx$ 500 seconds, which makes more than 8-times accelerated calculation.

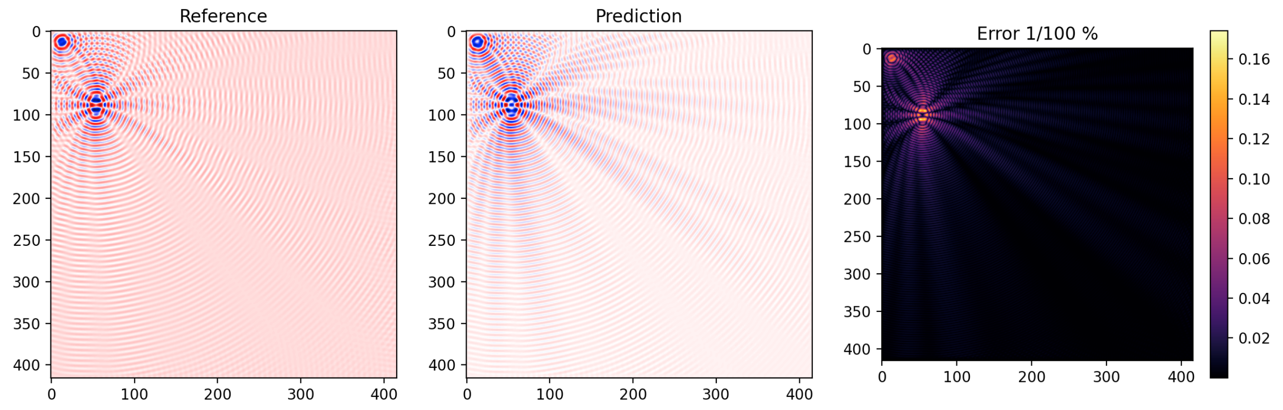

Figure 5.12: Figure shows three sources in the larger domain (416^3). The speedup using the 3D Helmnet is similar as in Figure 5.11

5.2.5 Missing dependency

The known issue of our solution is the weak influence on distant parts of the field from the source. The neural network was trained using the 96^3 domain and generalizes well until some distance from the source. This usually happens for larger domains if the radiation source is placed in one corner. The effect is the missing information about the waves in the other corner.

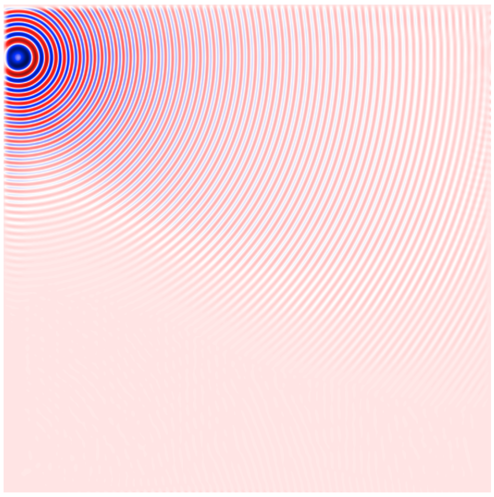

Figure 5.13: The domain with size 576^3 was not able to develop an optimal wavefield using Helmnet. When the solution started propagating to places further from the source, the computed part lost its information.

5.2.6 Homogeneous objects

The learned optimizer showed generalizability for predicting wavefields in objects with different morphology than the training data. Figure 5.14 shows the prediction of the wavefield with the present square in the region. Figure 5.15 even demonstrates this feature in a larger domain.

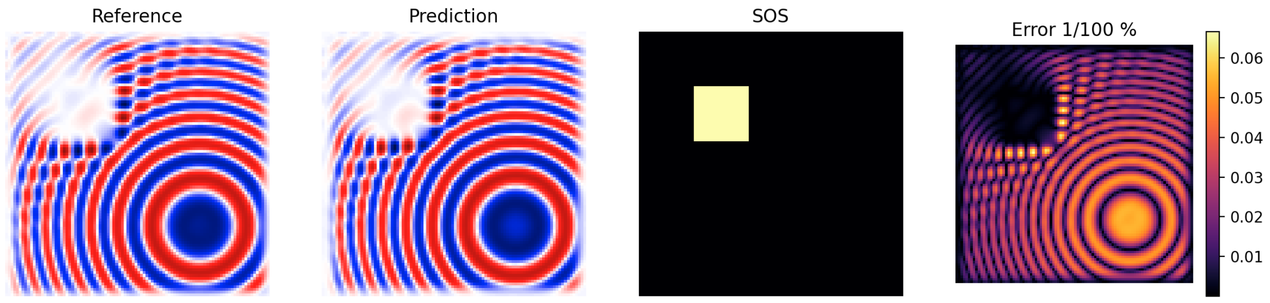

Figure 5.14: The wavefield prediction of the small square in the region with domain size 96^3.

