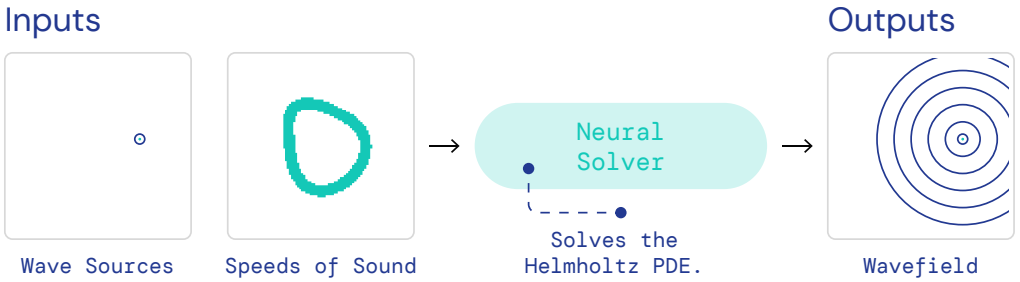
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2022
http://excel.fit.vutbr.cz

Approximation of Sound Propagation by Neural Networks

Son Hai Nguyen*



Abstract
Neural solvers have been increasingly explored with the aim of replacing computationally expensive conventional numerical methods for solving PDEs. This work focuses on solving the time-independent Helmholtz equation for transcranial ultrasound therapy. Most of the popular methods for modeling physics systems are based on U-net. However, convolutional neural networks require the data to be sampled on a regular grid. In order to try to lift this restriction, we propose an iterative solver based on graph neural networks. Unlike Physics-informed neural networks, our model needs to be trained only once, and only a forward pass is required to obtain a new solution given input parameters. The model is trained using supervised learning, where the reference results are computed using the traditional solver k-Wave. Our results show the model's unroll stability despite being trained with only 8 unroll iterations. Despite the model being trained on the data with a single source, it can predict wavefields with multiple sources and generalize to much larger computational domains. Our model can produce a prediction for sub-pixel points with higher accuracy than linear interpolation.

Keywords: Neural solver — Helmholtz equation — Graph neural network — Transcranial ultrasound — PDE

Supplementary Material:

*xnguye16@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The transcranial ultrasound therapy has been recently approved by Food and Drug Administration (FDA) for essential tremor, and Parkinson's disease treatment [31]. Currently, there are experimental treatments using ultrasound to create microbubbles, which enable targeted drug delivery for Alzheimer's disease and brain tumors treatment [16].

The ultrasound transmitter positioning requires a simulation of the ultrasound propagation through a skull, which depends on solving the Helmholtz equation. The waves have to be focused in the area of interest in order to make thermal ablation, and other treatments [12] possible. The simulation using traditional solvers requires a lot of computational resources. With a focus on reducing the simulation time, approximators based on neural networks have been proposed — neural solvers.

Most neural physical systems adopt *Convolutional*

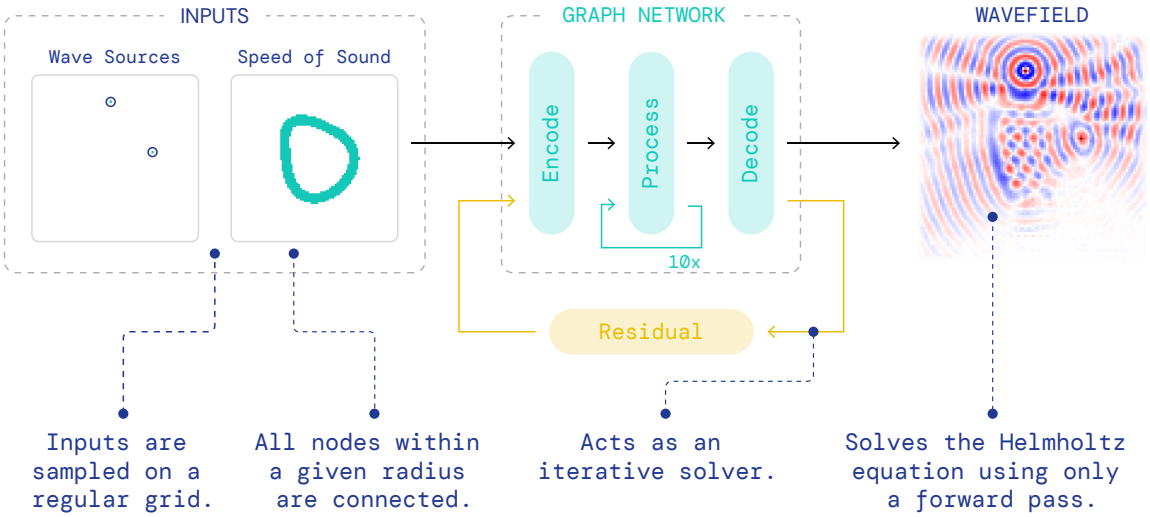


Figure 1. Proposed system is based on a graph neural network (GNN) used as an iterative solver. Given the speed of sound, wave sources, and residual (Equation (10)) of a wavefield prediction from the previous iteration, the model outputs a refined prediction of the unknown wavefield. Unlike PINN methods, our model does not require to be trained for each new set of PDE parameters.

Neural Networks (CNN) [34, 36] or *Physics-informed neural networks* (PINN) [26, 17, 7, 32, 10]. The first family of methods works only with data sampled on a regular grid. On the other hand, the latter group of methods is mesh-independent. However, the networks need to be retrained for each new set of PDE parameters.

Our method tackles these problems by employing *Graph Neural Networks* (GNN) [30, 11], allowing more variable mesh than CNNs. We use graph edges to control the message passing between the nodes, which correspond to the samples in a space. The network is structured as an iterative solver [27]. Only a forward pass of the model is required to solve a Helmholtz equation for a new set of parameters.

and is able to solve the Helmholtz equation for a new set of parameters using nothing but a forward pass.

1.1 Governing Equations

This work focuses on equations used in sound-waves modeling. Wave equation describes the propagation of waves through space:

$$\frac{\partial^2 u(\boldsymbol{x}, t)}{\partial t^2} = c^2 \nabla^2 u(t, \boldsymbol{x}), \tag{1}$$

where $u : \mathbb{R}^n \to \mathbb{C}$ is a wavefield, $n \in \mathbb{Z}^+$ denotes the dimensionality, t, $c \in \mathbb{R}^+$ is a speed of sound (SOS), and $\boldsymbol{x} \in \mathbb{R}^n$ stand for time variable and spatial coordinate, respectively.

However, in transcranial focused ultrasound therapies [16, 12, 31], the focused ultrasound beam is applied for a period of time exceeding the time required to reach a steady state. Consequently, the time-independent Helmholtz equation is used in this work to model the propagation of waves (Figure 2).

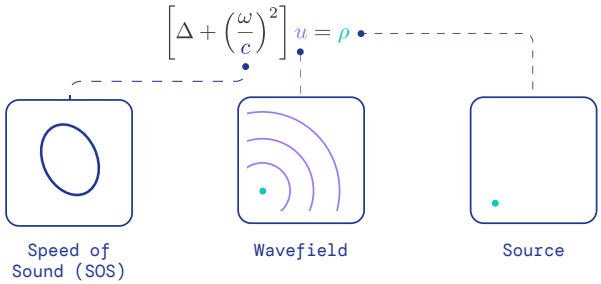


Figure 2. In this work, a speeds of sound c contains only a model of a skull, which has a different speed of sound than the surrounding medium. Source values ρ are zero, except at the location of a source, where the value is set to the source amplitude. *Inverse problem* is not contemplated in this work. Thus the wavefield u is the only unknown in the equation.

