AI-based approach to numerical PDEs

Journal of Computational Physics 378 (2019) 686–707

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M. Raissi [a], P. Perdikaris [b,*], G.E. Karniadakis [a]

[a] *Division of Applied Mathematics, Brown University, Providence, RI, 02912, USA*
[b] *Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA, 19104, USA*

Physics-informed Neural Network

Example: Solving a nonlinear Schrödinger equation with periodic boundary conditions

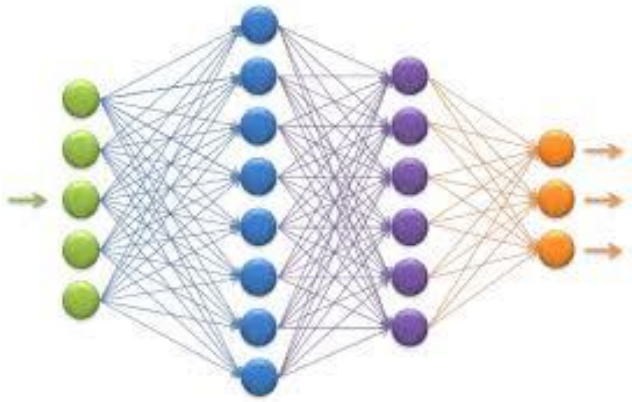
$$ih_t + 0.5h_{xx} + |h|^2 h = 0\ , \qquad x \in [-5,5],\ \ t \in [0, \pi/2],$$

$$h(x,0) = 2\,\mathrm{sech}(x), \qquad h(-5,t) = h(5,t),\ \ h_x(-5,t) = h_x(5,t).$$

Input to the network – any set of values (x,t),

Network - 5 FC hidden layers, 100 neurons per layer,

Output layer $\tilde{h}(x,t) = [\tilde{u}(x,t), \tilde{v}(x,t)]$.



Training loss function I

Define $\tilde{f} := i\tilde{h}_t + 0.5\tilde{h}_{xx} + \left|\tilde{h}\right|^2 \tilde{h}$

Physics-informed loss - $N_f = 20{,}000$ randomly selected collocation points (x_i, t_i) inside the domain

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\tilde{f}\left(x_i, t_i\right)\right|^2.$$

Loss at t=0 - $N_0 = 50$ points $\left\{x_j\right\}$, $-5 < x_j < 5$,

$$MSE_0 = \frac{1}{N_0}\sum_{j=1}^{N_0}\left|\tilde{h}\left(x_j, 0\right) - h\left(x_j, 0\right)\right|^2$$

Training loss function II

Boundary loss - $N_b = 50$ time samples $\{t_k\}$

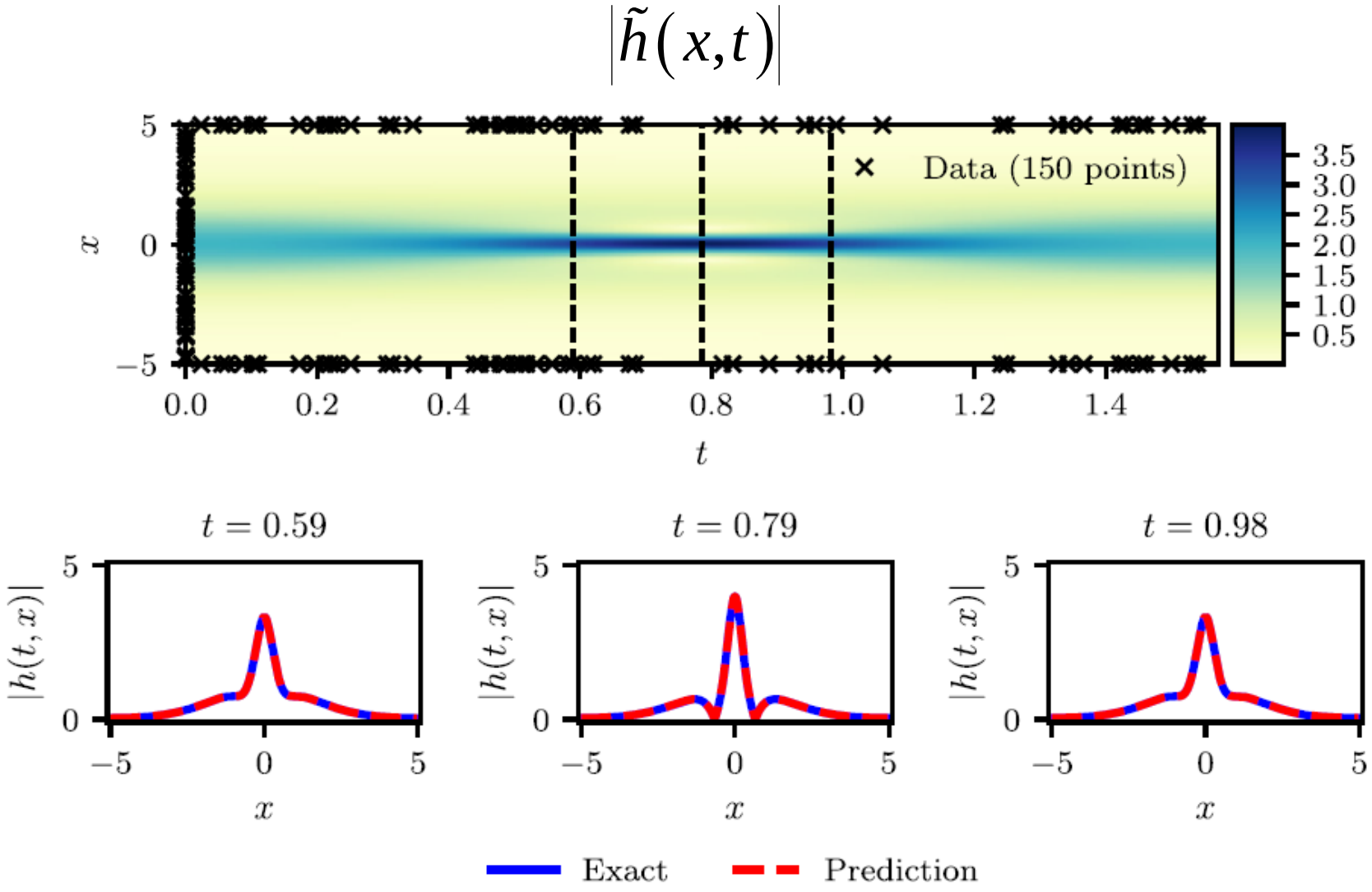
$$MSE_b = \frac{1}{N_b}\sum_{k=1}^{N_b}\left\{\left|\tilde{h}(-5,t_k)-\tilde{h}(5,t_k)\right|^2+\left|\tilde{h}_x(-5,t_k)-\tilde{h}_x(5,t_k)\right|^2\right\}$$

The network is trained with aggregated loss

$$MSE = MSE_f + MSE_0 + MSE_b$$

Notice that MSE_f requires "automatic differentiation" of the neural network by the parameters x,t. This is not trivial, but supported in platforms such as TensorFlow.