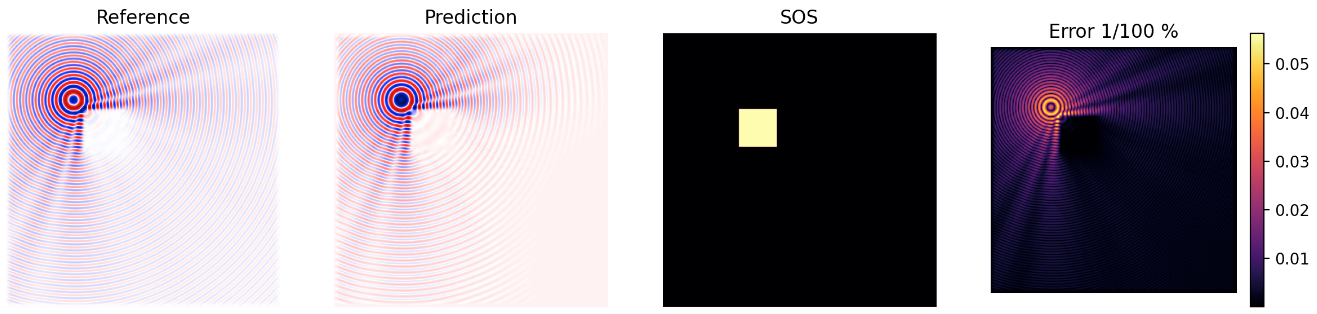

Figure 5.15: Similar experiment as Figure 5.14 but with the domain size 448^3.

5.2.7 Artificial skull

This experiment shows the ability to predict the wavefield of the artificial yet still similar data to real ones. Figure 5.16 shows the result of the wave prediction in the artificial skull cut by the y-axis.

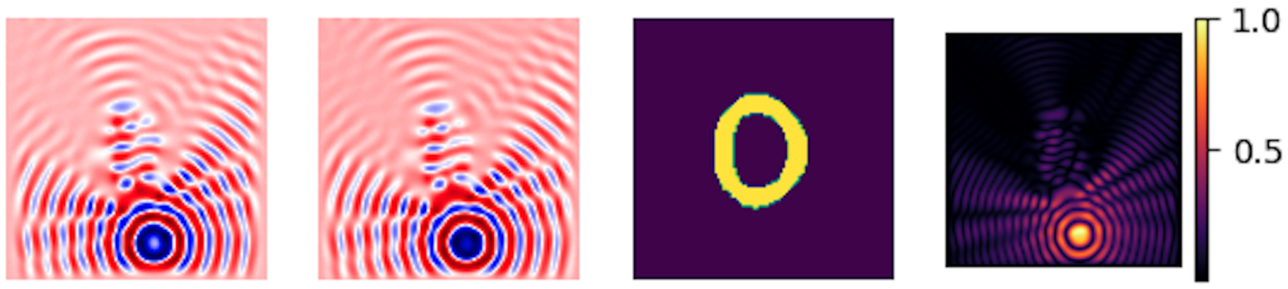

Figure 5.16: The residual of the predicted data was $\approx 7 * 10^{-6}$ and the difference from reference sample was 0.85%

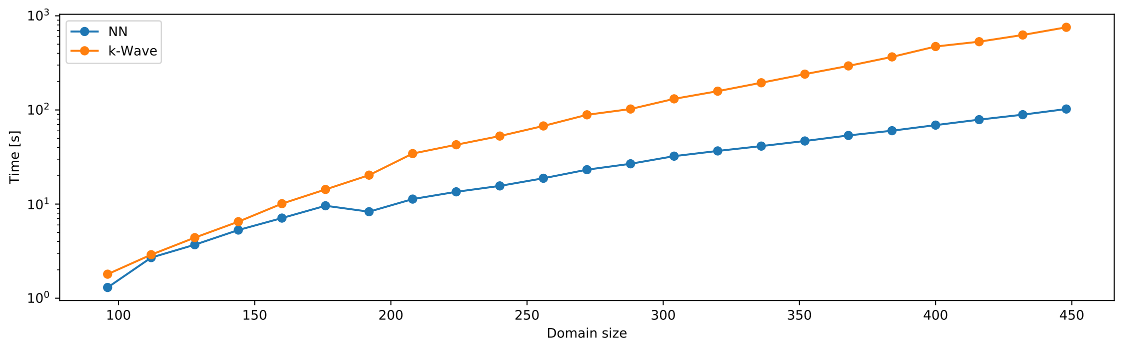

Figure 5.17: The chart shows the time required to compute the wavefield using our optimizer („NN“) and k-Wave. The axis „Domain size“ shows the size of the one side in the square domain.