Helmholtz equation The Helmholtz equation is a result of the separation of variables in the Wave equation (Equation (1)). Although a simulation of ultrasound propagating through the skull involves nonlinear, shear, and other effects, Stanziola et al. [34] present that these effects can be added after computing the wavefield modeled by Helmholtz equation. Thus, we focus on modeling a simplified model of wave propagation described by the Helmholtz equation subjected to the Sommerfeld radiation condition at infinity [33]:

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

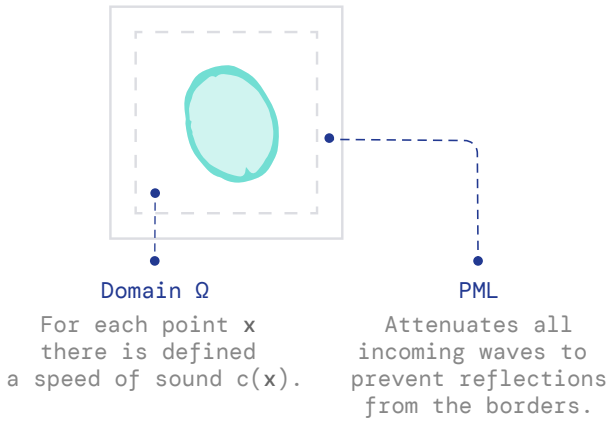


Figure 3. To satisfy Sommerfield condition (Equation (3)), an artificial layer (PML) is wrapped around the domain Ω to attenuate all incoming waves. Adapted from Stanziola et al. [34].

$$\text{s.t.} \lim_{|\boldsymbol{x}|\to\infty} |\boldsymbol{x}|^{\frac{n-1}{2}} \left(\frac{\partial}{\partial |\boldsymbol{x}|} - i\frac{\omega}{c_0} \right) u(\boldsymbol{x}) = 0, \quad (3)$$

where $n \in \mathbb{Z}^+$ is the number of dimensions, $\omega \in \mathbb{R}$ stands for the angular velocity of the source, $\boldsymbol{x} \in \mathbb{R}^n$ is a spatial coordinate, $c : \mathbb{R}^n \to \mathbb{R}^+$ is the inhomogeneous speed of sound (SOS) at a certain given point $\boldsymbol{x}$, $\rho : \mathbb{R}^n \to \mathbb{C}$ denotes the forcing term, and $u : \mathbb{R}^n \to \mathbb{C}$ is the **unknown** acoustic wavefield. The inhomogeneous SOS is considered only within a domain of interest Ω, for the rest of the domain $\partial\Omega$ the speed of sound is homogeneous with a value equal to c_0. Graphical example of the components of the Helmholtz equation are shown in Figure 2.

Perfectly Matched Layer In order to satisfy the Sommerfield radiation condition in a finite domain Ω, a boundary condition such as *Perfectly Matched Layer* (PML) [4] can be applied. PML is an artificial layer surrounding the domain Ω attenuating all incoming waves to prevent reflections from the domain's border. Consequently, PML simulates an infinite domain, as shown in Figure 3.

By employing a PML, an absorption term is introduced into the derivative operators:

$$\frac{\partial}{\partial \eta} \to \frac{1}{\gamma_\eta} \frac{\partial}{\partial \eta}, \quad (4)$$

where $\eta = x_1, x_2, \ldots, x_n$, x_j corresponds to the j-th spatial dimension, and where

$$\gamma_\eta = \begin{cases} 1, & \eta \in \Omega \\ 1 + \dfrac{i}{\omega}\sigma(\eta), & \text{otherwise} \end{cases} \quad (5)$$

where ω is the angular velocity from Equation (2), Ω is the domain of interest, and the absorption profile σ is defined by the following equation:

$$\sigma(\eta) = \sigma_{\max} \left(1 - \frac{\hat{\eta}}{\Delta L} \right)^2, \quad (6)$$

where $\sigma_{\max}$ is maximum PML absorption parameter, ΔL defines the width of a PML and $\hat{\eta}$ is the spatial distance from the domain border in a give dimension.

2. Related Works

In recent years, more emphasis has been put on researching neural operators. The main objective of these approximate solvers is to estimate the solution of a given PDE faster or lift restrictions created by conventional numerical solvers.

We consider PDEs of the following form:

$$\begin{aligned} (\mathcal{L}_a u)(\boldsymbol{x}) &= f, \quad x \in \Omega \\ u(\boldsymbol{x}) &= 0, \quad x \in \partial\Omega \end{aligned} \quad (7)$$

where a represents parameters of a differential operator $\mathcal{L}$, and u, f are functions on the spatial domain. f represents a forcing term, whereas u is the solution in the forward simulation.

Although *k-Wave* [37] belongs to the conventional solver family, it uses k-space pseudospectral method [5, 6, 22, 35], which allows sparser discretization than FEM [13] and FDM [23]. Unlike traditional solvers, neural solvers utilizes a neural network. In recent years, CNNs [34, 36] has become one of the most common approaches to model physical systems. However, these systems suffer by requiring input data to be sampled on a regular grid. Nonetheless, CNNs yield impressive results.

$$R = \mathcal{L}_a u - f \quad (8)$$

Authors of *Helmnet* [34] pursue the identical objective as our work — solving the Helmholtz equation (Equation (2)). Helmnet is structured as an iterative solver based on U-net [27], instead of passing the PDE parameters directly, the authors discovered that incorporating these parameters into the residual term R (Equation (8)) eases the network learning the mapping from the parameters of a PDE a to a solution u. Alternatively to the other approaches [36], Helmnet is trained using only a physics loss.

Recently, there has been growing interest in mesh-independent methods. Various *Physics-informed neural network* (PINN) methods [10, 32, 7, 26, 17] use *multilayer perceptron* (MLP) to map a spatial coordinate $\boldsymbol{x}$ directly to solution $u(\boldsymbol{x})$. In contrast with previously mentioned methods, PINN methods learn to represent the solution instead of computing one. The

networks need to be trained again for each new set of PDE parameters a. Nevertheless, querying a new position $\boldsymbol{x}$ requires only a forward pass of a model, making it mesh-independent. PINN methods utilize an automatic differentiation to compute the spatial derivatives used in a physics loss. Thus, PINN methods enforce strong form of the governing PDE.

Every physics system can be modeled using a graph, as samples produced by spatial sampling can be represented as nodes in a graph. Sanchez-Gonzalez et al. [28] connect nodes in a certain radius, whereas Pfaff et al. [24] operates directly with edges produced by mesher. These two methods networks [24, 28] are based on the *Encode – Process – Decode* architecture defined under the *Graph Network* framework [3]. However, the mentioned methods predict only the time evolution of time-dependent PDEs. Methods such as Li et al. [18], Anandkumar et al. [2] have shown a capability to solve time-independent PDEs. *Graph Kernel Network* (GKN) [2] learns the Green's function of a given PDE directly by encoding it in the model's weights. Unfortunately, the number of edges grows quadratically with the number of nodes. *Fourier Neural Operator* (FNO) [18] extends GKN by replacing the integral kernel with convolution in the Fourier space. A layer transforming data in the Fourier space and transforming it back to the physical space is called *Fourier Layer*.

3. Proposed Solution

To solve the Helmholtz equation, we propose an iterative solver defined by the following equation:

$$u_{k+1} = f_\theta(v, e, u_k, R_k), \tag{9}$$

where u_k and R_k are wavefield prediction and residual at k-th iteration, f_θ, v and e denote neural network, node features and edge features, respectively. The neural network f_θ is described more extensively in Section 3.2.

Unlike Helmnet [34], our solution employs supervised learning, thus a reference solution $u^*(\boldsymbol{x})$ is required. The following section provides a more detailed description the used dataset.

