Journal of Computational Physics (2019)

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Obstacle Segmentation based on the Wave Equation and Deep Learning

Adar Kahana[a,*], Eli Turkel[a], Shai Dekel[a], Dan Givoli[b]

[a]*Department of Applied Mathematics, Tel-Aviv University, Tel-Aviv 69978, Israel*
[b]*Department of Aerospace Engineering, Technion, Haifa 32000, Israel*

ARTICLE INFO

Article history:
Received 1 May 2013
Received in final form 10 May 2013
Accepted 13 May 2013
Available online 15 May 2013

Communicated by S. Sarkar

ABSTRACT

We model the inverse physical problem of identifying an underwater obstacle by using the acoustic wave equation. Measurements are collected in a set of sensors placed in the medium and we use this partial information to approximate a segmentation of the obstacle in its correct location. This is an ill-posed problem and so we propose a novel deep learning architecture that takes as input the sensor data and computes an approximate segmentation map of the obstacle.

1. Introduction

1.1. Physical model

We wish to locate obstacles in an underwater medium using acoustic wave propagation (sonar). Given measurements, we formulate the problem as an inverse problem [12]. Inverse problems are important mathematical problems since their solution is frequently information one cannot directly observe. They appear in a variety of fields including acoustics, geophysics, material testing (including non-destructive testing), radar and signal processing, computer imaging and vision, tomography, meteorology and oceanography, remote sensing and machine learning.

Solving inverse problems is a vast field and there are many examples e.g. [15, 26, 27]. A difficulty with inverse problems is that they are frequently ill-posed and so the solution need not exist or be unique. Additionally, if the parameters of the problem are perturbed, the solution might not depend continuously on the parameters. Since we have only very partial spatial information the ill-posedness is more likely to prevent much knowledge of the complete solution. Given properties of the medium and a set of measurements of the wave field at some points in space and time and assuming we know the location of the original source/s that generated the wave field we wish to find the location of a scatterer in the medium. The scatterer properties (shape, size, density etc.) are also of interest. Because of the ill-posedness of the inverse problem, one cannot retrieve the full wave equation solution when one has access to only partial data at only a small subset of it (sensors) and so we wish to find only some properties of the scatterer.

*Corresponding author: Tel.: +972-546-895-475;
e-mail: adarkahana@mail.tau.ac.il (Adar Kahana)

1.2. Innovation

Solving inverse problems, including the problem of scatterer location and identification, is a vast field and has been studied by many authors. e.g. [15, 26, 27]). Some direct methods for approximating the location and shape of the obstacle involve some prior knowledge; for example, assuming that the obstacle is a rectangle one can model it with 4 degrees of freedom. After modeling the obstacle using a set of parameters one can use an optimizer to reconstruct the parameters, trying to minimize a predefined cost function. This process is usually computationally intensive and requires intelligent construction of the cost function. Some recent developments in this field involve improving the obstacle parametrization, cost function or optimizer (e.g. [14, 16, 19]. Other methods include arrival time and adjoint techniques are discussed in the next section.

Recently there is a growing interest in applying machine learning to various inverse problems when real training data exists or synthetic training data can be generated [22, 21]. General machine and deep learning algorithms that handle input data and fit a certain classifier are commonly used. The general algorithms are tuned, by adjusting the algorithm parameters, for the specific task until a desired level of accuracy is achieved.

We propose a new method based on machine learning which does not require parametrization of the obstacle and can handle obstacles of arbitrary shapes. The training phase of our segmentation is computationally intensive, however inference using the trained model is quick and can be deployed to real time solutions. Moreover, the model is physically informed [22] making use of the wave problem context. The innovative architecture shows promising results and we present a new application combining a state of the art learning algorithm with the classical methods of applied mathematics.

2. Numerical development

2.1. Mathematical formulation

The wave problem is given by -

$$\begin{cases} \ddot{u}(\overrightarrow{x},t) = \nabla\cdot(c(\overrightarrow{x})^2\nabla u(\overrightarrow{x},t)) & \overrightarrow{x}\in\Omega, t\in(0,T] \\ u(\overrightarrow{x},0) = u_0(\overrightarrow{x}) & \overrightarrow{x}\in\Omega \\ \dot{u}(\overrightarrow{x},0) = v_0(\overrightarrow{x}) & \overrightarrow{x}\in\Omega \\ u(\overrightarrow{x},t) = f(\overrightarrow{x},t) & \overrightarrow{x}\in\partial\Omega_1, t\in[0,T] \\ \nabla u(\overrightarrow{x},t) = g(\overrightarrow{x},t) & \overrightarrow{x}\in\partial\Omega_2, t\in[0,T] \qquad \partial\Omega_1\cup\partial\Omega_2 = \partial\Omega \end{cases} \tag{1}$$

where $u(\overrightarrow{x},t)$ is the wave amplitude or acoustic pressure. For given initial conditions (u_0, v_0) and boundary conditions $(f, g$, of types Dirichlet and Neumann respectively) the wave problem (1) is well-posed. Therefore, small changes in the conditions cause small changes in the solution and there exists a unique continuous solution to the problem inside the domain.