Physics-informed Neural Network



Physics-informed NN

"We must note however that the proposed methods should not be viewed as replacements of classical numerical methods for solving partial differential equations (e.g., finite elements, spectral methods, etc.). Such methods have matured over the last 50 years and, in many cases, meet the robustness and computational efficiency standards required in practice. Our message … is that classical methods … can coexist in harmony with deep neural networks"

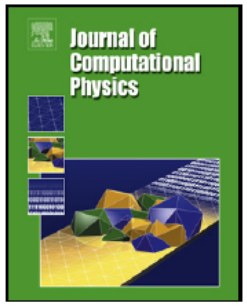
M. Raissia, P.Perdikarisb and G.E.Karniadakisa, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707

Journal of Computational Physics 413 (2020) 109458

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

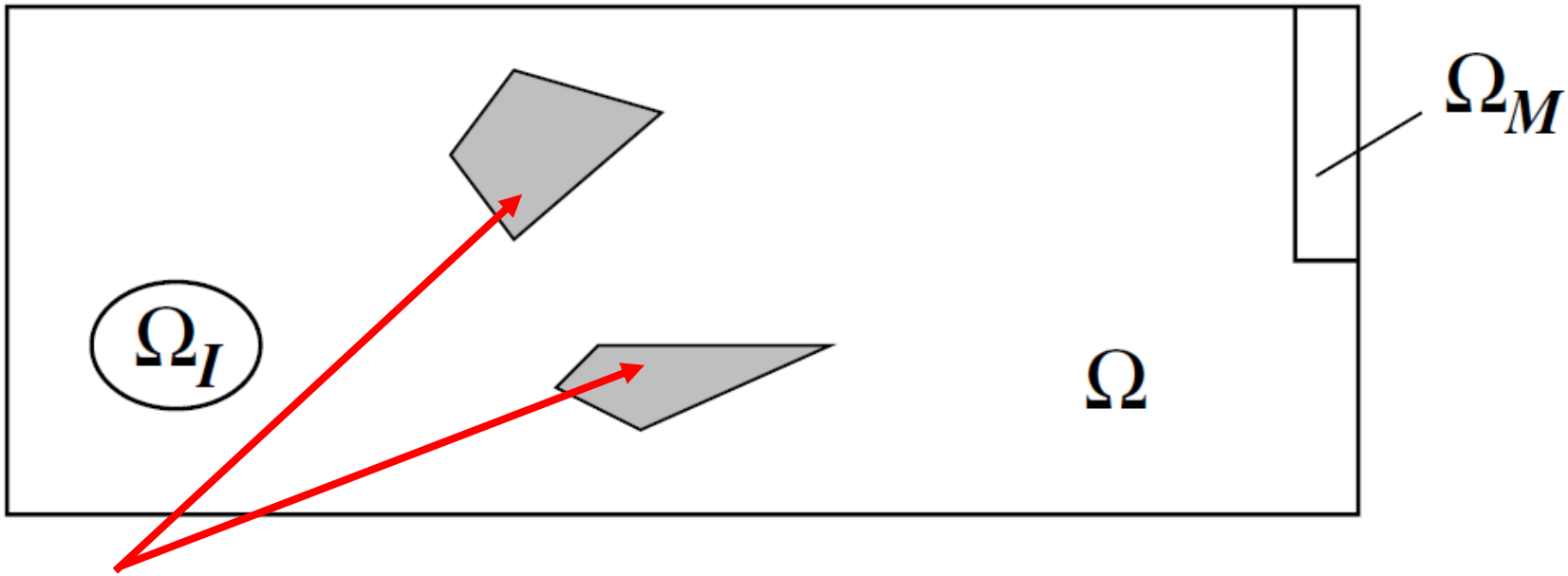


Obstacle segmentation based on the wave equation and deep learning

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*

Numerical methods for the wave equation



Obstacles (scatters)

Ω_M - Sensor domain

Ω_I - Source location

Basic notation

$$u(x,t):\Omega\times[0,T]\to\mathbb{R},\quad \Omega\subset\mathbb{R}^n,\qquad \nabla u:=\nabla_x u=\left(\frac{\partial u}{\partial x_1},\ldots,\frac{\partial u}{\partial x_n}\right)$$

Vector field $$F(x)=\left(F_1(x),\ldots,F_n(x)\right):\mathbb{R}^n\to\mathbb{R}^n$$

Divergence $$div(F)=\langle\nabla,F\rangle:=\sum_{i=1}^{n}\frac{\partial F_i}{\partial x_i}$$

Laplace operator $$\Delta u:=div(\nabla u)=\sum_{i=1}^{n}\frac{\partial^2 u}{\partial x_i^2},$$

Discrete Laplace as convolution $(n=2)$, $$\Delta_d=\begin{pmatrix}0 & 1/2 & 0\\ 1/2 & -2 & 1/2\\ 0 & 1/2 & 0\end{pmatrix}$$

Numerical methods for the wave equation

$$\frac{\partial^2 u}{\partial t^2}(x,t) = div\left(c^2(x)\nabla u(x,t)\right),\ 0 \le t \le T,\ x \in \Omega.$$

$$u = 0,\quad x \in \partial\Omega, \qquad u(x,0) = u_0(x),\ \frac{\partial u}{\partial t}(x,0) = v_0(x).$$

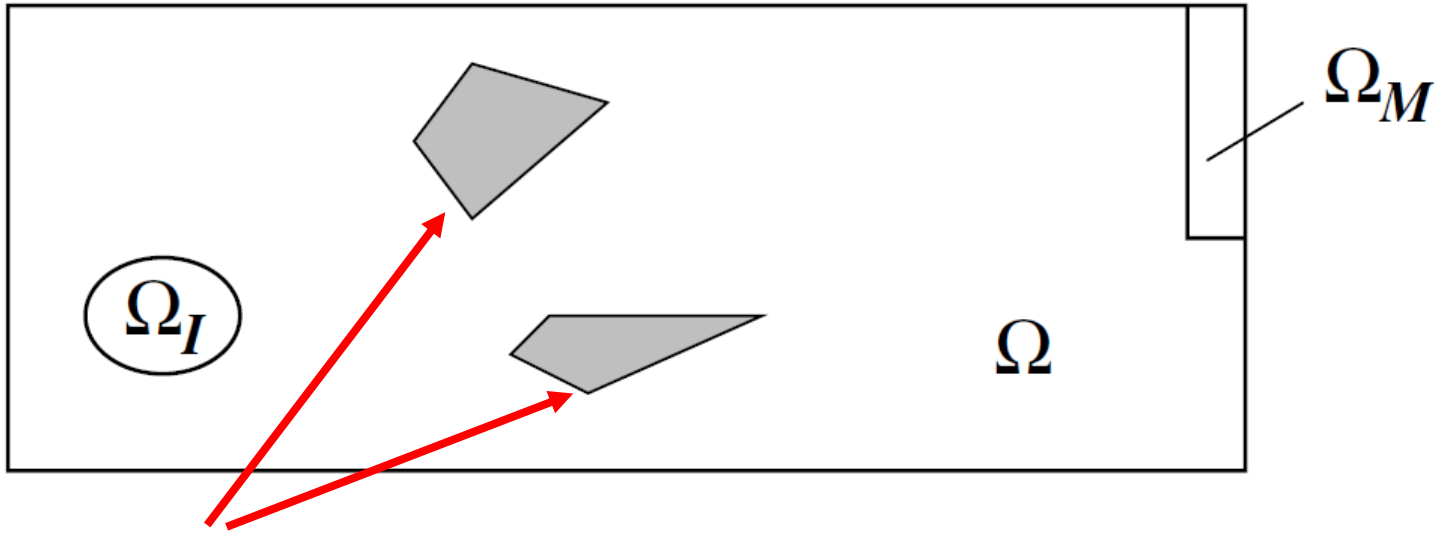
The source support $\Omega_I \subset \Omega$ can be represented through b.c:

$$u(x,0) = \chi_{\Omega_I}(x)u_0(x),\quad \frac{\partial u}{\partial t}(x,0) = \chi_{\Omega_I}(x)v_0(x).$$

Finite difference scheme (c constant, $\Delta_d(x, j)$ the discrete Laplace, $\Delta t = \Delta x = 1$)

$$u(x, j+1) = c^2\Delta_d u(x, j) + 2u(x, j) - u(x, j-1).$$

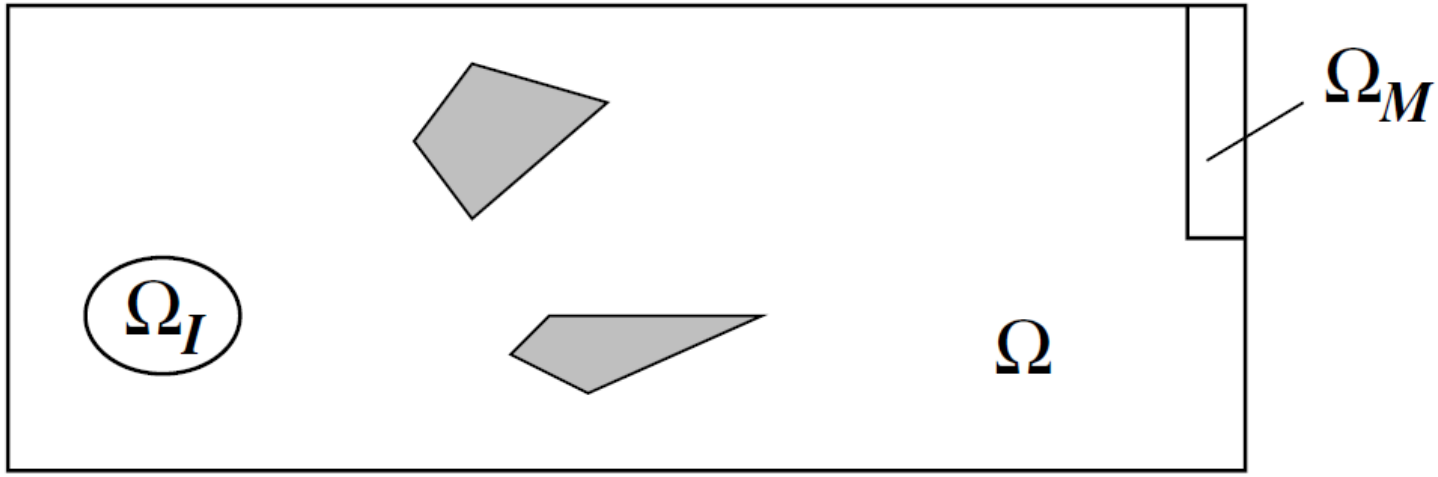
Source location through time reversal re-focusing



Input: Obstacle locations, $u(x,T), u_t(x,T),\ x \in \Omega_M$.

Goal: find Ω_I, the unknown source location

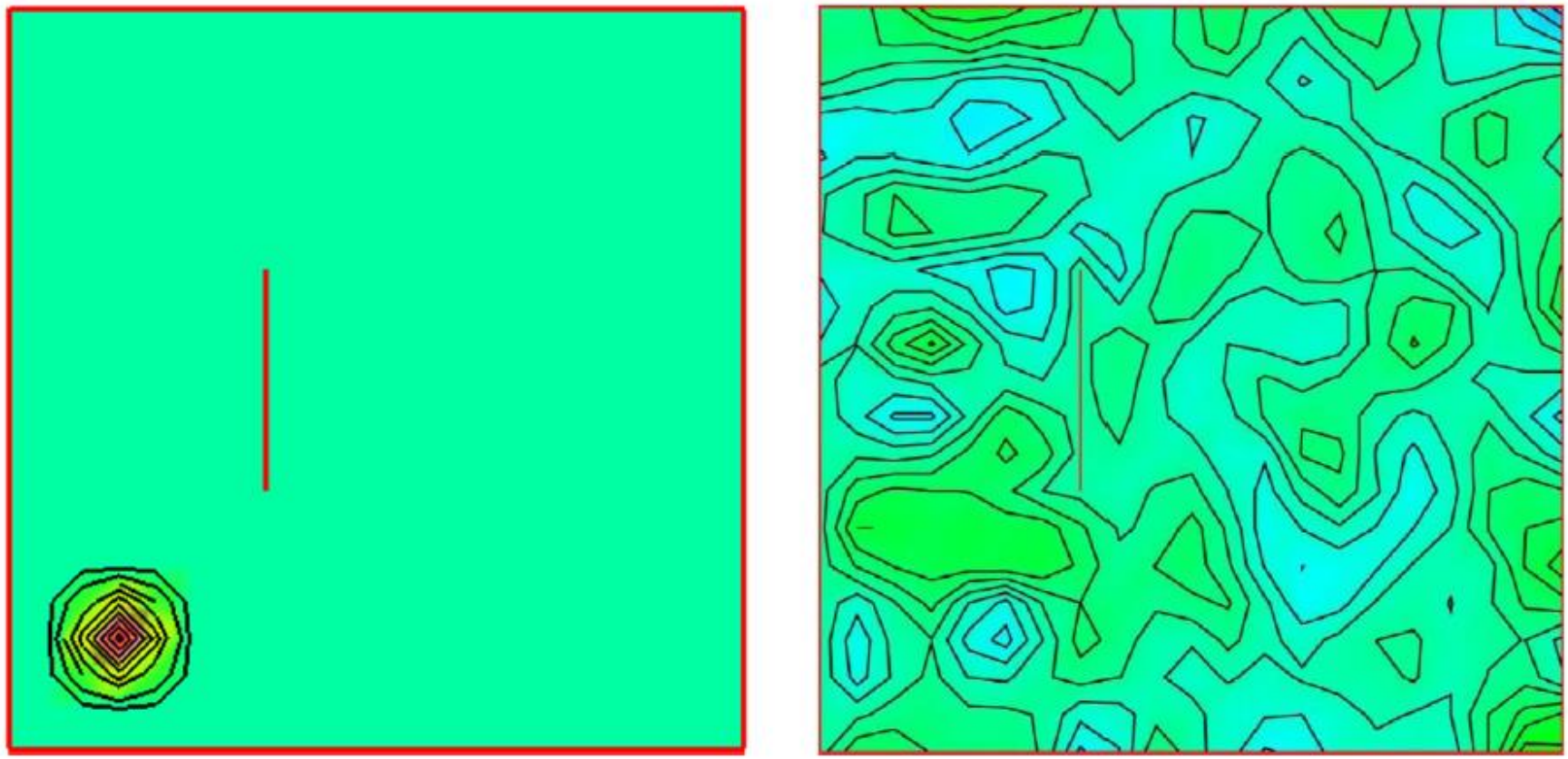
Source location through time reversal re-focusing



Method (*): Reversed finite difference scheme for the wave equation backwards from time T to time 0 based only on information from Ω_M at time T. This means initializing $u(x,T)=0$ for $x \in \Omega \setminus \Omega_M$.