3.1 Skull Dataset

We use the method proposed by Stanziola et al. [34] for synthetic skulls as sound speed maps. The shape of the idealized skulls is created by summing circular harmonics with random amplitude and shape. Since the whole work considers normalized units $\omega = 1\,\mathrm{rad/s}$ and background speed of sound of $1\,\mathrm{m/s}$, the skull thickeness ranges from 2 to $10\,\mathrm{m}$ with sound speed varying from 1.5 to $2\,\mathrm{m/s}$. We set the grid size to 384×384 with $0.25\,\mathrm{m}$ grid spacing to allow later downsampling if needed

We employed k-Wave [37] to generate ground-truth solutions for each sample. For each skull, three random wave source positions are generated. Based on the linearity of the Helmholtz equation, solutions can be summed in order to create a new, more complex solution with multiple sources (Figure 5). Training, validation, and test sets contain 24000, 3000, and 3000 wavefields samples, respectively. The dataset is available on a public repository[1].

3.2 Model Architecture

Figure 4 depicts used neural network, which is based on Graph Network [3] with Encode – Process – Decode structure [24, 28].

All parts of the process block ϕ^e, ϕ^v, both encoders ϵ^v, ϵ^e and decoder δ^v are implemented using a two-layer MLP with ReLU activation function [1] and a residual connection. The latent and output size is 64, except δ^v, whereas the output size equals to the model's output size. The MLP is depicted in Figure 6.

All parameters from the Helmholtz equation, such as SOS map, source map, or spatial coordinate, are used as an input along with the prediction and its residual from a previous iteration. All nodes are sampled on a uniformly spaced grid, where all nodes within a radius $r = 0.02$ are connected.

Encode Encoder consists of two separate MLPs ϵ^v, ϵ^e for nodes and edges, respectively. These encoders transform input features (procesess each node and each edge separately) into latent vector of size 64. Each node feature v_i with a spatial position $\boldsymbol{x}_i$ is composed of SOS map$c(\boldsymbol{x}_i)$, wave source distribution $\rho(\boldsymbol{x}_i)$ and PML absorption term $\sigma(\boldsymbol{x}_i)$. The distance information $|x_i j|$ and $x_i - x_j$ between connected nodes is encoded in the edges.

The SOS map is sampled using 96×96 regular grid. Thus the data is downsampled using a factor of 4. Connections between nodes are created using radius $r = 0.02$, node coordinates are normalized to range $[0, 1]$.

Additionally, we use connections between every n-th node (hop connection), increasing the model's receptive field (Figure 7). The hop connections can be seen as a multi-resolution graph [19, 25] and should enable the network to predict larger area in less unroll iterations. Our model contains hop connections between every 3-th, 5-th and 10-th node.

[1] https://sc-nas.fit.vutbr.cz:10443/xnguye16/ssw-dataset

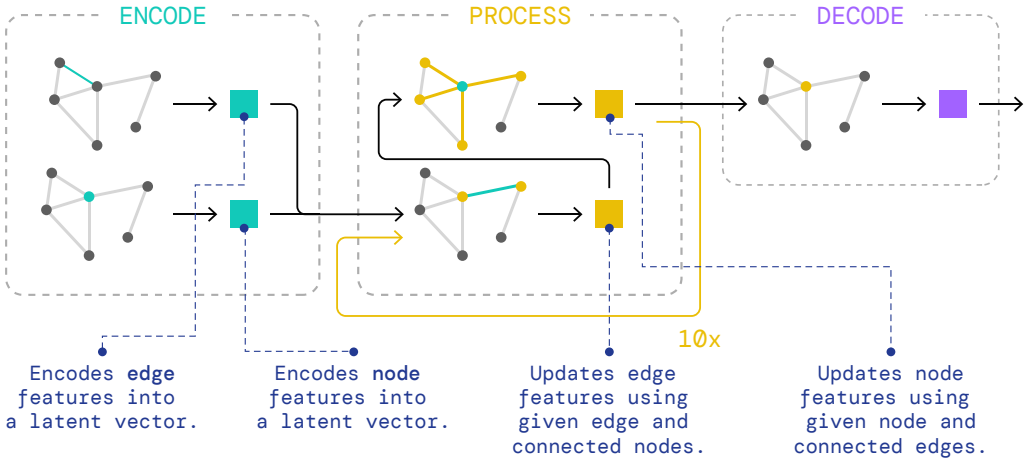


Figure 4. The network architecture consists of by three parts — encode, process, and decode. In the encoder stage, all features are respectively transformed into latent vectors. A message-passing mechanism is executed in the process stage, where information from neighboring nodes is aggregated to produce updated features for the target node and edges. Lastly, the decoder transforms data from latent space to output space.

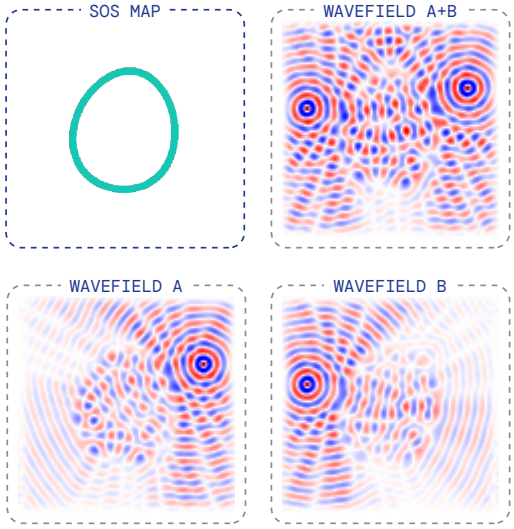


Figure 5. Synthetic skulls were generated by summing several elliptical harmonics [34]. Reference wavefields were computed using k-Wave [37]. For each SOS map, three wavefields were computed, each with a different source location. Due to the linearity of the Helmholtz equation, wavefields can be added together to produce a new, more complex wavefield.

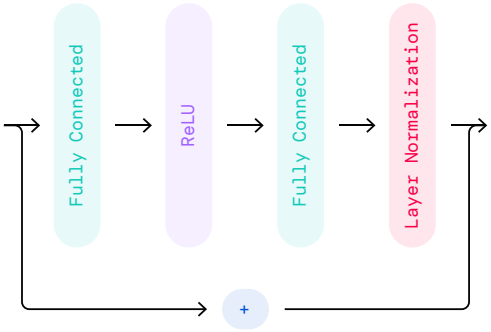


Figure 6. The main building block for the whole network is based on two-layer MLP with a ReLU activation function and a residual connection.

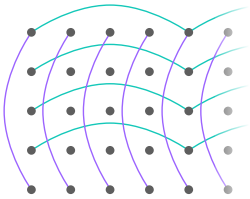


Figure 7. Hop connections connects every n-th node, in this particular case $n = 4$. Hop connections $n \in \{3, 5, 10\}$ are used in the proposed solution.

Process Processor unit is a derivation of a processor defined by Pfaff et al. [24], it consists of P identical blocks — Graph Block [3]. Each block contains a separate set of weights.

As depicted in Figure 8, the Graph Block consists of three parts: ϕ^e, ϕ^v and $\rho^{e\to v}$. Function ϕ^e is implemented using an MLP. It encodes data of an edge and nodes connected to it into a new edge feature. Then edges connected to a node are aggregated using function $\rho^{e\to v}$. In our work, we use the mean function as an aggregation function. Aggregated features are transformed using a function ϕ^v into an updated version of node features. The previously mentioned function ϕ^v is implemented using MLP as well. The described functionality is also referred to as *message passing* [10]. One Graph Block corresponds to one message passing. More Graph Blocks result in a larger receptive field of the network.

Decode Last stage of the network δ^v decodes node features from latent space into the scaled wavefield prediction βu_{k+1}, where $\beta = 500$ is a scaling term (see Section 3.4).