We are interested in identifying an obstacle - Ω_s - which is a perfect acoustic reflector. The obstacle is a small sub-region Ω_s such that $\Omega_s\subset\Omega$ and the boundary condition on Ω_s is homogeneous and of type Dirichlet An alternative is setting the wave velocity $c(\overrightarrow{x})^2$ to zero inside Ω_s. The shape, size, location and other properties of Ω_s are unknown.

To find the obstacle we measure the solution of the wave equation at various times. It is physically impossible to record the signal at every spatial point of the domain. Instead, in a physical experiment we can record the acoustic pressure in only a small number of sensors spread around the domain. The output is a set of measurements $u(\overrightarrow{x_n},t_k)$ where $\{\overrightarrow{x_n}\}_{n=1}^{N_{sensors}}$ is a sequence of sensor locations such that $N_{sensors}$ is the number of sensors placed in the medium and $\{t_k\}_{k=0}^{N_{steps}}$ is a time series consisting of N_{steps} time steps. We synthetically simulate the physical experiment of wave propagation to get measurements in artificial sensors in the medium Ω. Without knowing the entire solution $u(\overrightarrow{x},t)$, but only the recordings $u(\overrightarrow{x_n},t_k)$, we want to recover Ω_s.

There are several methods of identifying Ω_s in the literature. For example, assuming prior knowledge about the shape and size one can parameterize Ω_s and then try to "guess" Ω_s, run the simulation and define a scoring function to determine if the guess was correct. If not, then one uses a new guess with different parameters until one minimizes the misfit between the solution and the measurements. Alternatively, the TRAC algorithm [5] assumes no knowledge of the shape of the obstacle however, the final solution gives only the position but not the shape of the obstacle. This optimization process frequently requires a lot of computational time. There are several methods that improve this process e.g. [20]. The arrival time algorithm e.g. [2, 3, 18, 23, 25], does not require a priori knowledge of the

shape but again yields only the location but not the shape of the obstacle. We note, that learning methods have been proposed to improve the time arrival scheme [28, 29]. Another technique is the adjoint method e.g. [6, 1, 24] where the shear modulus distribution is reconstructed from given strain data. In general, the adjoint method identifies the properties of the medium, and not a specific object. All objects are represented as large local changes in the medium properties. When one examines the result, the obstacle stands out as a region in the medium with very different properties (e.g., wave velocity) than the background. This includes the location, shape and size of the region where the properties are very different. Since these methods do not find the shape directly they do not give a good resolution for it. Arrival Time imaging is notorious as being very crude, and even the adjoint method tends to give a blurred picture of obstacles. We propose a learning method that requires no prior knowledge about the shape or location of the obstacle but gives both the obstacle location and a sharper obstacle identification than the other approaches. Additionally, once the deep learning model is trained, using it to predict an obstacle given input sensor measurements, takes only a few milliseconds.

The setup consists of generating multiple scenarios differing by the shape, size and location of Ω_s. For a given Ω_s we solve the wave equation and save the solution only in the sensors $u(\overrightarrow{x_n}, t_k)$. We randomly generate multiple scatterers differing in the shape, size and location of the scatterer and mark them as Ω_{s_q} such that for the generation of $N_{samples}$ scenarios we have $1 \leq q \leq N_{samples}$. After simulating all the forward processes we have a set of measurements $\{u_q(\overrightarrow{x_n}, t_k)\}_{q=1}^{N_{samples}}$ which serves as the data-set. This setup can be done either experimentally or synthetically - in this paper we used synthetic measurements, as explained in the following section.

2.2. Numerical scheme

We assume that the large domain, including the obstacle, is rectangular and model the domain using a uniform two dimensional grid sized $M \times M$ (one can easily generalize to any domain). We specify a boundary condition on the external boundaries and solve inside the domain. Each sensor is a node $\{(x_k, y_k)\}_{k=1}^{N_{sensors}}$ where the number of sensors is much smaller than the number of nodes in the domain. We model the obstacle as a shape inside the domain such that $u(\overrightarrow{x}, t)$ at every node inside the obstacle has the value zero throughout the entire computation procedure. This results in a stair-casing process which adds numerical errors. However, for finding the location and shape of the obstacle these stair-casing errors are negligible. The numerically approximated scatterer will be marked by O_q, $1 \leq q \leq N_{samples}$. Each of the scatterer representations is a binary segmentation - zeros inside the obstacle and ones outside - yielding a homogeneous Dirichlet boundary condition for the numerical representation of the scatterer.