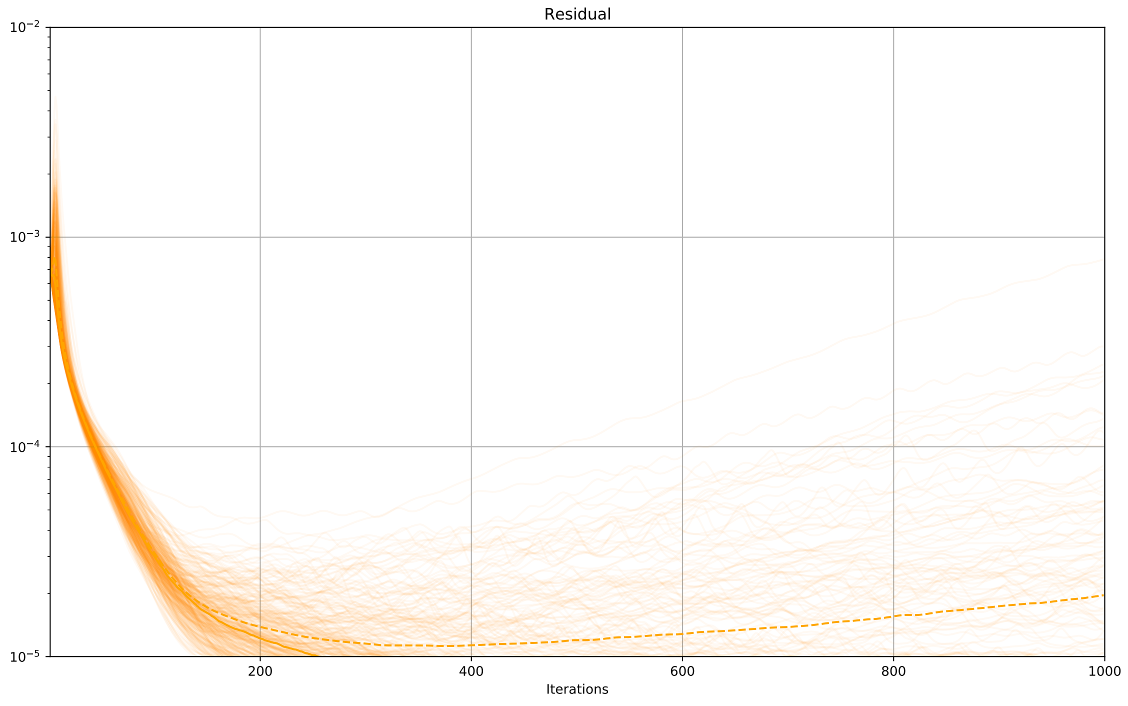

Figure 5.18: This figure shows evolution of 500 samples in 1000 iterations using learned optimizer.