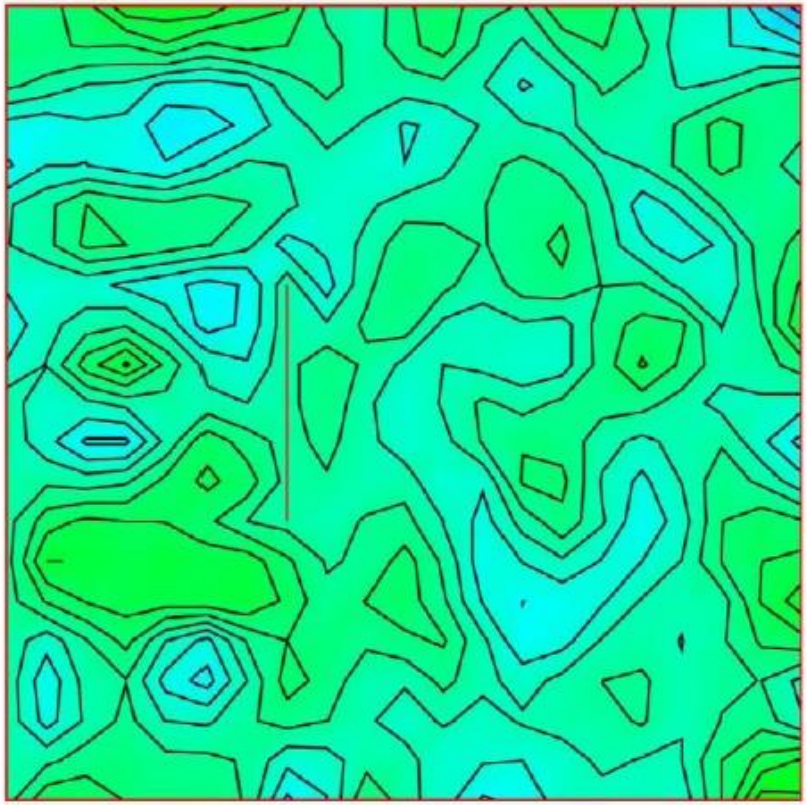
(*) D. Givoli, Time Reversal as a Computational Tool in Acoustics and Elastodynamics, Journal of Computational Acoustics, 22 (2014).

Example for forward propagation

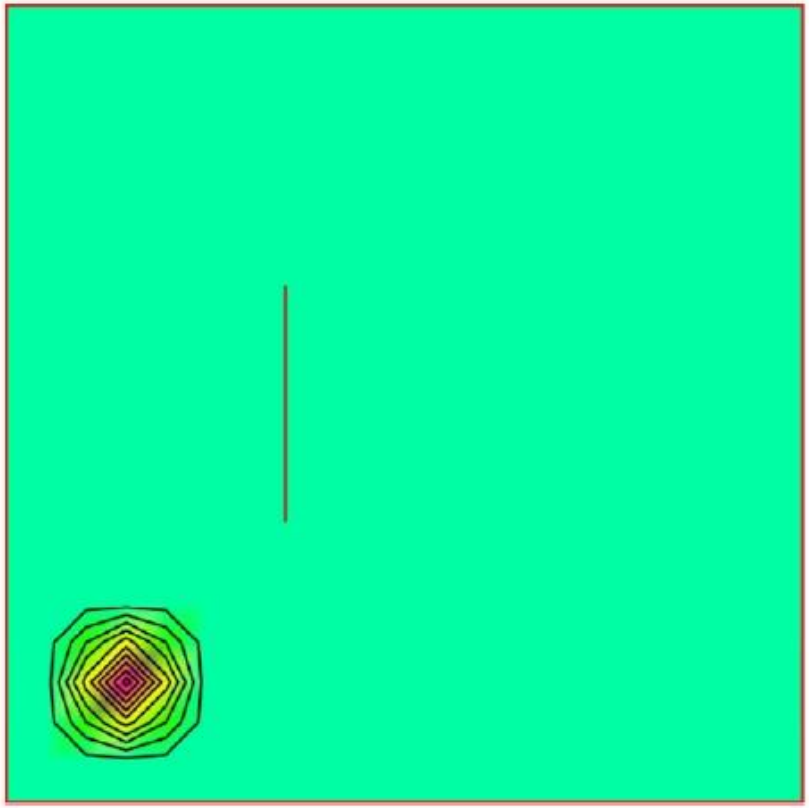


(a) (b)