We approximate the solution of the wave equation in space and time using the explicit compact second order accurate central difference finite difference scheme in both space and time. The time step is limited by the Courant-Friedrichs-Lewy condition:

$$\Delta t \leq \frac{1}{c_{\max}\sqrt{2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)}}, \tag{2}$$

where Δx and Δy are the distances between two adjacent grid points in the directions x and y respectively.

We solve the equation for N_{steps} and save the sensor recordings. After simulating the forward process we get a matrix of size $N_{steps} \times N_{sensors}$ containing the synthetically generated acoustic pressure measurements received in each sensor in each time step. To create the training and testing data-sets we run this simulation multiple times, where the difference between each and every simulation is the shape, location and size of the scatterer O_q. Each simulation produces a matrix containing the synthetic sensor data for the specific randomly generated scatterer. After saving the outputs of all forward synthetic simulations we get a tensor of size $N_{samples} \times N_{steps} \times N_{sensors}$. The labels which we try to predict are the obstacle segmentations, each represented by a $M \times M$ sized binary image. We remark that in most tests the number of samples was 25,000 and $M = 128$ which implies that the total storage size of the dataset is 500MB. An example of the source and sensors placement and an arbitrary scatterer is given in Figure 1.

After training our segmentation model, we apply it to a test set of sensor measurements. Each such inference produces a predicted binary segmentation map of the obstacle, denoted as $\bar{O}_q$. The binary image is the equivalent numerical interpretation of Ω, so the scheme recovers the medium in which the PDE is solved. We split the data-set into two subsets - a training set with N_{train} samples and a testing set with N_{test} samples ($N_{train} + N_{test} = N_{samples}$). To assess the performance of the model we use the test subset and compare each prediction $\bar{O}_q$ to a known O_q where $1 \leq q \leq N_{test}$. To compare $\bar{O}_q$ and O_q we have several measures of the error:

- **Mean Square Error:** $\frac{1}{M^2 \cdot N_{test}} \sum_{q=1}^{N_{test}} \|O_q - \bar{O}_q\|^2$, where: $\|O_q - \bar{O}_q\| = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{M} |(O_q)_{ij} - (\bar{O}_q)_{ij}|^2}$.

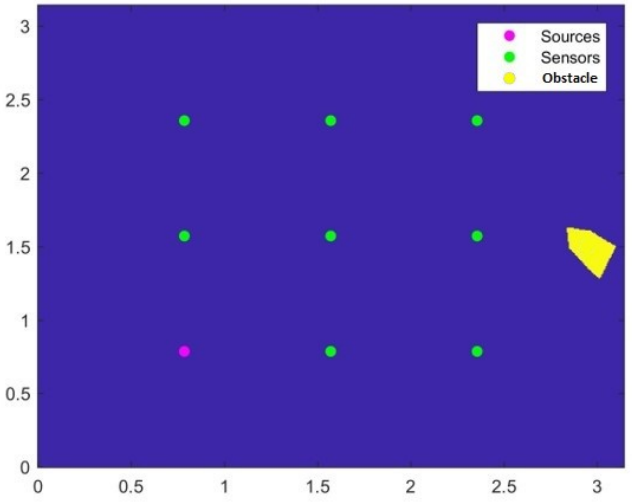


Fig. 1: Example of a 2D medium with 8 sensors (green), a source location (pink) and a binary segment of an obstacle (yellow)

- **Intersection over Union (IOU):** A popular measure for segmentation: $\frac{1}{N_{test}}\sum_{q=1}^{N_{test}}\frac{|O_q\cap\bar{O}_q|}{|O_q\cup\bar{O}_q|}$, where $|O_q \cap \bar{O}_q|$ implies the number of pixels with value 1 in both O_q and $\bar{O}_q$ and $|O_q \cup \bar{O}_q|$ implies the number of pixels with value 1 in either O_q or $\bar{O}_q$.

2.3. Deep-learning framework

We build a neural network such that the input is the sensor data recorded in each simulation with a different scatterer. The neural network is composed of either convolutional or fully connected (dense) layers whose coefficients (weights) are initially unknown. Each layer also applies a component-wise non-linear activation before it passes forward its output. We define a target function (specifics are given in the subsections) and iterate using a gradient descent algorithm to minimize a target function with respect to all the weights. We design the network architecture (number and sizes of convolution and dense layers) to get the best convergence of the gradient descent algorithm, resulting in a better intersection over union score on a predefined testing data-set.