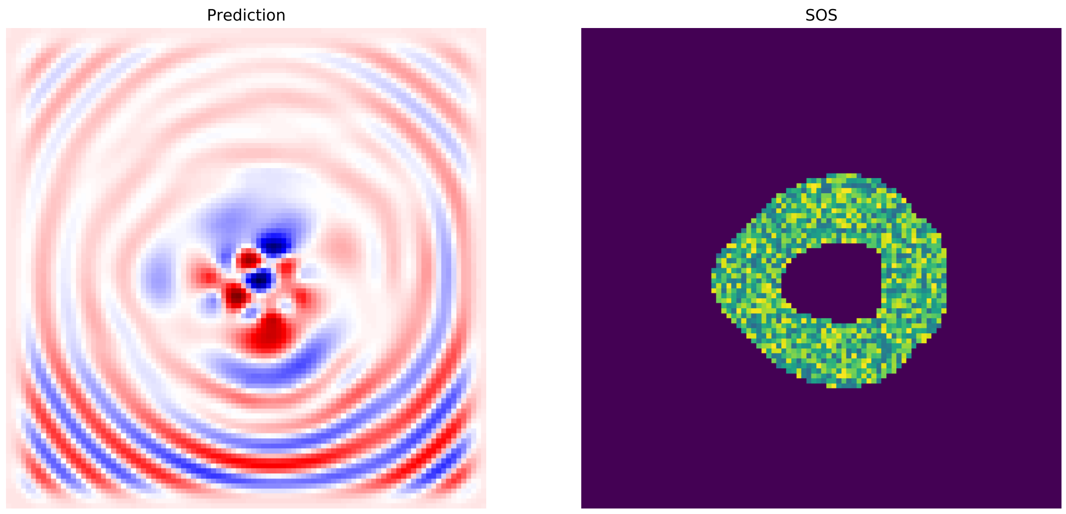

Figure 5.19: Even though the model was trained using entirely homogeneous media, we showed that the solution also converges for the media with heterogeneous structure. This sample was created using the injection of uniform noise between -0.3 and 0.3, and the residual achieved $\approx 8 * 10^{-6}$ after 250 iterations.

5.2.8 GMRES

Setting the system of equation in 3D for GMRES is difficult and we were able to solve the maximal domain of 64^3.

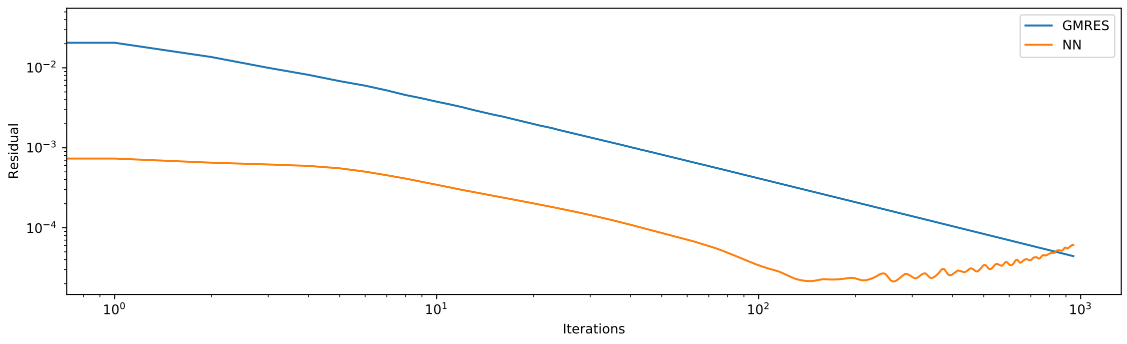

Figure 5.20: The „ NN “ curve is our learned optimizer and its performance on one sample. The curve „GMRES“ is a result of the GMRES method. The residual $4.4 * 10^{-5}$ was achieved by a learned optimizer after 80 iterations, while this was a result of a GMRES after 1000 iterations. Notice that our method does not converge to the perfect result, only to a particular approximation level. This is one of the issues of our method. The curves are related to Figure 5.21.

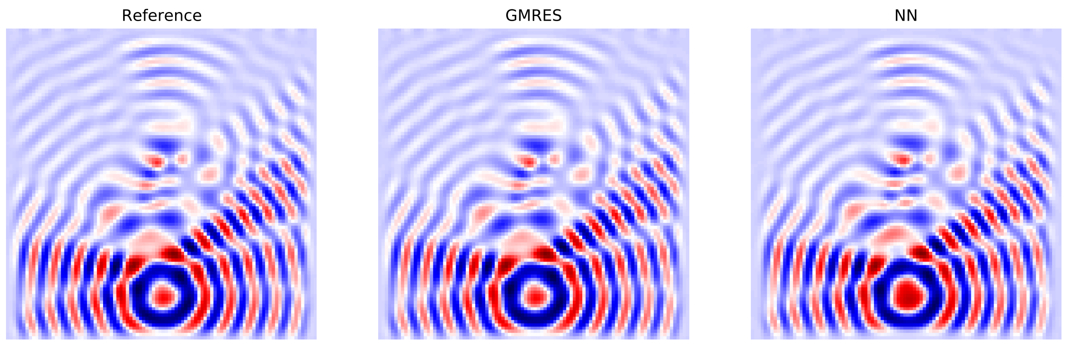

Figure 5.21: The first image shows the reference solution generated by k-Wave. The second image shows a solution optimized by GMRES, with residual $4 * 10^{-6}$. The solution needed 3500 iterations to achieve this residual ($\approx$ 175s). Our learned optimizer predicts the last image. The residual is $2.16 * 10^{-5}$ ($\approx$ 2.9s). The domain size of this figure is 64^3.

Chapter 6

Final Considerations

6.1 Conclusions

The main goal of this work was to propose a method that enables health officers to visualize the state of wavefield in the human brain during the application of ultrasound and to explore similar methods. Predicting the wavefield is generally a computationally intensive task, especially in 3D, where computational capability and memory often restrict the calculation.