Example for time reversal with $\Omega_M = \Omega$

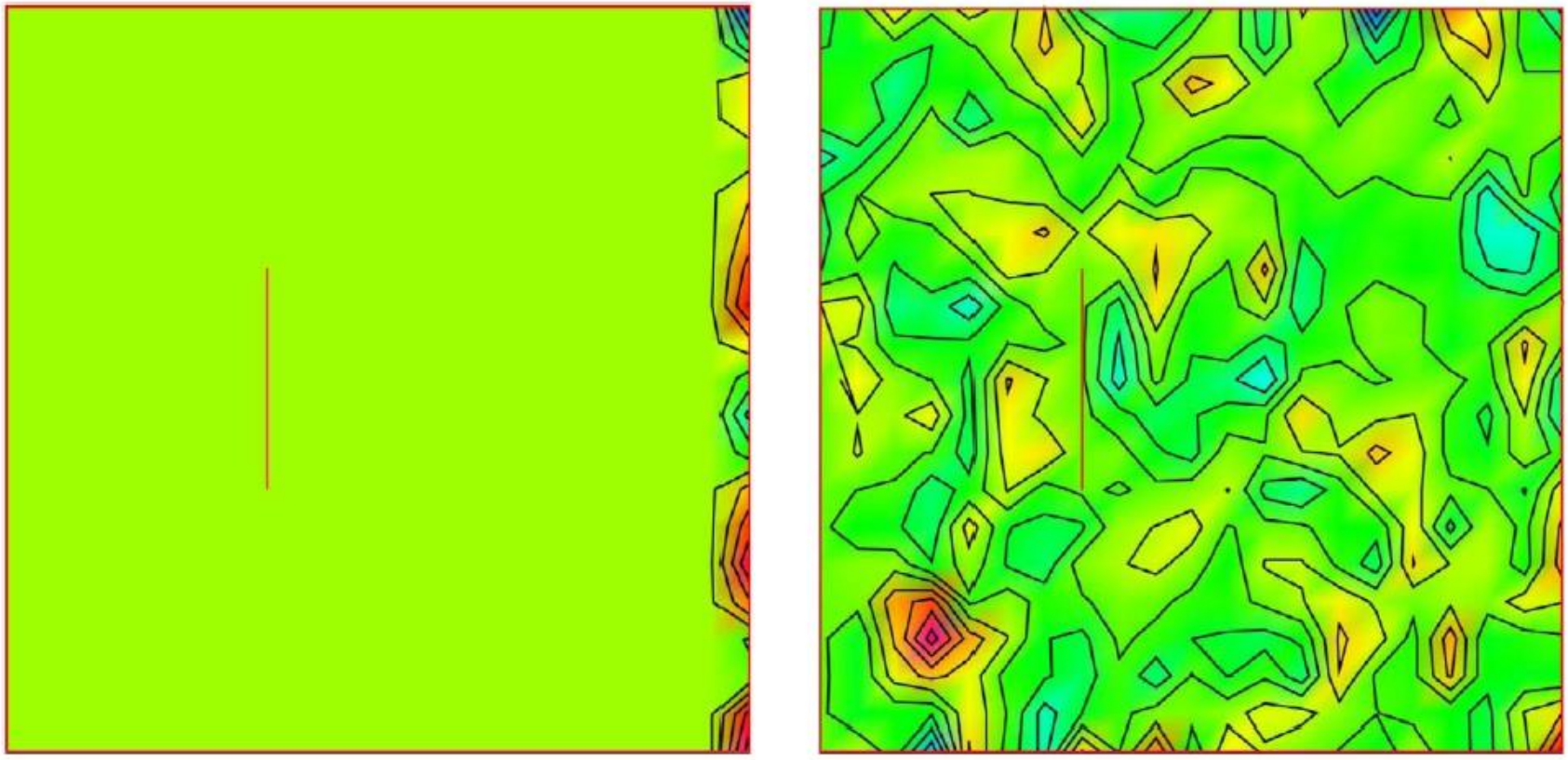


(a)



(b)

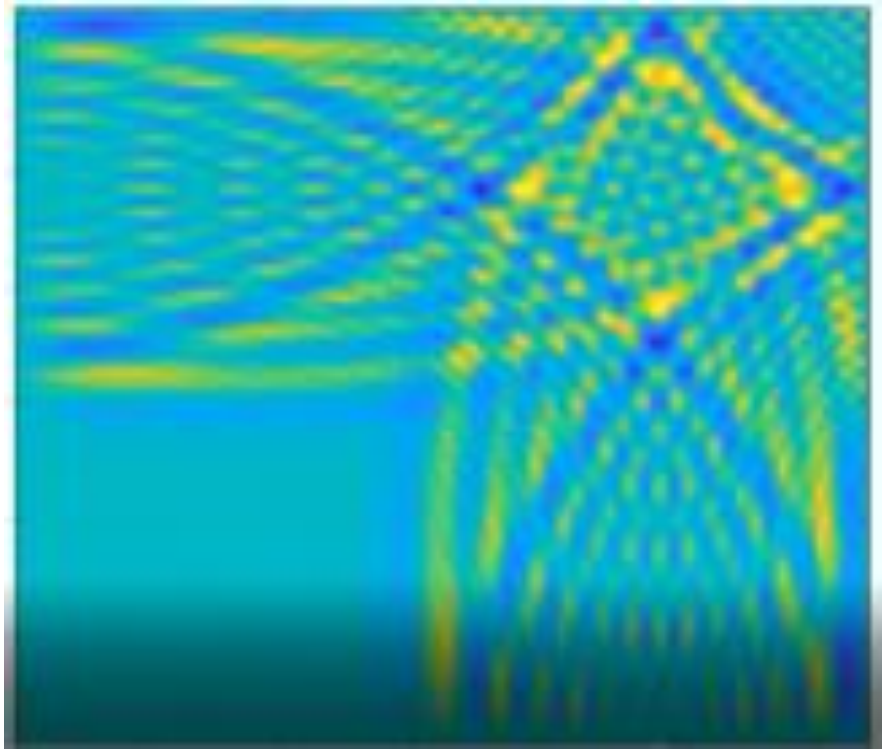
Example for time reversal with $\Omega_M \subset \Omega$



(a) (b)

Source location - AI-based toy problem

Input – only $u(x,T)\ldots u_t(x,T)$ is missing.



Output - $u(x,0)$ Source location at time 0.

Source location - AI-based toy problem

$u_t(x,T)$ is missing → problem is relatively 'ill-posed'.

Method – Train a regression convolutional network using software simulations that outputs (x, y) coordinates of predicted source.