The fully connected layers try to learn all the connections between the input of the layer and the its output using multiplication by unknown weights. This process is done by matrix multiplication - for an input matrix A of size $N_{train}\times N_{features}$ and an unknown matrix W of size $N_{features}\times N_{layersize}$ the layer output is $\mathcal{L}(A) = W\cdot A + b^T$ where b is a bias vector of unknown weights as well. The unknown elements are then approximated during the training process using the gradient descent algorithm, with respect to a target function.

The convolution layers try to learn local patterns in the data. They consist of a set of filters of unknown weights w_i. Convolving the input of the layer with each of the filters produces the output of the layer: $\mathcal{L}(A)_i = w_i * A$. The filters weights are then approximated during the training process using the gradient descent algorithm as for the fully connected layer weights. The difference between the fully connected and the convolutional layers is finding global connections versus local patterns.

We also use pooling layers, which are typically used in deep learning, to reduce the dimension of a layer's output. For example, a 2×2 max-pooling layer applied on a matrix takes only the maximal value from each cluster of 4 elements. These layers do not include weights and are irreversible.

Each layer is followed by an nonlinear activation - the output is manipulated in a way that will suit the data. We use a rectifier ($x \to max(0, x)$) and sigmoid ($x \to \frac{1}{1+e^{-x}}$). The sigmoid function serves as a good differentiable approximation to the Heaviside step function. Each pixel in the output of the network is a probability so after using the sigmoid activation these probabilities are centered around 0 and 1, suitable to the binary output we want.

After designing the network (number of layers, type and size of each layer etc.) we use a gradient descent algorithm to determine all the weights. We define a loss (target) function which the gradient descent tries to minimize with respect to all the weights. If the network was designed correctly and the problem geometry was learned, the

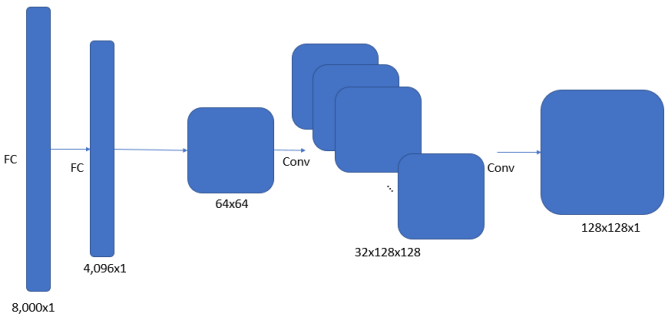


Fig. 2: Naive network architecture

gradient descent algorithm will converge. Therefore, after a certain number of iterations (called epochs) we expect a set of discovered weights that given new data (measurements that were not used for training) as input, will produce an output that predicts the segmentation with high accuracy that can be checked using the metrics defined in 2.2.

Although we are interested in minimizing the IOU loss during the training phase, the optimization method requires a differentiable loss function to compare the true labels of the sample O_q and the predicted ones $\tilde{O}_q$. Therefore, one good surrogate loss function is the Negative Log-Likelihood (NLL). After each optimizer iteration we calculate $NLL = -\frac{1}{N_{train} \cdot M \cdot M} \sum_{q=1}^{N_{train}} \sum_{i=1}^{M} \sum_{j=1}^{M} (O_q)_{i,j} \log((\tilde{O}_q)_{i,j})$ and train the network to minimize this target function.

Another good surrogate loss function that approximates the IOU loss is the soft-IOU loss. The proposed method is to approximate the intersection over union using:

$$\frac{1}{N_{train}} \sum_{q=1}^{N_{train}} \frac{< O_q, \tilde{O}_q >}{|O_q|_1 + |\tilde{O}_q|_1 - < O_q, \tilde{O}_q >},$$

where $< O_q, \tilde{O}_q > := \sum_{i=1}^{M} \sum_{j=1}^{M} \left((O_q)_{i,j} \cdot (\tilde{O}_q)_{i,j} \right)$ and $|O_q|_1 := \sum_{i=1}^{M} \sum_{j=1}^{M} (O_q)_{i,j}$. We use the NLL loss for the first gradient descent iterations (epochs) and after we get saturation (i.e. the NLL loss difference between epochs is very small) we switch to using the soft-IOU loss for another few epochs (~15). We still calculate the NLL loss during these epochs and observe a much lower NLL loss.

The output of the network is a matrix of probabilities with size $M \times M$ denoted by $\tilde{O}$. Each coordinate of $\tilde{O}$ is the probability of that coordinate to be inside Ω_s. Hence, $\tilde{O}$ is the probability of pixels to be inside the scatterer. We convert $\tilde{O}$ into a binary image using a threshold to get $\bar{O}$ and check the accuracy using the methods discussed in (2.2).