The goal was finished, and we proposed, implemented and evaluated the novel method for predicting the wavefield after applying ultrasound in the steady-state. Our method showed promising experimental results compared to traditional approaches and showed a perspective view for its further development.

We demonstrated that the method works faster than traditional approaches, especially in larger domains and makes it feasible to predict wavefields in larger domains in 3D. A domain with size 416^3 performed more than eight times faster computation than k-Wave. Our work showed generalization to various shapes, ultrasound source locations, multiple sources and domain sizes. The tested domain sizes that worked well were between 64^3 to 496^3, which exceeded the maximal domain size that was commutable using on the machine in k-Wave.

The solution to PDE-like problems using neural networks brings advantages, not restricted only to the Helmholtz equation. This work also provided plenty of recommendations for future work. The most important of them are reminded in the following section.

6.2 Future work

Despite the work bringing positive results, there is still plenty of space for further development, experiments and research in this field of study. Besides the enormous potential applications for physics-informed neural networks in numerical problems, there are also many options to improve this research.

The implementation's current state is closer to the prototype and for demonstration purposes than for production. It is strongly recommended to rewrite the whole framework into the new one using a faster neural network library (e.g. *Jax*[1]). Another important recommendation is to apply more-than-memory training using gradient checkpointing. This would enable using the much higher amount of „virtual memory". It would be possible to perform experiments with different configurations and hyperparameters with higher mem-

[1]https://github.com/google/jax

ory limits. The training batch size in our work is four exclusively because of memory reasons. This, along with the replay buffer, should also be increased and find the optimal values for these parameters. We found out that the model can be tuned to converge faster, slower, longer etc. Finding the trade-off between the maximal possible accuracy and convergence speed should also be an objective of future work in this field.

Bibliography

[1] Abrahao, A., Meng, Y., Llinas, M., Huang, Y., Hamani, C. et al. First-in-human trial of blood–brain barrier opening in amyotrophic lateral sclerosis using MR-guided focused ultrasound. *Nature communications.* Nature Publishing Group. 2019, vol. 10, no. 1, p. 1–9.

[2] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D. et al. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems.* 2016, vol. 29.

[3] Axelsson, O. Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations. *Linear algebra and its applications.* Elsevier. 1980, vol. 29, p. 1–16.

[4] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J. et al. *Templates for the solution of linear systems: building blocks for iterative methods.* SIAM, 1994.

[5] Beisteiner, R. and Lozano, A. M. Transcranial ultrasound innovations ready for broad clinical application. *Advanced Science.* Wiley Online Library. 2020, vol. 7, no. 23, p. 2002026.

[6] Bellman, R. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America.* National Academy of Sciences. 1952, vol. 38, no. 8, p. 716.

[7] Bellman, R. Dynamic programming. *Science.* American Association for the Advancement of Science. 1966, vol. 153, no. 3731, p. 34–37.

[8] Berenger, J.-P. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of computational physics.* Elsevier. 1994, vol. 114, no. 2, p. 185–200.

[9] Bermúdez, A., Hervella Nieto, L., Prieto, A., Rodri, R. et al. An optimal perfectly matched layer with unbounded absorbing function for time-harmonic acoustic scattering problems. *Journal of computational Physics.* Elsevier. 2007, vol. 223, no. 2, p. 469–488.

[10] Blechschmidt, J. and Ernst, O. G. Three ways to solve partial differential equations with neural networks—A review. *GAMM-Mitteilungen.* Wiley Online Library. 2021, vol. 44, no. 2, p. e202100006.

[11] Bojarski, N. N. The k-space formulation of the scattering problem in the time domain. *The Journal of the Acoustical Society of America.* Acoustical Society of America. 1982, vol. 72, no. 2, p. 570–584.

[12] Boscaini, D., Masci, J., Rodolà, E. and Bronstein, M. Learning shape correspondence with anisotropic convolutional neural networks. *Advances in neural information processing systems.* 2016, vol. 29.

[13] Bouchoux, G., Bader, K. B., Korfhagen, J. J., Raymond, J. L., Shivashankar, R. et al. Experimental validation of a finite-difference model for the prediction of transcranial ultrasound fields based on CT images. *Physics in Medicine & Biology.* IOP Publishing. 2012, vol. 57, no. 23, p. 8005.

[14] Chai, T. and Draxler, R. R. Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific model development.* Copernicus GmbH. 2014, vol. 7, no. 3, p. 1247–1250.