Training set – 3,000 pairs of source locations and wave images at time T

Testing set – 1,000 pairs

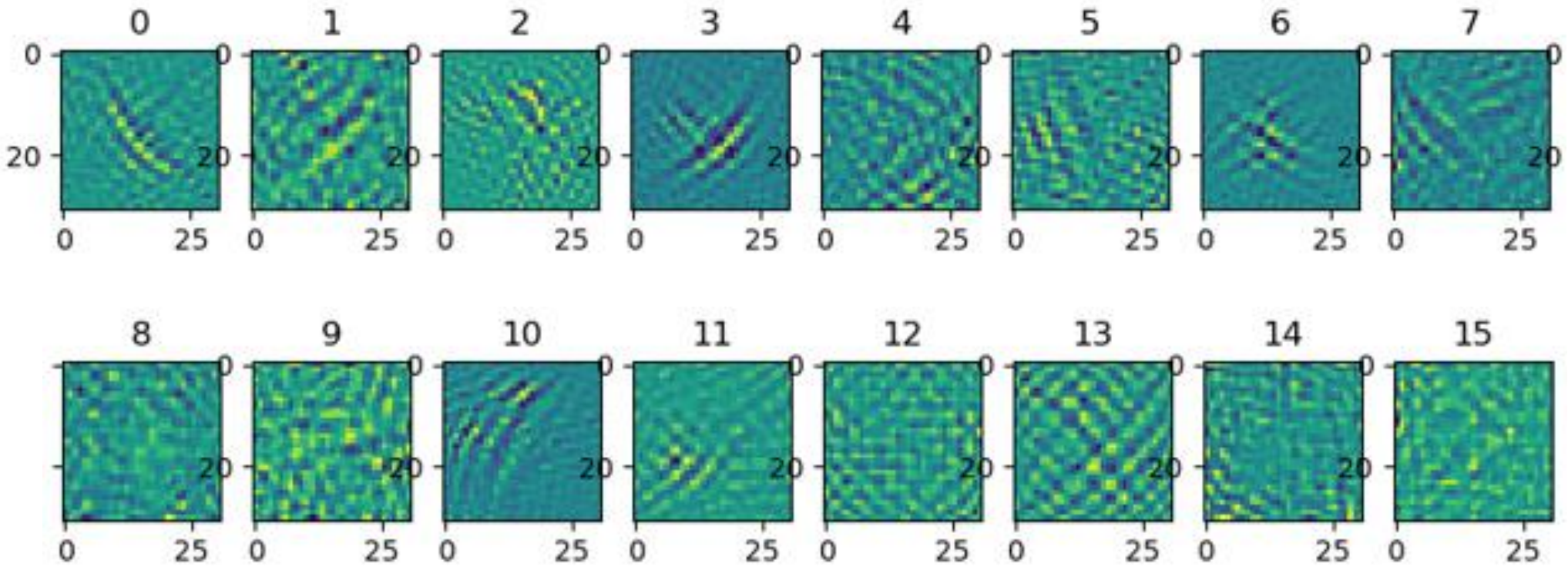
4-fold runs

Source location – Convolutional Neural Network architecture

Layer 0 (input)	128x128 image of $u(x,T)$
Layer 0 to layer 1	16 conv filters of size 31x31
Layer 1 to layer 2	32 conv filters of size 5x5
Layer 2 to layer 3	32 conv filters of size 31x31
Layer 3 to output	Fully connected
Output	predicted (x,y) coordinates of source.

Source location – trained conv filters

Visualization of the convolution filters at layer 1



Source location – results (4-fold)

Pixels range	Success rate
Exact location	36%
Up to one pixel away	72%
Up to two pixels away	87%
Up to three pixel away	93%

Obstacle segmentation

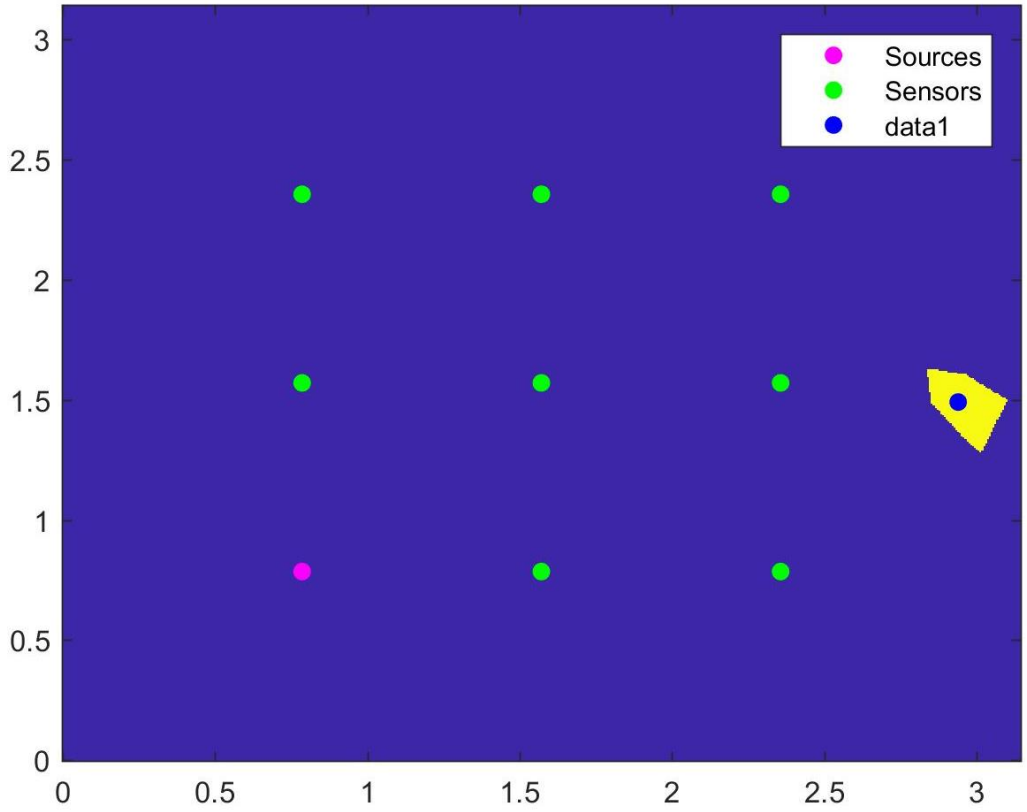
Input - sensor data $0 = t_0 < t_1 < \cdots < t_m = T$,