2.3.1. Naive deep learning architecture

The initial approach we consider is a neural network with a very simple architecture - two fully connected layers followed by two convolution layers. The goal is to convert the multivariate time series data (network input) to a two dimensional spatial data (network output) and a transition from fully connected to convolution layers is suitable here. The input is a column-stacked recordings in the sensors (that we synthetically produce during the solution of the wave problem) and the output is a probabilistic segmentation map of the obstacles. In this case the input data is column-stacked values recorded in the sensors for each sample, yielding a $N_{train} \times N_{steps} \cdot N_{sensors}$ sized matrix.

We train the network to minimize the NLL and soft-IOU loss functions using the ADAM optimizer [17]. Results are given in section 3 and a sketch of the network architecture is given in Figure 2.

2.3.2. A space-time flow deep learning architecture

The motivation for this architecture comes from the physical nature of the problem we are solving. The input data for the network is a time series of recordings, a tensor of size $N_{train} \times N_{steps} \times N_{sensors}$. The input data is a time series while the output is a segmentation map of the medium, i.e. data in space. We want the network to manipulate the data in a fashion that accounts for a transition from time to space. We propose a different architecture, more complex then the naive one, to take these elements in account. The architecture is demonstrated in figure 3. We train this network with the NLL and soft-IOU loss functions using the ADAM optimizer as well.

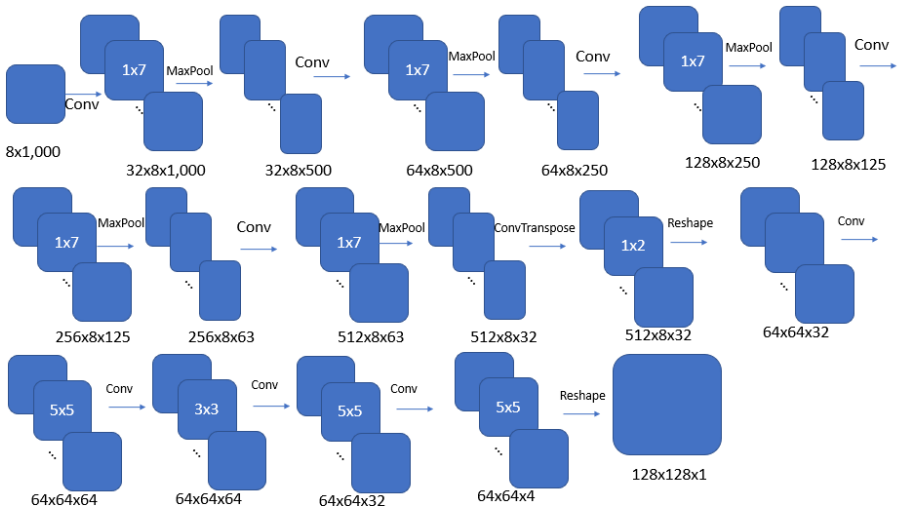


Fig. 3: Architecture of the space-time flow network

The first set of layers are 1-dimensional convolutions and pooling - transferring information from the channel (recordings in each sensor) to a new dimension which will be the spatial dimension. Results for this architecture are given in section 3 and a sketch of the network architecture is given in Figure 3.

2.4. Physically informed novel architecture

Although the results using advanced deep learning methods are very good, we propose to enhance them with an additional module that makes the network "aware" of the physical problem it is trying to approximate. We want to exploit the physical aspects of the equation and integrate them into the network. There are several methods of building neural networks that take classical methods from the applied math literature, enforce them on the learning mechanism [22] and achieve remarkable results. We propose a method specifically tailored to the time evolution of the wave problem.

We design an additional loss function component for the network using the properties of the physical wave problem that are used during the training phase. The input of the network are all the true recordings in the sensors. So, in the training part we have $\{u_q(\overrightarrow{x_n}, t_k)\}_{q=1}^{N_{samples}}$, which are the true recordings in the sensors used for training. For each sample we predict $\bar{O}_q$. We then solve the wave problem using the same numerical scheme and get a set of sensor measurements $\{\bar{u_q}(\overrightarrow{x_n}, t_k)\}_{q=1}^{N_{samples}}$ which are the predicted sensor measurements for each obstacle. We use the mean square error to compare the predicted and true recordings in the sensors. We define the loss function for the network as the weighted sum $loss_{total} = \alpha \cdot loss_{NLL} + (1-\alpha) \cdot loss_{Phy}$ where $loss_{NLL}$ is the NLL loss while $loss_{Phy}$ is the mean square error between the true and predicted sensor recordings. The parameter α is chosen by trial and error. Using $\alpha = 1$ is using exactly the architecture described in the previous subsections and using $\alpha = 0$ means approximating based only on the wave equation.