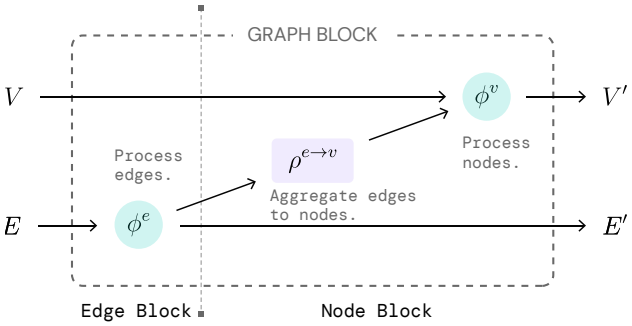


Figure 8. Graph block consists of two parts: edge block and node block. Edge block uses features of a given edge a its nodes to update edge's features. Node block updates a node features based on all nodes connected to the target node. Functions ϕ^e, ϕ^v are implemented using MLPs, aggregation function $\rho^{e\to v}$ is using the *mean* function in our work.

3.3 Residual Calculation

Due to explicit residual calculation, an approximation of the Laplacian has to be computed. After Equation (2) is plugged in Equation (8) the residual is described as follows:

$$R = \left[\nabla^2 + \left(\frac{\omega}{c(\boldsymbol{x})}\right)^2\right] u(\boldsymbol{x}) - \rho(\boldsymbol{x}), \qquad (10)$$

Based on the conducted experiments (Section 4.2), we use pseudo-spectral method to approximate the derivatives, which is restricted to the regular grid.

FFT-based Stanziola et al. [34] approximates first-order derivatives using FFT-based derivative (Equation (11)), which are then composed into the Laplacian. The Laplacian can not be computed directly as a result of using PML as the boundary condition (Equation (4)).

$$\frac{\mathrm{d}}{\mathrm{d}\eta} f(\eta) \approx \mathcal{F}^{-1}\left(\mathcal{F}\left(ik_\eta f(\eta)\right)\right), \qquad (11)$$

where $\eta \in \{x_1, x_2, \ldots, x_n\}$ denotes a spatial dimension, k_η represents wavenumbers in a given direction, $\mathcal{F}$, $\mathcal{F}^{-1}$ are Fourier transform and its inverse, respectively.

Average Gradient on Star Unlike the FFT-base method, the Average Gradient on Star (AGS) and Per-Cell linear Estimation (PCE) methods can be utilized on an irregular grid. In a 2D mesh, for a triangle t with vertices v_i, v_j, v_k, PCE is defined as follows [21]:

$$\nabla f_t \approx (f_j - f_i)\frac{(v_i - v_k)^\perp}{2A_t} + (f_k - f_i)\frac{(v_j - v_i)^\perp}{2A_t}, \qquad (12)$$

where A_t is an area of the triangle t, f_i is a value of a vertex v_i and $(e)^\perp$ denotes a perpendicular vector e to the vector e.

To compute per-vertex gradients, *Average Gradient on Star* (AGS) averages all gradients from neighboring vertices:

$$\nabla f(v) \approx \frac{1}{\sum_{i\in\mathcal{N}(v)} A_i} \sum_{i\in\mathcal{N}(v)} A_i \nabla_{PCE} f_i \qquad (13)$$

where $\nabla_{PCE} f_i$ is defined by Equation (12) and $\mathcal{N}(v)$ is set of vertices connected to the vertex v.

3.4 Model Training

As mentioned before, the model is trained using **supervised** learning. Any addition of physics terms resulted in unstable training and constraining the model's ability to learn more than only a small neighborhood around the source.

Loss function Naturally, mean-squared error is applied as a loss function

$$L = \frac{1}{N} \sum ||\hat{u}_T - \beta u^*||_2^2, \qquad (14)$$

where $\beta = 500$ is a scaling term, N is the number of graph samples, $\hat{u}_T$ is the predicted solution after T unroll iterations. In other words, loss function takes into account only a prediction from the last unroll iteration.

todobug no indent The value of scaling term β was selected empirically. It can be seen as scaling the source amplitude by β. Usage of the scaling term, changes the magnitude as well as direction of loss function gradients. Nonetheless, we are not certain, why the scaling term is essential for successful training.

Training phases Although the model can be trained end-to-end, the two-phased approach requires only half of the computational time (Table 1). The first phase involves training the network with 3 unroll iterations for $\approx$ 70k optimization iterations. As illustrated in Figure 9, the network is then fine-tuned for $\approx$ 10k optimization steps with 8 unroll iterations. Both phases are trained using the Adam optimizer [15] with a learning rate $\alpha = 3e-5$. The network is trained on 8 A100 40GB GPUs using Pytorch Lightning[2] distributed data parallel (DDP) accelerator. Despite the fact that the batch size is set to 2, the effective batch size is 16 due to gradient averaging across 8 GPUs.

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

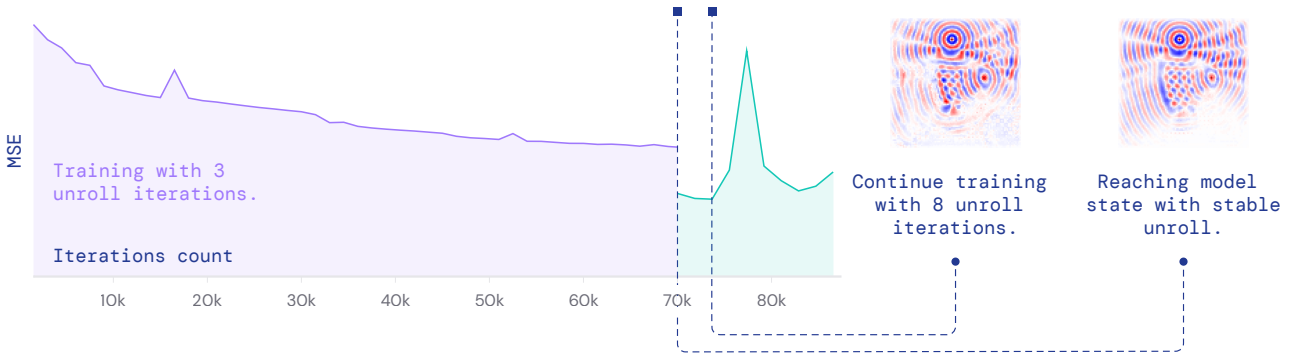


Figure 9. Even though end-to-end training requires the same number of optimization steps as the two-phased approach, it requires more time since the duration of one optimization step is longer. Thus we opted for training with two phases. The first part of training comprises 70k optimization iterations with three unroll iterations. Afterward, the network is fine-tuned for less than 10k optimization iterations with 8 unroll iterations. Here the network learns to stabilize unrolling. The displayed error curve was calculated on the validation set. Thus, peak in the second phase of the training signalizes overfitting.

Table 1. Two phased training significantly reduces the used computational resources compared to the end-to-end approach. The training time was measured on a computer with 8 A100 40GB.

	End-to-end	**2 phases**
MSE	14.301	14.577
Duration [h]	21	10
Optim. steps	≈ 80k	

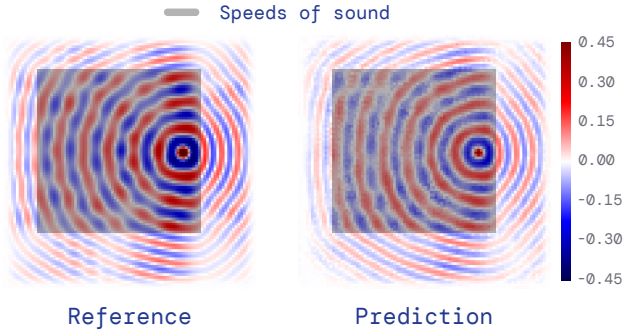


Figure 10. Although the model was trained only on synthetic skulls, it is able to predict wavefield even with the square heterogenity in the middle.