[15] Chang, W. S., Jung, H. H., Zadicario, E., Rachmilevitch, I., Tlusty, T. et al. Factors associated with successful magnetic resonance-guided focused ultrasound treatment: efficiency of acoustic energy delivery through the skull. *Journal of neurosurgery.* American Association of Neurological Surgeons. 2016, vol. 124, no. 2, p. 411–416.

[16] Chen, T., Xu, B., Zhang, C. and Guestrin, C. Training deep nets with sublinear memory cost. *ArXiv preprint arXiv:1604.06174.* 2016.

[17] Chen, X., Wu, S. Z. and Hong, M. Understanding gradient clipping in private SGD: A geometric perspective. *Advances in Neural Information Processing Systems.* 2020, vol. 33, p. 13773–13782.

[18] Clement, G. T. and Hynynen, K. A non-invasive method for focusing ultrasound through the human skull. *Physics in Medicine & Biology.* IOP Publishing. 2002, vol. 47, no. 8, p. 1219.

[19] Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J. et al. Boosting adversarial attacks with momentum. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2018, p. 9185–9193.

[20] Elman, H. C. *Iterative methods for large, sparse, nonsymmetric systems of linear equations.* 1982. Dissertation. Yale University.

[21] Ernst, O. G. and Gander, M. J. Why it is difficult to solve Helmholtz problems with classical iterative methods. *Numerical analysis of multiscale problems.* Springer. 2012, p. 325–363.

[22] Gao, H., Sun, L. and Wang, J.-X. PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *Journal of Computational Physics.* Elsevier. 2021, vol. 428, p. 110079.

[23] Gimeno, L. A., Martin, E., Wright, O. and Treeby, B. E. Experimental assessment of skull aberration and transmission loss at 270 kHz for focused ultrasound stimulation of the primary visual cortex. In: IEEE. *2019 IEEE International Ultrasonics Symposium (IUS).* 2019, p. 556–559.

[24] Grieb, J., Barbero García, I. and Lerma, J. L. Spherical harmonics to quantify cranial asymmetry in deformational plagiocephaly. *Scientific reports*. Nature Publishing Group. 2022, vol. 12, no. 1, p. 1–10.

[25] Gumerov, N. A. and Duraiswami, R. A broadband fast multipole accelerated boundary element method for the three dimensional Helmholtz equation. *The Journal of the Acoustical Society of America*. Acoustical Society of America. 2009, vol. 125, no. 1, p. 191–205.

[26] He, J. and Xu, J. MgNet: A unified framework of multigrid and convolutional neural network. *Science china mathematics*. Springer. 2019, vol. 62, no. 7, p. 1331–1354.

[27] He, K., Zhang, X., Ren, S. and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE international conference on computer vision*. 2015, p. 1026–1034.

[28] Hersh, D. S. and Eisenberg, H. M. Current and future uses of transcranial focused ultrasound in neurosurgery. *Journal of neurosurgical sciences*. 2017, vol. 62, no. 2, p. 203–213.

[29] Hynynen, K. and Jolesz, F. A. Demonstration of potential noninvasive ultrasound brain therapy through an intact skull. *Ultrasound in medicine & biology*. Elsevier. 1998, vol. 24, no. 2, p. 275–283.

[30] Jaeger, H. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the „echo state network“ approach*. GMD-Forschungszentrum Informationstechnik Bonn, 2002.

[31] Jia, Z. A refined iterative algorithm based on the block Arnoldi process for large unsymmetric eigenproblems. *Linear algebra and its applications*. Elsevier. 1998, vol. 270, 1-3, p. 171–189.

[32] Jin, K. H., McCann, M. T., Froustey, E. and Unser, M. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*. IEEE. 2017, vol. 26, no. 9, p. 4509–4522.

[33] Johnson, S. G. Notes on FFT-based differentiation. In:. 2011.

[34] Johnson, S. G. Notes on perfectly matched layers (PMLs). *ArXiv preprint arXiv:2108.05348*. 2021.

[35] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G. et al. In-datacenter performance analysis of a tensor processing unit. In: *Proceedings of the 44th annual international symposium on computer architecture*. 2017, p. 1–12.

[36] Kandel, I. and Castelli, M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*. Elsevier. 2020, vol. 6, no. 4, p. 312–315.

[37] Kapturowski, S., Ostrovski, G., Quan, J., Munos, R. and Dabney, W. Recurrent experience replay in distributed reinforcement learning. In: *International conference on learning representations*. 2018.

[38] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ArXiv preprint arXiv:1412.6980.* 2014.