Implementing this method we face two difficulties:

- Recall that $\bar{O}_q$ is a thresholded segmentation. The network output $\tilde{O}$ is a non binary probability segmentation with $0 \leq (\tilde{O})_{i,j} \leq 1$ where $1 \leq i, j \leq M$. If we use $\tilde{O}$ to produce the predicted sensor measurements, the theoretical problem is that the boundaries of the scatterer, instead of being zero (reflective), are probabilities between 0 and 1. Therefore, we get random behavior around the scatterer and the comparison between the true and predicted sensor measurements (calculating $loss_{Phy}$) will produce useless results. Also, if we threshold $\tilde{O}$ to get a binary image the loss function will have a non-differentiable element and the gradient descent algorithm used in training the algorithm will fail.

  In order to overcome this problem, instead of considering the scatterer as a reflective element in the medium - we hope that the wave propagation velocity is close to zero inside the scatterer. Hence, we solve eq. (1) but with a modified operator:

$$\ddot{u}(\overrightarrow{x}, t) = \nabla \cdot ((1 - \tilde{O}(\overrightarrow{x}))c(\overrightarrow{x})^2 \nabla u(\overrightarrow{x}, t)) \tag{3}$$

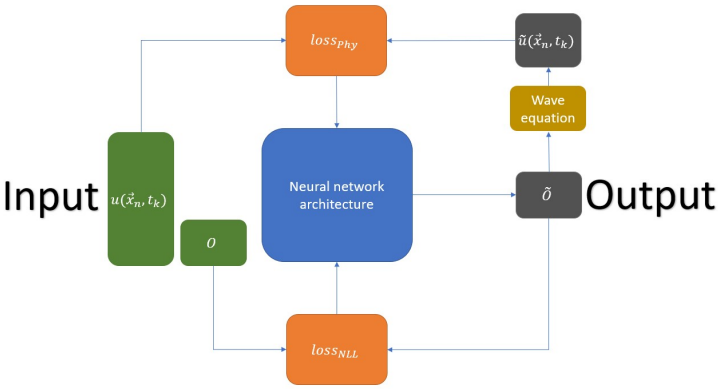


Fig. 4: Detailed process of using the physically informed loss

Thus, if the network predicts correctly, the velocity inside the scatterer will be near zero and the waves will not propagate inside, as desired. Otherwise, the loss will be high, setting a penalty for the next gradient descent step.

- The loss function which determines the next gradient descent step, has to be differentiable. This check constrains the implementation to only differentiable actions. We use a finite difference solver which can be constructed using only linear operations. So we can implement a function that solves the wave equation and can fit the loss function. Using Keras [10], we observe that the automatic differentiation mechanism [8] recognizes the implementation as differentiable and trains the network using this custom loss function.

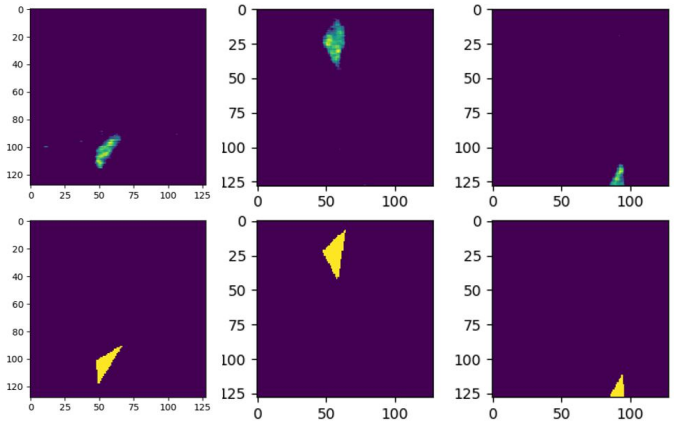
The entire process is described in Figure 4.