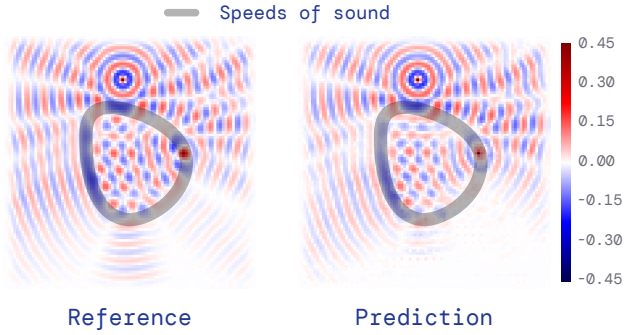


Figure 11. The model is able to predict wavefields even with multiple sources, despite it was trained with single-source samples.

4. Results

To evaluate the performance of the trained model, we put our solution under multiple tests. Firstly, the model's ability to generalize is demonstrated using samples outside of the training and validation distribution, including a more than 5-times larger domain. The reference solutions are computed using k-Wave Toolbox [37]. Additionally, our model is compared to Helmnet [34], where the MSE is used as an evaluation metric:

$$\text{MSE} = \frac{1}{N}||\hat{u} - u^*||_2^2, \tag{15}$$

where N is the nodes count, $\hat{u}$ and u^* are predicted and reference wavefield, respectively.

4.1 Generalization

Since our model has been trained only on synthetic skulls, a square SOS map is out of the training distribution. As depicted in Figure 10, our model is able to predict SOS maps, which it has never seen, indicating that the model learned to solve the Helmholtz equation.

Furthermore, Figure 11 depicts that the model learned the interaction between waves from multiple sources, although it was trained on samples with a single source.

Unroll stability For most samples with 96×96 grid size, 8 unroll iterations are sufficient to predict the wavefield for the whole computational domain. Nonetheless, more complex wavefields require more than 8 unroll iterations due to more wave reflections that need to be simulated. However, a larger domain requires more unroll iterations. To test whether our model can predict wavefields even in larger domains, we measured unroll stability for 128 iterations in a $96\times$

96 domain. As depicted in Figure 12, the error does not rapidly diverge until approximately 70-th iteration. To reduce the unroll divergence, a replay buffer [14] is utilized. The replay buffer contains 400 quaternions (u_k, R_k, e, v). The iteration index k is randomly initialized to an integer from 0 to the half of the maximum iterations count Q — we set $Q = 256$. After training the model with the replay buffer, the error does not diverge (Figure 12). Thus, the model learned to do "nothing", when the wavefield is solved.

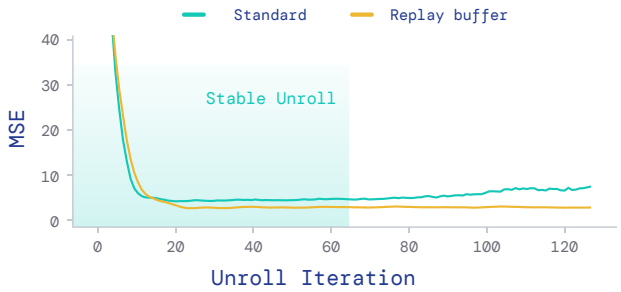


Figure 12. Although, the model is trained with only 8 unroll iterations, the error does not greatly diverge until 70-th iteration. To mitigate the unroll divergence, we use a replay buffer. The unroll stability was tested in a 96×96 domain.

Larger domains Since the model depends on the nodes coordinates $x_i \in [0, 1]$ (see Section 3.2), it is not invariant to the domain size. To create a size-invariant model, we first train the model during the first phase as described in Section 3.4. During the second phase, the input features corresponding to the coordinates of the nodes are set to zero. With that modification, we are able to train a size-invariant model, as shown in Figure 13, where the model predicts the wavefield for a 512×512 domain.

4.2 Derivative Approximators

The FFT-based approximation of the residual is more accurate than the AGS approximation. Although the AGS is able to approximate the shape of the gradient well, the magnitude error is significantly larger. The proposed model is sensitive to directional and magnitude error in the Laplacian due to calculating the residual explicitly. As illustrated in Figure 14, with more significant residual error, the model can predict the wavefields only to a certain distance from the source.

4.3 Irregular Grids

To alleviate the constraint of using data sampled on a regular grid, we experimented with data sampled on different grids (see Figure 15). As experiments in Section 4.2 show, using derivative approximators other than the FFT-based is not feasible. Thus, we opt for

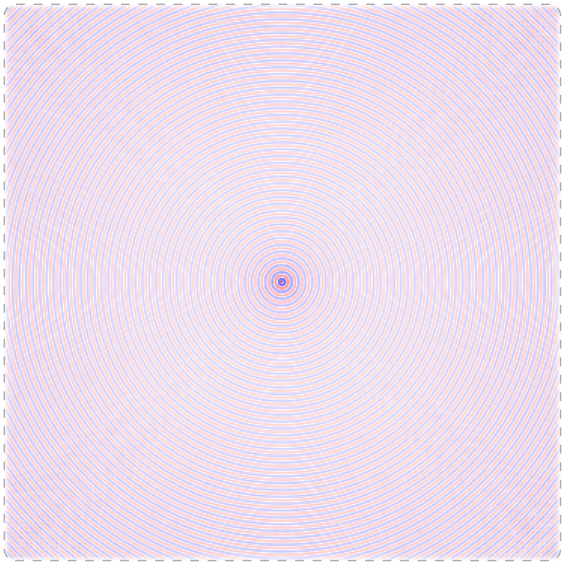
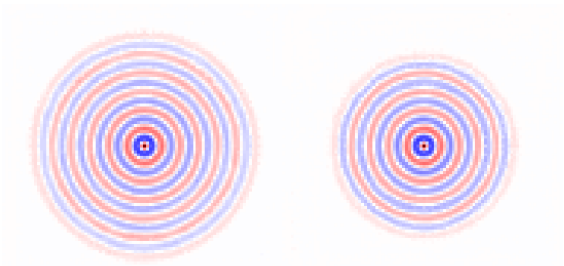


Figure 13. Our model is able to perform inference even on larger domains 512×512. However, during fine-tuning, the position of each node (used as an input feature) is set to zero. If the position is to zero before the first phase of training, the model is not able to converge to a state of stable unrolling.



(a) FFT-based derivative **(b)** AGS

Figure 14. Despite having only three unroll iterations, the model with the FFT-based derivative approximation, is able to predict significantly larger area than with the AGS approximation. When using AGS, the predicted area does not grow noticeable with more unroll iterations.

utilizing FFT for computing the Laplacian. Although the Fourier transform is defined on graphs [29], these methods are more computationally expensive than the regular FFT, which makes them impractical.

In order to use FFT with irregular grids, we utilize a linear interpolator. Irregularly sampled data are interpolated on a regular grid to compute the Laplacian. Laplacian can then be computed as with a regular grid. As the last step, the Laplacian computed on a regular grid is sampled on the original irregular grid.

Regular Grid Samples are sampled on a regular grid — they are evenly spaced.

Random Grid Samples coordinates are produced

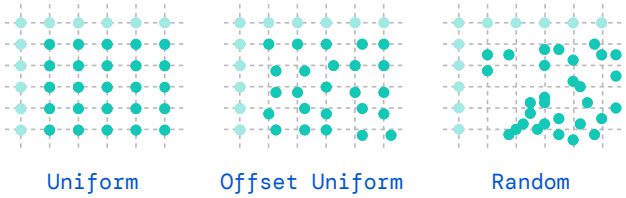


Figure 15. We utilize FFT-based derivative approximation to compute the Laplacian. Thus we use a regular grid in the proposed solution. We attempted to lift the restriction of using the regular grid by several experiments described in Chapter 4.3.

using a uniform random generator. As illustrated in Figure 15, sampling with a random generator fails to produce evenly spaced samples — producing samples too close to each other results in numerical instabilities.