- source location
- boundary conditions

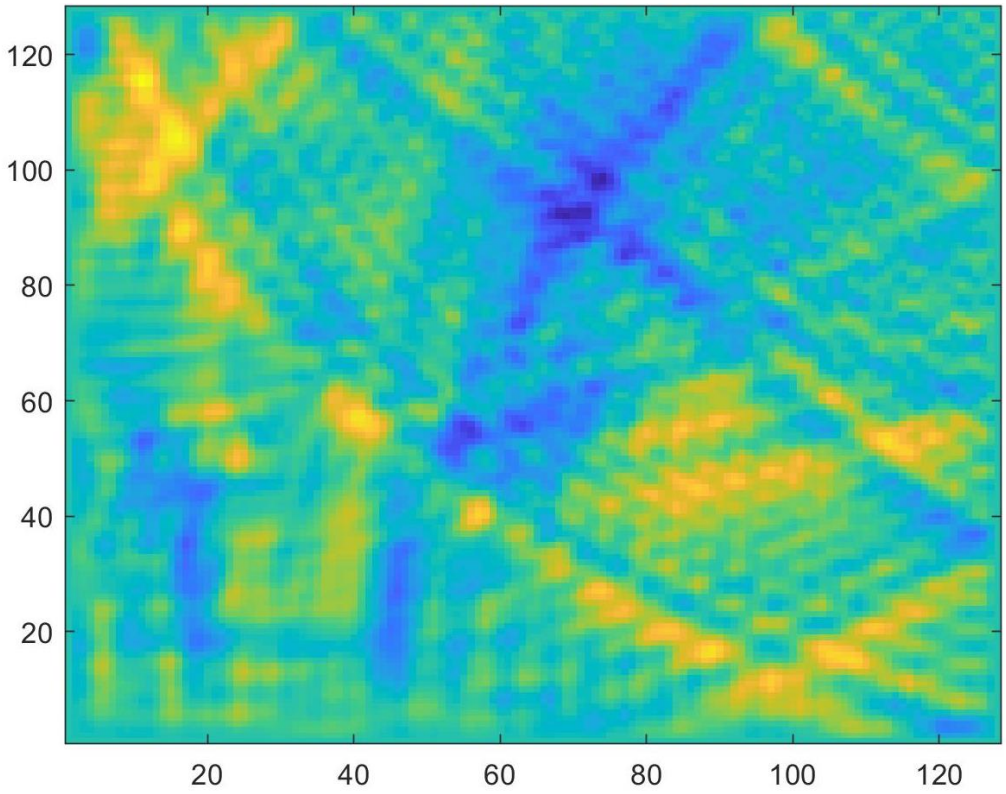
Output - obstacle segmentation map

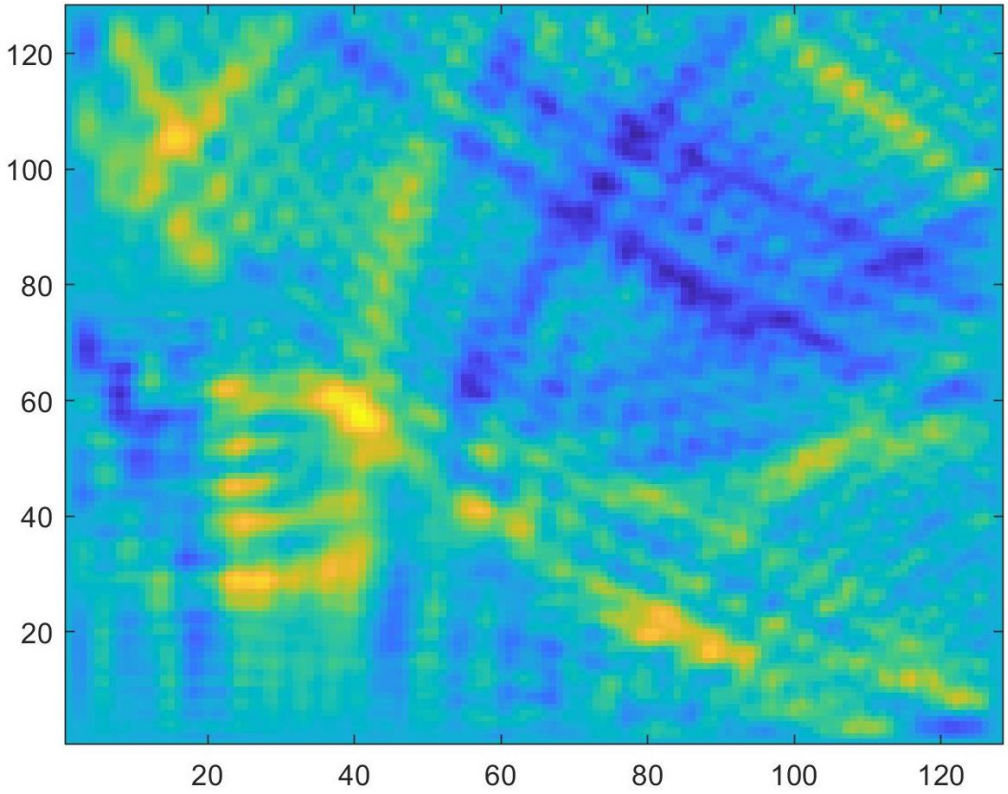
Current state-of-the-art using numerical methods + optimizations: <u>unknown single circular obstacle</u>

Here we assume $c = const$ away from boundary and obstacles.



Waves at t = 500 with different obstacles

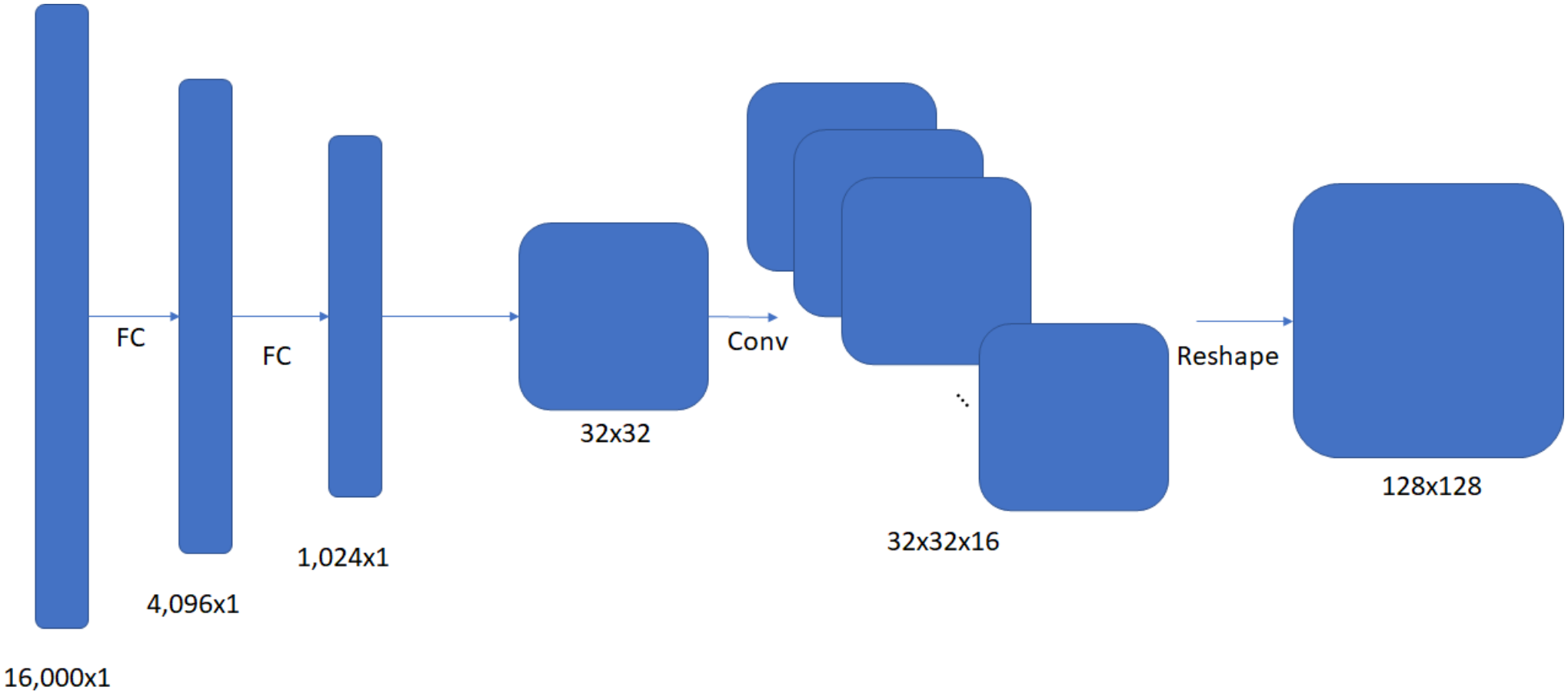




Obstacle segmentation – network architecture

Input: 2,000 time samples for each of the 8 sensors

Output: segmentation map of obstacle. Each pixel is logistic regression variable with value [0,1]



Obstacle segmentation – Training loss

Dataset of 20,000 random polygonal obstacles.

$p_I(x) \in \{0,1\}$- ground truth. $p_I(x) = 1$ if location x is part of the obstacle in sample I.