3. Numerical tests and results

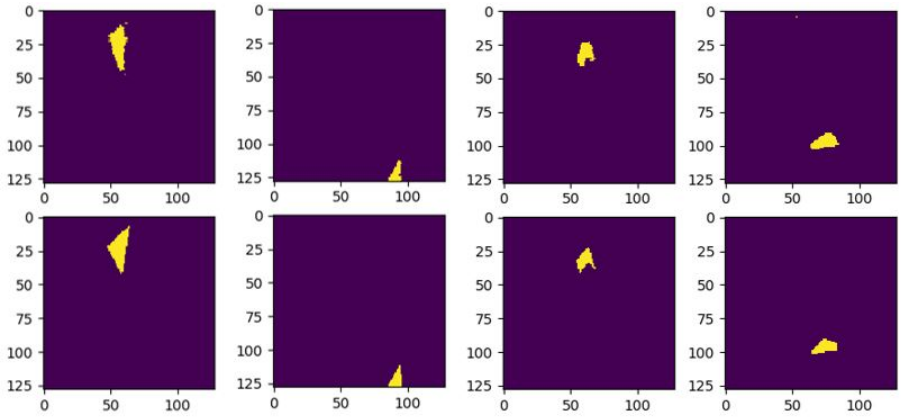
We created 25,000 simulations with varying arbitrarily shaped polygons (not necessarily convex). The polygons were created by generating a random number of edges, edge lengths and angles between edges. The number of edges and edge lengths were generated using normal distribution while the angles were generated using a uniform distribution. We split the samples such that 90% of the samples are used for training and the rest for testing. The medium size is $[0, \pi] \times [0, \pi]$ uniformly split to 128×128 points. We used 8 fixed sensors as in figure 1 and 1000 time steps. We emphasize the very small amount of measurements - adding more gives better results but we want to test the architectures on more difficult tasks (in terms of data shortage and limited resources). So, to ensure we prove our method truly generalizes and does not simply over-fit the synthetic experimental data, we keep a very small ratio between the number of training set samples and the number of possible different obstacle geometries. Additionally, after the training is complete the model is saved, using the model to predict the scatterer from a new sensor measurement (experimentally produced or synthetically) takes milliseconds.

Each sample in the data-set has a 8×1000 matrix (the recordings in the sensors) of real values and a label - a 128×128 binary image of the arbitrarily shaped polygon. The intersection over union score is the chosen accuracy metric. Training the network architecture of section 2.3.1 produces remarkable results with a 66% intersection over union score for 32 million weights. In this experiment we use the NLL loss for training with around 400 epochs (trained for a couple of minutes on a single nVidia Geforce 1050 on a laptop). The threshold used to compare the predictions and true grounds was 0.5 (if the probability of the pixel to be inside the obstacle is over 50%, the pixel is declared inside the obstacle).

Using the architecture of section 2.3.2 we achieve the same 66% IOU result on the testing data but this time for a much smaller number of weights - around 4 million. To empirically prove that the architecture of section 2.3.2 is more efficient we conducted a different test. We used a lighter dense layer in the naive version with only 2 million weights and compared with the new architecture with 2 million weights as well. The results were 51% IOU score for the architecture od section 2.3.1 and 57% IOU score for the architecture of section 2.3.2, giving us higher accuracy of



(a) Example images of the probability of the pixels to be in the scatterer. Top images are probabilities and bottom are true binary segmentation maps



(b) Example images of the predicted segmentation maps after using a threshold on the probabilities. Top images are predicted and thresholded segmentation maps and bottom are true binary segmentation maps

Fig. 5

more then 2 standard deviations. Additionally, we reached convergence for a much smaller number of epochs without over-fitting.

We tested the architecture of section 2 with the physically informed loss described in section 2.4. With the same amount of data, the calculation of the physically informed loss involves solving the wave equation 25,000 times in each gradient descent step which was not possible with our current resources and we had to reduce the problem in order to create test scenarios. The first interesting result was that for a much smaller case, specifically with 100 time steps instead of 1000 and for both $\alpha = 0$ and $\alpha = 1$ we obtained the same IOU score of 40%. We also showed that for $\alpha = 0.5$ the model was able to train and converge but not in a sufficient time in order to reach the saturation - we cannot yet quantify the contribution of the physically informed loss but we do expect a contribution.

Examples of the segmentation maps the network produced (before and after setting a threshold for the probabilities) are given in Figure 5. We observe that even non-convex obstacles were inferred correctly. We remark that images with no obstacles were also generated and located correctly.

4. Conclusion

We presented a new approach for recovering a scatterer in an underwater medium. We solve the acoustic wave equation numerically to create a data-set. We then trained several neural networks with different architectures and parameters where the goal is to retrieve both the location and shape of the unknown scatterer. The method we propose

does not need a parametrization of the scatterer and can recover the location and shape of complex geometries. After training the model, the inference takes milliseconds and the model can be used for real-time applications.

We also presented a method to use the solution of the wave equation embedded in the neural network and create a physically informed process. We penalize the model using the physical solution of the wave equation to get more accurate and robust results. The extension of the model to the three dimensional case is currently being investigated.

References

[1] U. Albocher, A.A. Oberai, P.E. Barbone and I. Harari, *Adjoint-weighted equation for inverse problems of incompressible plane-stress elasticity*, Comput. Methods Appl. Mech. Engrg. 198(3032)24122420, 2009.

[2] R.V. Allen, *Automatic earthquake recognition and timing from single traces*, Bulletin Seismological Society of America, 68(5), 15211532, 1978.