Offset Regular Grid To create data sampled on a non-regular grid with evenly spaced samples, we perturb data sampled on a regular grid. Given a grid (96×96), the sample coordinates are perturbed using the following method:

$$\hat{x} = x + \mathcal{U}(-\epsilon, \epsilon), \tag{16}$$

where x is the sample coordinates and $\mathcal{U}$ stands for uniform distribution. The perturbation $\epsilon = 0.0026$ is calculated as a 25% perturbation in a 96×96 grid ($\epsilon = 0.25 * 1/96$).

Results As can be seen in Figure 16, with data sampled on a regular grid, the model performed the best. Any irregularity in the data sampling grid results in models incapable of stable unrolling. Although predicted wavefields on irregular grids do not look overly different from the prediction on a regular grid (Figure 16), the difference is more distinguishable in Table 2.

Table 2. Model is not capable of stable unrolling on an irregular grid. The irregular grid increases the model's error substantially. Models are evaluated only with 3 unroll iterations due to limited access to the computational resources.

	Regular	Offset Reg.	Random
MSE	24.8	27.1	27.9
Optim. steps	$\approx 63k$	$\approx 70k$	$\approx 120k$

4.4 Super-resolution

To test the model's ability to predict wavefield for the upsampled points, we insert data sampled on the irregular grid into the uniformly sampled data in order to increase resolution in certain areas of the computational domain (Figure 17).

During the training, we simulate super-resolution by generating upsampled 500 points using a uniform random generator. The upsampled data is then concatenated with the uniformly sampled data. Since the FFT is defined well only on a regular grid, the residual can be computed only for uniformly sampled data. Thus, we obtain the residual of random points by interpolating the residual from uniformly sampled neighbors.

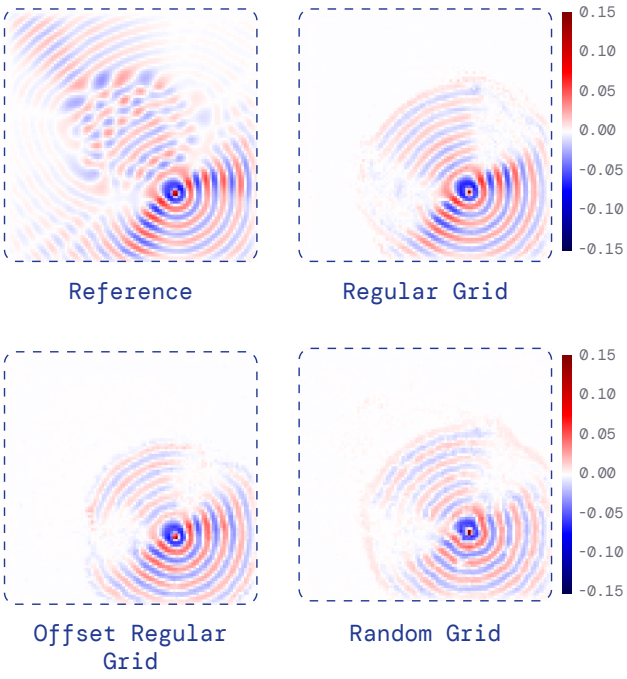


Figure 16. Proposed model produces the best results with data sampled on a regular grid. However, data sampled on offset regular grid still produces usable wavefields. However, randomly sampled data suffers from an interpolation error, due to unevenly sampled domain. All models are evaluated only with 3 unroll iterations due to limited access to the supercomputer

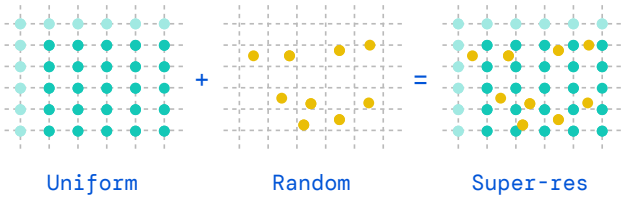


Figure 17. To train the network to perform prediction for upsampled data, 500 random points are concatenated to the uniformly sampled data. The residual for the upsampled data is computed by interpolating residual from the uniformly sampled neighbors.

Results To evaluate predictions of upsampled points by our model, the MSE is computed only from the upsampled points and not from the uniformly sampled data.

As shown in Table 3, our model yields lower error than the interpolation method. Despite that, the MSE of the upsampled points (MSE $= 20.88$) is substantially

higher in comparison with the prediction of uniformly sampled points (MSE $= 13.65$). We hypothesize that the higher error of the upsampled points is caused by their irregularity, as shown by experiments described in Section 4.3.

Table 3. As a baseline solution, we used interpolation of the neighboring nodes. Compared with our model, the MSE error of the upsampled points is notably lower. Nonetheless, the MSE of the predicted uniform samples is 13.65, which is significantly lower.

	Interpolated	**Predicted**
MSE	22.51	**20.88**

4.5 Training Noise

Pfaff et al. [24] demonstrated that noise injection during training improved the unroll stability and lowered overall error. The interpretation is that the error accumulating during unrolling can be reduced by simulating this error on the network input. It leads the network to learn to correct the error, thus lowering the overall accumulated error occurring during unrolling.

Table 4. Injecting 1.2% training noise described by Equation (17), simulates prediction error. Using the training noise forces the model to learn to correct its prediction error during unrolling.

Noise [%]	0	0.6	1.2	2.4	4.8
MSE	14.58	14.56	**14.37**	15.76	15.43

Although, Pfaff et al. [24] uses noise injection in the prediction of dynamical systems, we test the training noise in our proposed solution as well. We analyzed the distribution of the noise after 8 unroll iterations and decided to model the error within 10%. The training noise is set to the 8-th root of the error. The unroll error depends on the predicted wavefield, due to that fact we model the training noise as the multiplicative noise. The following equation describes the training noise:

$$\tilde{u_k} = u_k + u_k * \text{noise}, \tag{17}$$

where u_k is the predicted wavefield at k-th iteration, by injecting the noise in wavefield u_k, the noise is implicitly added to the residual R_k as well.

As shown in Table 4, by injecting noise 1.2% during training, the model performance improved. Hence, the model learned to reduce the prediction error from previous iterations. Any higher training noise resulted in worse results.

4.6 Pruning

One of the primary motivations behind neural solvers is to reduce the time required to obtain a PDE solution. Due to that reason, we analyze pruning the weights of the network, which might improve the network speed. However, Frankle and Carbin [8] formulate the *Lottery Ticket Hypothesis* (LTH), stating that it can increase the model accuracy as well. Frankle and Carbin [8] uses the *Iterative Magnitude Pruning* (IMP) to find Winning Tickets — pruned models performing better than the unpruned one.

To formulate sparsity of the network, P_m denotes the percentage of unpruned weights — $P_m = 75\%$, when 25% of weights are pruned.

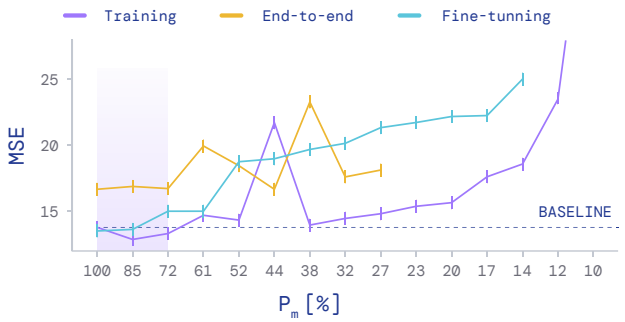


Figure 18. Any pruned network, with better performance than the best unpruned network (baseline), is referred to as *winning ticket*. Winning tickets occurs only when the network is pruned before the first phase of the training, where any networks with $P_m \geq 72\%$ is are a winning ticket (purple area). P_m denotes the percentage of unpruned weights.