[39] Krishna, V., Sammartino, F. and Rezai, A. A review of the current therapies, challenges, and future directions of transcranial focused ultrasound technology: advances in diagnosis and treatment. *JAMA neurology.* American Medical Association. 2018, vol. 75, no. 2, p. 246–254.

[40] Kumar, S. K. On weight initialization in deep neural networks. *ArXiv preprint arXiv:1704.08863.* 2017.

[41] Legon, W., Sato, T. F., Opitz, A., Mueller, J., Barbour, A. et al. Transcranial focused ultrasound modulates the activity of primary somatosensory cortex in humans. *Nature neuroscience.* Nature Publishing Group. 2014, vol. 17, no. 2, p. 322–329.

[42] Lu, Y., Zhong, A., Li, Q. and Dong, B. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In: PMLR. *International Conference on Machine Learning.* 2018, p. 3276–3285.

[43] Mao, Z., Jagtap, A. D. and Karniadakis, G. E. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering.* Elsevier. 2020, vol. 360, p. 112789.

[44] McDannold, N., Clement, G. T., Black, P., Jolesz, F. and Hynynen, K. Transcranial magnetic resonance imaging–guided focused ultrasound surgery of brain tumors: initial findings in 3 patients. *Neurosurgery.* Oxford University Press. 2010, vol. 66, no. 2, p. 323–332.

[45] Meng, X. and Karniadakis, G. E. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. *Journal of Computational Physics.* Elsevier. 2020, vol. 401, p. 109020.

[46] Nabavizadeh, A., Song, P., Chen, S., Greenleaf, J. F. and Urban, M. W. Multi-source and multi-directional shear wave generation with intersecting steered ultrasound push beams. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control.* IEEE. 2015, vol. 62, no. 4, p. 647–662.

[47] Pinton, G., Aubry, J.-F., Fink, M. and Tanter, M. Effects of nonlinear ultrasound propagation on high intensity brain therapy. *Medical physics.* Wiley Online Library. 2011, vol. 38, no. 3, p. 1207–1216.

[48] Powell, W. B. What you should know about approximate dynamic programming. *Naval Research Logistics (NRL).* Wiley Online Library. 2009, vol. 56, no. 3, p. 239–249.

[49] Pulkkinen, A., Werner, B., Martin, E. and Hynynen, K. Numerical simulations of clinical focused ultrasound functional neurosurgery. *Physics in Medicine & Biology.* IOP Publishing. 2014, vol. 59, no. 7, p. 1679.

[50] Putzky, P. and Welling, M. Recurrent inference machines for solving inverse problems. *ArXiv preprint arXiv:1706.04008.* 2017.

[51] Raissi, M., Perdikaris, P. and Karniadakis, G. E. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *ArXiv preprint arXiv:1711.10561*. 2017.

[52] Raissi, M., Yazdani, A. and Karniadakis, G. E. Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data. *ArXiv preprint arXiv:1808.04327*. 2018.

[53] Raymond, S. J. and Camarillo, D. B. Applying physics-based loss functions to neural networks for improved generalizability in mechanics problems. *ArXiv preprint arXiv:2105.00075*. 2021.

[54] Read, W. Analytical solutions for a helmholtz equation with dirichlet boundary conditions and arbitrary boundaries. *Mathematical and computer modelling*. Elsevier. 1996, vol. 24, no. 2, p. 23–34.

[55] Rizzuti, G., Siahkoohi, A. and Herrmann, F. J. Learned iterative solvers for the Helmholtz equation. In: European Association of Geoscientists & Engineers. *81st EAGE Conference and Exhibition 2019*. 2019, vol. 2019, no. 1, p. 1–5.

[56] Robertson, J., Martin, E., Cox, B. and Treeby, B. E. Sensitivity of simulated transcranial ultrasound fields to acoustic medium property maps. *Physics in Medicine & Biology*. IOP Publishing. 2017, vol. 62, no. 7, p. 2559.

[57] Ronneberger, O., Fischer, P. and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In: Springer. *International Conference on Medical image computing and computer-assisted intervention*. 2015, p. 234–241.

[58] Rosnitskiy, P. B., Yuldashev, P. V., Sapozhnikov, O. A., Gavrilov, L. R. and Khokhlova, V. A. Simulation of nonlinear trans-skull focusing and formation of shocks in brain using a fully populated ultrasound array with aberration correction. *The Journal of the Acoustical Society of America*. Acoustical Society of America. 2019, vol. 146, no. 3, p. 1786–1798.

[59] Saad, Y. and Schultz, M. H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*. SIAM. 1986, vol. 7, no. 3, p. 856–869.