$s_I \in \mathbb{R}^n$ - vector representation of sample I at the layer before last,

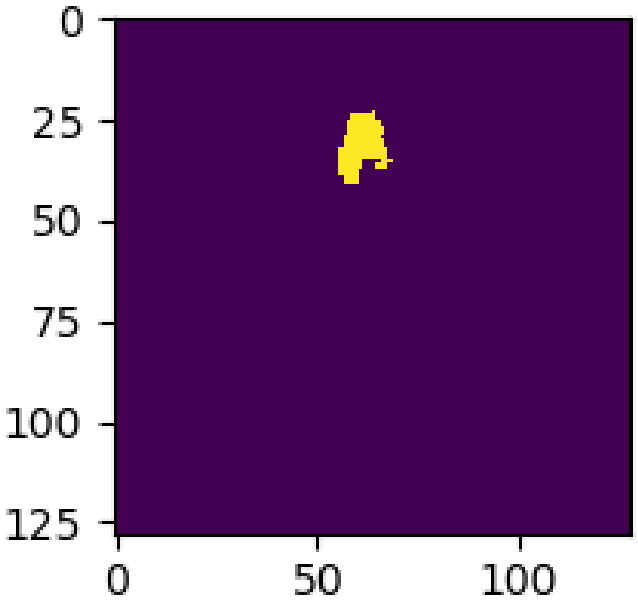
$$\hat{p}_I(x) = \frac{1}{1+e^{-(\langle W_x, s_I \rangle + b_x)}}, \quad W_x \in \mathbb{R}^n, \; b_x \in \mathbb{R}, \; x \in [0,127]^2.$$

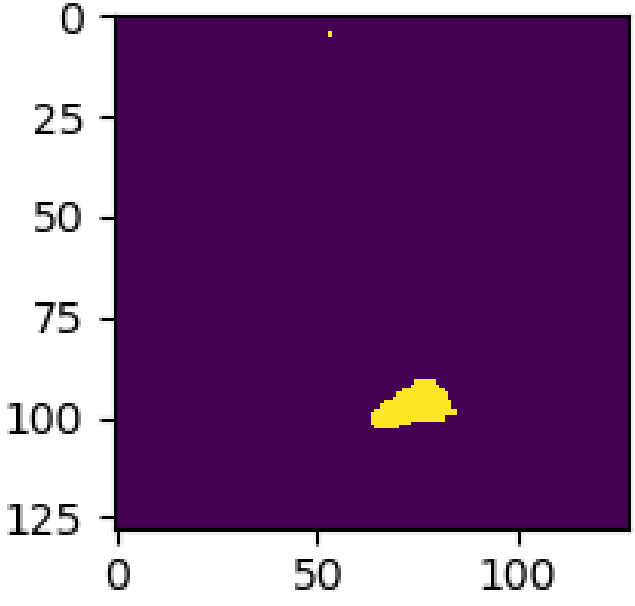
Minimization of Negative Log-Likelihood loss

$$NLL = -\frac{1}{\#I}\frac{1}{128^2}\sum_I \sum_{x \in [0,127]^2} p_I(x)\log \hat{p}_I(x) + \left(1 - p_I(x)\right)\log\left(1 - \hat{p}_I(x)\right)$$

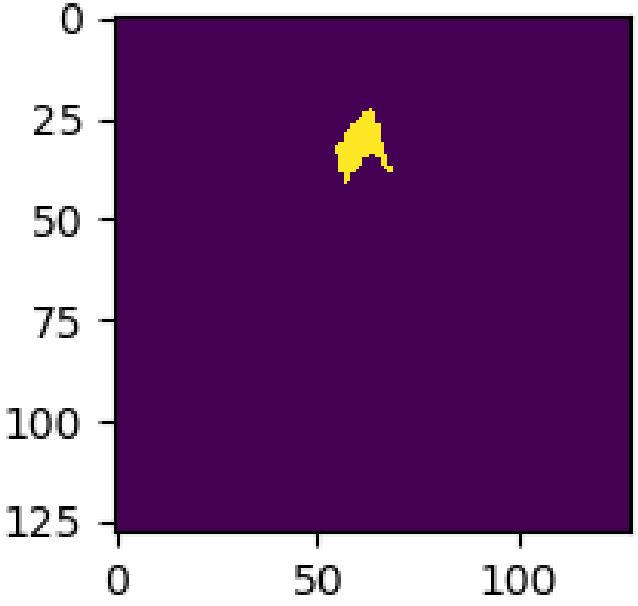
Obstacle segmentation – results

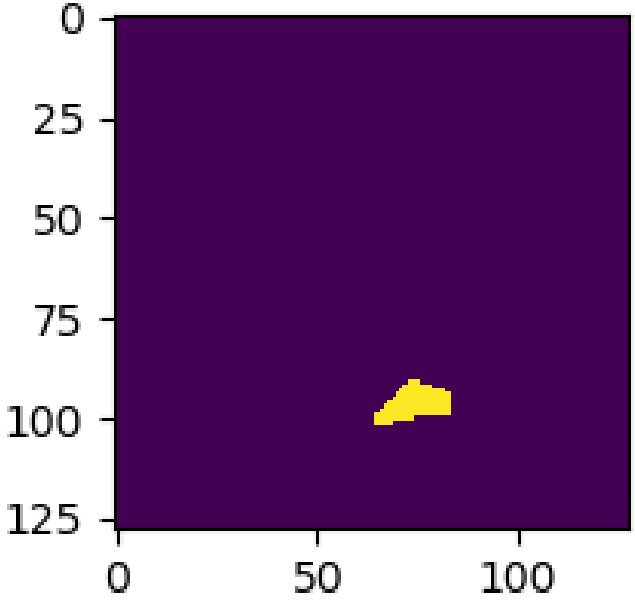
Prediction





Ground Truth





Obstacle segmentation – results

$$0 \leq IOU(A,B) := \frac{|A \cap B|}{|A \cup B|} \leq 1$$

Mean IOU	Median IOU
0.621	**0.66**

Physics-informed obstacle segmentation

- Regularization of the solution using the wave equation.
- The known source, sensors and output obstacle image $\{\hat{p}(x)\}$, $0 \le \hat{p}(x) \le 1$, provide a solution $\hat{u}$ (function of $\hat{p}$) to

$$\frac{\partial^2 \hat{u}}{\partial t^2}(x,t) = c^2 div\left(\left(1-\hat{p}(x)\right)^2 \nabla \hat{u}(x,t)\right),\ 0 \le t \le T,\ x \in [0,127]^2.$$

- So, if the sensors are located at $\{x_k\}_{k=1}^{8}$, then

$$MSE := \frac{1}{8}\frac{1}{T}\sum_{k=1}^{8}\sum_{j=1}^{T}\left(u(x_k, j) - \hat{u}(x_k, j)\right)^2 .$$

- Training loss - linear combination of NLL and physics informed MSE.

Physics-informed obstacle segmentation

- $\{\hat{u}(x_k, j)\}$ are a determined by obstacle map $\hat{p}(x)$.
- But…how do we pass this information to the loss function?
- We add to the architecture of the neural network, one additional layer per time stamp, $\hat{u}(x, j), 1 \leq j \leq T$.
- We apply finite difference forward passes

$$\hat{u}(x, j+1) = F(x, c, \hat{p}, \hat{u}(\cdot, j), \hat{u}(\cdot, j-1)).$$

- We collect from each layer j the values $\{\hat{u}(x_k, j)\}_{k=1}^{8}$ and pass to the loss function.