[3] M. Baer and U. Kradolfer, *An automatic phase picker for local and teleseismic events*, Bulletin of the Seismological Society of America, 77(4), 14371445, 1987.

[4] E. Amitt, D. Givoli and E. Turkel, *Combined Arrival-Time Imaging and Time Reversal for Scatterer Identification*, Computer Methods in Applied Mechanics and Engineering (CMAME) 313:279302, 2017.

[5] F. Assous, M. Kray, F. Nataf and E. Turkel, *Time Reversed Absorbing Condition: Application to Shape Reconstruction*, Inverse Problems 27(6), June 2011.

[6] P. E. Barbone, A. A. Oberai and I. Harari, *Adjoint-weighted variational formulation for direct solution of inverse heat conduction problem*, Inverse Problems, 23:23252342, 2007.

[7] C. Bardos and M. Fink, *Mathematical foundations of the time reversal mirror*, Asymptotic Anal. 29 (2002) 157182.

[8] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, *Automatic Differentiation in Machine Learning: a Survey*, Journal of Machine Learning Research, 18: 143, (2018).

[9] P. Blomgren, G. Papanicolaou and H. Zhao, *Super-resolution in time-reversal acoustics*, J. Acoust. Soc. Am. 111 (2002) 230248.

[10] F. Chollet, *Keras*, `https://keras.io/`

[11] D. Colton, R. Ewing and W. Rundell, *Inverse Problems in Partial Differential Equations* (SIAM, 1990). Applied Mathematical Sciences, Vol. 93 (Springer, New York, 2013), pp. xiv+405.

[12] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, 3rd edn. Applied Mathematical Sciences, Vol. 93 (Springer, New York, 2013), pp. xiv+405.

[13] M. Fink, F. Wu, D. Cassereau and R. Mallart, *Imaging through inhomogeneous media using time reversal mirrors*, Ultrason. Imaging 13 (1991) 179199.

[14] D. Givoli and E. Turkel, *Time reversal with partial information for wave refocusing and scatterer identification*, Comput. Methods Appl. Mech. Eng. 213 (2012) 223242.

[15] V. Isakov, *Inverse Problems for Partial Differential Equations*, Applied Mathematical Sciences, Vol. 127, 2nd edn. (Springer, 2006).

[16] A. Kahana, E. Turkel and D. Givoli, *Convective Wave Equation and Time Reversal Process for Source Refocusing*, J. Comput. Acoust. 26(1850016), No.02, (2018).

[17] D. P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980, (2014).

[18] L. Kuperkoch, T. Meier, J. Lee, J., and W. Friederich, *Automated determination of P-phase arrival times at regional and local distances using higher order statistics*, Geophysical Journal International, 181(2), 11591170, 2010.

[19] I. Levi, E. Turkel and D. Givoli, *Time reversal for elastic wave refocusing and scatterer location recovery*, J. Comput. Acoust. 23(1450013) (2015) 129.

[20] T. Levin, E. Turkel and D. Givoli *Obstacle Identification using the TRAC Algorithm*, International Journal for Numerical Methods in Engineering 118(2):61-92, 2019.

[21] H. Niu, E. Reeves and Peter Gerstoft, *Source localization in an ocean waveguide using supervised machine learning* The Journal of the Acoustical Society of America, 142, 1176 (2017).

[22] M. Raissi, P. Perdikaris, G.E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations* , J. Comput. Phys. 378(686-707), (2019).

[23] C.D. Saragiotis, L.J. Hadjileontiadis and S.M. Panas, S. M. *PAI-S/K: A robust automatic seismic Pphase arrival identification scheme*, IEEE Transactions on Geoscience and Remote Sensing, 40(6), 13951404, 2002.

[24] R. Seidl, R. Ernst, *Iterative time reversal based flaw identification*, Computers Mathematics Applications 72, 879-892, 2016.

[25] R. Sleeman, and T. Van Eck, T., *Robust automatic P-phase picking: An on-line implementation in the analysis of broadband seismogram recordings*, Physics of the Earth and Planetary Interiors, 113, 265275, 1999.

[26] A. Tarantola, *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation* (Elsevier, Amsterdam and New York, 1987), pp. xvi+613.

[27] C. R. Vogel, *Computational Methods for Inverse Problems*, Frontiers in Applied Mathematics, Vol. 23 (SIAM, Philadelphia, PA, 2002), pp. xvi+183.

[28] J. Wang, Z. Xiao, C. Liu, D. Zhao and Z Yao *Deep Learning for Picking Seismic Arrival Times*, JGR Solid Earth 124(7) 6612-6624, 2019.

[29] W. Zhu and G. C. Beroza, *A Deep-Neural-Network-Based Seismic Arrival Time Picking Method*, submitted to Geophys. J. Int.