[60] Shah, B. R., Lehman, V. T., Kaufmann, T. J., Blezek, D., Waugh, J. et al. Advanced MRI techniques for transcranial high intensity focused ultrasound targeting. *Brain*. Oxford University Press. 2020, vol. 143, no. 9, p. 2664–2672.

[61] Song, P., Urban, M. W., Manduca, A., Zhao, H., Greenleaf, J. F. et al. Comb-push ultrasound shear elastography (CUSE) with various ultrasound push beams. *IEEE transactions on medical imaging*. IEEE. 2013, vol. 32, no. 8, p. 1435–1447.

[62] Song, P., Zhao, H., Manduca, A., Urban, M. W., Greenleaf, J. F. et al. Comb-push ultrasound shear elastography (CUSE): a novel method for two-dimensional shear elasticity imaging of soft tissues. *IEEE transactions on medical imaging*. IEEE. 2012, vol. 31, no. 9, p. 1821–1832.

[63] Stanziola, A., Arridge, S. R., Cox, B. T. and Treeby, B. E. A Helmholtz equation solver using unsupervised learning: Application to transcranial ultrasound. *Journal of Computational Physics.* Elsevier. 2021, vol. 441, p. 110430.

[64] Sun, J. and Hynynen, K. Focusing of therapeutic ultrasound through a human skull: a numerical study. *The Journal of the Acoustical Society of America.* Acoustical Society of America. 1998, vol. 104, no. 3, p. 1705–1715.

[65] Taflove, A., Hagness, S. C. and Piket May, M. Computational electromagnetics: the finite-difference time-domain method. *The Electrical Engineering Handbook.* Elsevier Amsterdam, The Netherlands. 2005, vol. 3.

[66] Tallec, C. and Ollivier, Y. Unbiasing truncated backpropagation through time. *ArXiv preprint arXiv:1705.08209.* 2017.

[67] Thuerey, N., Holl, P., Mueller, M., Schnell, P., Trost, F. et al. Physics-based Deep Learning. *ArXiv preprint arXiv:2109.05237.* 2021.

[68] Treeby, B. E. and Cox, B. T. K-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields. *Journal of biomedical optics.* International Society for Optics and Photonics. 2010, vol. 15, no. 2, p. 021314.

[69] Treeby, B. E., Jaros, J., Rendell, A. P. and Cox, B. Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using ak-space pseudospectral method. *The Journal of the Acoustical Society of America.* Acoustical Society of America. 2012, vol. 131, no. 6, p. 4324–4336.

[70] Wang, S., Maarten, V. and Xia, J. Acoustic inverse scattering via Helmholtz operator factorization and optimization. *Journal of Computational Physics.* Elsevier. 2010, vol. 229, no. 22, p. 8445–8462.

[71] White, P. J., Clement, G. T. and Hynynen, K. Longitudinal and shear mode ultrasound propagation in human skull bone. *Ultrasound in medicine & biology.* Elsevier. 2006, vol. 32, no. 7, p. 1085–1096.

[72] Yang, L., Zhang, D. and Karniadakis, G. E. Physics-informed generative adversarial networks for stochastic differential equations. *SIAM Journal on Scientific Computing.* SIAM. 2020, vol. 42, no. 1, p. A292–A317.

[73] Young, D. M. and Jea, K. C. Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods. *Linear Algebra and its applications.* Elsevier. 1980, vol. 34, p. 159–194.

[74] Yu, B. et al. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics.* Springer. 2018, vol. 6, no. 1, p. 1–12.

[75] Zhu, Y., Zabaras, N., Koutsourelakis, P.-S. and Perdikaris, P. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics.* Elsevier. 2019, vol. 394, p. 56–81.

[76] Zienkiewicz, O., Kelly, D. and Bettess, P. The Sommerfeld (radiation) condition on infinite domains and its modelling in numerical procedures. In: *Computing Methods in Applied Sciences and Engineering, 1977, I.* Springer, 1979, p. 169–203.

Appendix A

Running the solution

The main functionality implemented in this thesis is training the model and performing inference. It is required to install the Python libraries defined in `requirements.txt` and strongly recommended use multiple GPUs for the training and ideally also for the inference.

A.1 Training

Before running the training is possible to set the parameters in the file `hparams.json`. The file contains the training hyperparameters and the configuration like available GPUs or path to the training data. For the start of the training, run the following command:

```
python3 train.py
```

A.2 Inference

The only parameters that need to be set for the inference are the available GPU, number of iterations, source positions and path to the target file in the format `.npy`. They must be set in the file `hparams.json`. After setting parameters user runs the following command:

```
python3 predict.py
```