Our proposed method differs by employing two-phased training. Pruning can be applied at the beginning of the first phase or the second one. We followed the LTH methodology and reset the model weights after each pruning. Three different scenarios were tested:

Training The pruning is applied before the whole training starts, which is the beginning of the first training phase — training with 3 unroll iterations.

Fine-tuning As the name suggests, the model is pruned at the beginning of the second training phase (fine-tunning) — training with 8 unroll iterations. The model weights are reset to the state at the beginning of the second training phase, meaning the weights are not set to initialization ones.

End-to-end End-to-end training with 8 unroll iterations were tested, where the network was trained end-to-end. The network is pruned at the beginning of the training — same as Frankle and Carbin [8].

Results As Figure 18 shows, the winning ticket can be obtained only with the two-phased training. We

Table 5. Despite preserving the total message passing count equal to 60, exchanging the unrolls for message pass count in a single unroll iteration results in higher error. We assume that the unroll iterations are essential because of the residual. Without residual as an input, the model is not able to predict complex wave reflections. GB refers to a Graph Block, which corresponds to a single message passing.

	60 Unrolls, 1 GB	20 Unrolls, 3 GBs	12 Unrolls, 5 GBs	6 Unrolls, 10 GBs	4 Unrolls, 15 GBs	2 × 30 GBs
MSE	46.27	14.52	**14.00**	19.69	21.66	21.88

believe that the *fine-tuning* approach failed to produce any winning tickets is due to not resetting the weight to the ones from initialization. *End-to-end* approach failed because it requires significantly more optimization steps than the two-phased approach (see Chapter 3.4). Leaving the two-phased training approach to be most likely to produce any winning tickets. The winning tickets were obtained only with low levels of pruning $P_m \geq 72\%$, meaning the network might not be too overparametrized. Despite that there are multiple methods for stabilizing the LTH [20, 9], due to the expensive nature of the LTH experiments, we did not conduct any related experiments.
Although the winning tickets are not sufficiently sparse to utilize sparse multiplication or improve inference speed, iterative pruning can be used to increase the model's accuracy.

4.7 Other Hyperparameters

Hop Connections The decision to use hop connections was based on the results from the first phase of the training. The error decreased significantly with hop connections in the first phase. However, as can be seen in Table 6, after fine-tuning, where the model truly learns the propagation of sound waves, hop connections result only in an insignificant error decrease.

Table 6. Reason behind the usage of multiple hop connections in the proposed solution was based on the results from the first stage of the training. Nonetheless, MSE from the fine-tuning stage prove that the hop connections are unnecessary.

	Hops $\{3, 5, 10\}$	No Hops
First Phase	24.78	26.12
Fine-tuning	14.58	14.62

We hypothesize that the hop connections redundancy in the latter stage of the training is that instead of memorizing wavefield patterns, the network starts to act as an iterative solver, where the prediction requires only a smaller neighborhood, rather than the whole domain.

Unroll Iterations vs. Message Pass Count This section examines the trade-off between unroll iterations and message passing count. A new residual is computed from the predicted wavefield with every unroll iteration. On the contrary, a Graph Block passes only a message (hidden state) to the next block. In the following experiments, we fixed the total message passing count to 60, only the ratio between unroll iterations and graph blocks changes.
As Table 5 shows, unroll iterations are essential for the wavefield prediction. More Graph Blocks (GB) results in more learnable parameters — increasing the model's capacity. Nevertheless, model with 5 Graph Blocks has the lowest error. We suppose that unroll iterations are crucial due to the addition of residual to the network's input. We observed that without the residual as input, the model cannot predict more complex wave reflections and interactions. We assume, it is due to inability of the network to learn a laplacian operator. This experiment was conducted after the design choices, for that reason the proposed solution has 10 Graph Blocks.

4.8 Comparison with Helmnet

As depicted in Figure 19, the accuracy of Helmnet [34] is remarkably higher than the accuracy of our solution. We believe that the higher error of our model is caused by not using a replay buffer. Thus, our model is not trained with enough unroll iterations to learn to predict the wavefield with such low error. Our assumption is supported by Figure 20, when our model reaches a certain level of error, it stops to improve the predicted wavefield.

Due to a larger amount of data to be processed (nodes and edges), our model is significantly slower than Helmnet and k-Wave. Thus, failing at one of the main aspects of the neural solvers.

5. Conclusions

Our Graph Network is capable of solving large domains such as 512 × 512, thus moving beyond "toy" problems. Unfortunately is slower than Helmnet as well as the reference solver k-Wave. In addition, Helmnet achieves a lower error by almost two orders of magnitude compared to our solutions. Nevertheless, our model is able to perform super-resolution, where

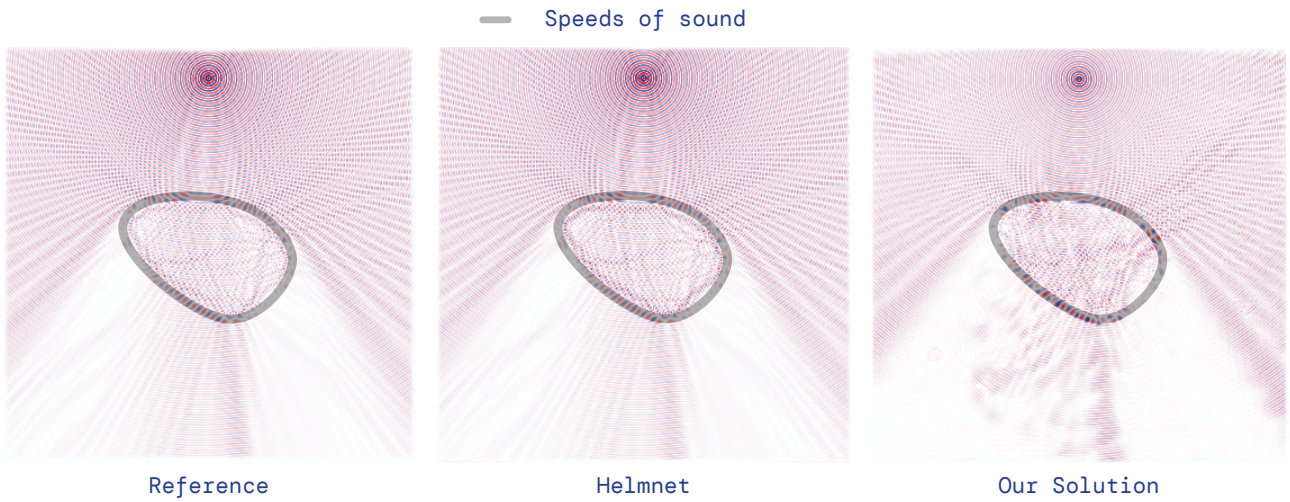


Figure 19. Helmnet produces significantly more accurate wavefield than our solution. Nevertheless, our solution produce adequate solution, but it cumulates a higher error with every iteration.

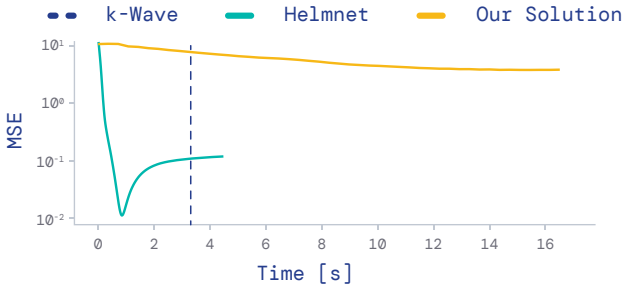


Figure 20. Our model is notably slower than Helmnet and k-Wave, and is not able to reach low error as Helmnet. We assume, that the slower speed is because graph contains more data to process (edges) than images, despite containing the same amount of information. The experiment was conducted on a machine with a GPU A100.

it reached a lower error than the baseline method — linear interpolation.

Even though our model was trained only on samples with a single source, it can predict a wavefield with multiple sources, proving that our model learned the interaction between sound waves. Additionally, our model can perform inference in a 512×512 computational domain, although the model was trained only in 96×96 domain.

We demonstrated that the Graph Networks are able to solve a second-order time-independent PDE within a large computational domain. Also, we outlined the problem of the difficulty of training an effective model on an irregular grid. Lastly, we tested a neural solver against the Lottery Ticket Hypothesis [8]. We were able to produce winning tickets, but only with a low level of sparsity $P_m \geq 72\%$. The winning tickets are not sparse enough to utilize sparse multiplication, but the iterative pruning can be used to improve the accuracy of the model.

In the future, given the flexibility of Graph Networks, an emphasis can be given to training on the data in a 2D space and later translating it to the 3D space. We hypothesize that AGS could replace the spectral method to compute the residual with denser sampling.

Acknowledgements

Firstly, I am very grateful to my supervisor prof. Ing. Adam Herout Ph.D. for the patient guidence and advice he has provided. I am extremely grateful to my girlfriend MUDr. Jana Hantáková for mental support as well as neurological works analysis. I am also deeply indebted to Dr. Antonio Stanziola for the extensive guidence, his never-ending positivity and support. Additionally, I would like to express many thanks to doc. Ing. Jiří Jaroš Ph.D. for his advicing and access to the supercomputer. Last but not least, I thank to Ing. Marta Jaroš for access to the computational resources.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018. URL http://arxiv.org/abs/1803.08375.

[2] Anima Anandkumar, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Nikola Kovachki, Zongyi Li, Burigede Liu, and Andrew Stuart. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020. URL https://openreview.net/forum?id=fg2ZFmXFO3.

[3] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL http://arxiv.org/abs/1806.01261.

[4] Alfredo Bermudez, Luis Hervella-Nieto, Andrés Prieto, and R. Rodrıguez. An optimal perfectly matched layer with unbounded absorbing function for time-harmonic acoustic scattering problems. *Journal of Computational Physics*, 223: 469–488, 05 2007. doi: 10.1016/j.jcp.2006.09.018.

[5] Norbert N. Bojarski. The k-space formulation of the scattering problem in the time domain. *Journal of the Acoustical Society of America*, 72: 570–584, 1982.

[6] Norbert N. Bojarski. The k-space formulation of the scattering problem in the time domain: An improved single propagator formulation. *Acoustical Society of America Journal*, 77(3):826–831, March 1985. doi: 10.1121/1.392051.

[7] Steffen Eger, Paul Youssef, and Iryna Gurevych. Is it time to swish? comparing deep learning activation functions across NLP tasks. *CoRR*, abs/1901.02671, 2019. URL http://arxiv.org/abs/1901.02671.

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[9] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. The lottery ticket hypothesis at scale. *CoRR*, abs/1903.01611, 2019. URL http://arxiv.org/abs/1903.01611.

[10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1263–1272. JMLR.org, 2017.

[11] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005. doi: 10.1109/IJCNN.2005.1555942.

[12] David S. Hersh and Howard M. Eisenberg. Current and future uses of transcranial focused ultrasound in neurosurgery. *J Neurosurg Sci*, 62(2): 203–213, Apr 2018.

[13] Frank Ihlenburg and Ivo Babuska. Finite element solution of the Helmholtz equation with high wave number Part I: The h-version of the FEM. *Computers & Mathematics With Applications*, 30: 9–37, 1995.

[14] Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1lyTjAqYX.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[16] Vibhor Krishna., Francesco Sammartino., and Ali Rezai. A Review of the Current Therapies, Challenges, and Future Directions of Transcranial Focused Ultrasound Technology: Advances in Diagnosis and Treatment. *JAMA Neurol*, 75 (2):246–254, 02 2018.

[17] Isaac E. Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5): 987–1000, 1998. doi: 10.1109/72.712178.

[18] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

[19] Wenzhuo Liu, Mouadh Yagoubi, and Marc Schoenauer. Multi-resolution Graph Neural

Networks for PDE Approximation. In *Artificial Neural Networks and Machine Learning – ICANN 2021*, volume 12893 of *Lecture Notes in Computer Science*, pages 151–163. Springer International Publishing, September 2021. doi: 10.1007/978-3-030-86365-4_13. URL https://hal.archives-ouvertes.fr/hal-03448278.

[20] Jaron Maene, Mingxiao Li, and Marie-Francine Moens. Towards understanding iterative magnitude pruning: Why lottery tickets win. *ArXiv*, abs/2106.06955, 2021.

[21] Claudio Mancinelli, Marco Livesu, and Enrico Puppo. Gradient field estimation on triangle meshes. 10 2018. doi: 10.2312/stag.20181301.

[22] T. Douglas Mast, Laurent P. Souriau, Donald L. Liu, Makoto Tabei, Adrian I. Nachman, and Robert C. Waag. A k-space method for large-scale models of wave propagation in tissue. *IEEE Trans Ultrason Ferroelectr Freq Control*, 48(2): 341–354, Mar 2001.

[23] Gregory A. Newman and David L. Alumbaugh. Frequency-domain modelling of airborne electromagnetic responses using staggered finite differences. *Geophysical Prospecting*, 43:1021–1042, 1995.

[24] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

[25] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[26] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

[28] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8459–8468. PMLR, 2020. URL http://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

[29] Aliaksei Sandryhaila and José M. F. Moura. Discrete signal processing on graphs: Graph fourier transform. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6167–6170, 2013. doi: 10.1109/ICASSP.2013.6638850.

[30] Franco Scarselli, Sweah Liang Yong, Marco Gori, Markus Hagenbuchner, Ah Chung Tsoi, and Marco Maggini. Graph neural networks for ranking web pages. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '05, page 666–672, USA, 2005. IEEE Computer Society. ISBN 076952415X. doi: 10.1109/WI.2005.67. URL https://doi.org/10.1109/WI.2005.67.

[31] Bhavya R. Shah, Vance T. Lehman, Timothy J. Kaufmann, Daniel Blezek, Jeff Waugh, Darren Imphean, Frank F. Yu, Toral R. Patel, Shilpa Chitnis, Richard B. Dewey, Joseph A. Maldjian, and Rajiv Chopra. Advanced MRI techniques for transcranial high intensity focused ultrasound targeting. *Brain*, 143(9):2664–2672, 09 2020.

[32] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

[33] Arnold Sommerfeld. Die greensche funktion der schwingungslgleichung. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 21:309–352, 1912. URL http://eudml.org/doc/145344.

[34] Antonio Stanziola, Simon R. Arridge, Ben T. Cox, and Bradley E. Treeby. A helmholtz equa-

tion solver using unsupervised learning: Application to transcranial ultrasound. *Journal of Computational Physics*, 441:110430, Sep 2021. ISSN 0021-9991. doi: 10.1016/j.jcp.2021.110430. URL http://dx.doi.org/10.1016/j.jcp.2021.110430.

[35] Makoto Tabei, T. Douglas Mast, and Robert C. Waag. A k-space method for coupled first-order acoustic propagation equations. *J Acoust Soc Am*, 111(1 Pt 1):53–63, Jan 2002.

[36] Nils Thuerey, Philipp Holl, Maximilian Müller, Patrick Schnell, Felix Trost, and Kiwon Um. Physics-based deep learning. *CoRR*, abs/2109.05237, 2021. URL https://arxiv.org/abs/2109.05237.

[37] Bradley E. Treeby and Benjamin T. Cox. k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields. *Journal of Biomedical Optics*, 15(2):1 – 12, 2010. doi: 10.1117/1.3360308. URL https://doi.org/10.1117/1.3360308.