MESH-FREE UNSUPERVISED LEARNING-BASED PDE SOLVER OF FORWARD AND INVERSE PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a novel neural network-based partial differential equations solver for forward and inverse problems. The solver is grid free, mesh free and shape free, and the solution is approximated by a neural network. We employ an unsupervised approach such that the input to the network is a points set in an arbitrary domain, and the output is the set of the corresponding function values. The network is trained to minimize deviations of the learned function from the PDE solution and satisfy the boundary conditions. The resulting solution in turn is an explicit smooth differentiable function with a known analytical form.

Unlike other numerical methods such as finite differences and finite elements, the derivatives of the desired function can be analytically calculated to any order. This framework therefore, enables the solution of high order non-linear PDEs. The proposed algorithm is a unified formulation of both forward and inverse problems where the optimized loss function consists of few elements: fidelity terms of L_2 and L_∞ norms, boundary conditions constraints and additional regularizers. This setting is flexible in the sense that regularizers can be tailored to specific problems. We demonstrate our method on a free shape 2D second order elliptical system with application to Electrical Impedance Tomography (EIT).

1 INTRODUCTION

Partial differential equations are fundamental in science and mathematics with wide applications in medical imaging, signal processing, computer vision, remote sensing, electromagnetism and more. Classical methods such as finite differences, finite volume and finite elements are numerical discretization-based methods where the domain is divided into a uniform grid or polygon mesh. The differential equation is then reduced to a system of algebraic equations. These methods may have some limitations: the solution is numeric and may suffer from high condition number, highly dependent on the discretization and even the second derivative is sensitive to noise.

In the last few years, deep learning and neural network-based algorithms are extensively used in pattern recognition, image processing, computer vision and more. Recently, the deep learning approach had been adopted to the field of PDEs as well by converting the problem into a machine learning one. In *Supervised learning*, the network maps an input to an output based on example input-output pairs. This strategy is used in inverse problems, where the input to the network is a set of observations/measurements (e.g. x-ray tomography, ultrasound) and the output is the set of parameters of interest (tissue density etc.) Feigin et al. (2018); Lucas et al. (2018); McCann et al. (2017); Seo et al. (2019). *Unsupervised learning* on the other hand is a self-learning mechanism where the natural structure presents within a set of data points is inferred.

Algorithms for forward and inverse problems in partial differential equations via unsupervised learning were recently introduced. The *indirect* approach utilizes a neural network as a component in the solution. Li et al. (2018) for example, proposed the NETT (Network Tikhonov) approach to inverse problems. NETT considers regularized solutions having a small value of a regularizer defined by a trained neural network. Khoo & Ying (2018) introduced a novel neural network architecture, Switch-Net, for solving the wave equation based inverse scattering problems via providing maps between the scatterers and the scattered field. Han et al. (2018) developed a deep learning-based approach that can handle general high-dimensional parabolic PDEs. To this end, the PDEs are reformulated using backward stochastic differential equations. The latter is solved by a temporal discretization

and the gradient of the unknown solution at each time step is approximated by neural network. Li et al. (2019) approximate the solution map of linear and nonlinear problems by a deep network.

Direct algorithms solve the forward problem PDEs by directly approximating the solution with a deep neural network. The network parameters are determined by the optimization of a cost function such that the optimal solution satisfies the PDE, boundary conditions and initial conditions. Chiaramonte & Kiener (2017) addressed the forward problem by constructing a one layer network which satisfies the PDE within the domain. The boundary conditions were analytically integrated in the cost function. They demonstrated their algorithm on the Laplace and hyperbolic conservation law PDEs. Sirignano & Spiliopoulos (2017) proposed a deep learning forward problem solver for high dimensional PDEs. Their algorithm was demonstrated on the American option free-boundary equation. Raissi et al. (2017) focused on continuous time models and solved the Burgers and Shrödinger equations.

In this work we address the forward and inverse PDE problems via a direct unsupervised method. Our key contributions are four fold: (1) inverse problems can be solved in the same framework as the forward problems. (2) In the forward part we extend the standard L_2-based fidelity term in the cost function by adding L_∞-like norm. Moreover, (3) some regularization terms which impose a-priori knowledge on the solution can be easily incorporated. (4) An important feature of our construction is the ability to handle free-form domain in a mesh free manner. We demonstrate our algorithm by a second order elliptic equation, in particular the Electrical Impedance Tomography (EIT) application on circular and three other arbitrary domains.

2 Mathematical Formulation

Let Ω be a bounded open and connected subset of $\mathbb{R}^d$, and $A = A(x) = (a^{ij}(x))$ be any given $d \times d$ symmetric positive definite matrix of functions for $1 \leq i, j \leq d$. Let $b = b(x) = (b^j(x))$ be any given n-tuple of functions and let $c = c(x)$ be any given function. A second order operator $\mathcal{L}$ is said to be in divergence form, if $\mathcal{L}$ acting on some u has the form

$$\mathcal{L}u = \partial_i(a^{ij}(x)\partial_j u) + b^j(x)\partial_j u + c(x)u, \quad i, j = 1, \ldots, d \tag{1}$$

where we use the Einstein summation convention. Consider the partial differential problem with Dirichlet boundary conditions

$$\begin{aligned} \mathcal{L}u &= 0, && x \in \Omega \\ u(x) &= u_0(x), && x \in \partial\Omega. \end{aligned} \tag{2}$$

The *forward problem* solves u given the coefficients $\theta := \{a^{ij}(x), b^j(x), c(x)\}$ while the *inverse problem* determines the coefficients set θ given u.

The proposed algorithm approximates the solutions in both problems by neural networks $u(x; w_u), a^{ij}(x; w_{ij}), b^j(x; w_b), c(x; w_c)$ such that the networks are parameterized by w_u, w_{ij}, w_b, w_c, and the input to the network is $x \in \mathbb{R}^d$. Figure 1 depicts a network architecture of u in $\mathbb{R}^2$. The network consists of few fully connected layers with *tanh* activation and linear sum in the last layer. The network is trained to satisfy the PDE with boundary conditions by minimizing a cost function. In the forward problem

$$\mathcal{F}(u) = \lambda\|\mathcal{L}u\|_2^2 + \mu\|\mathcal{L}u\|_\infty + \|u - u_0\|_{1,\partial\Omega} + \mathcal{R}^F(u), \tag{3}$$

and in the inverse problem

$$\mathcal{I}(\theta) = \lambda\|\mathcal{L}u\|_2^2 + \mu\|\mathcal{L}u\|_\infty + \|\theta - \theta_0\|_{1,\partial\Omega} + \mathcal{R}^I(\theta). \tag{4}$$

The first two terms enforce the solution to satisfy the equation. The first term minimizes the error in L_2 sense while the second term minimizes the maximal error. Ideally, the solution to the PDE has to be satisfied at every point x. This is considered as a *strong* solution. In the deep learning framework the optimization is carried out on mini batches of points. The first term minimizes the operator in the integral or average sense and is therefore insensitive to a point jump in the value of the integrand which yields a *weak* solution. This manifest itself by point discontinuities $u(x)$ that are seen empirically as well. The L_∞ term promotes a strong solution since it takes care of possible error points. The third term imposes boundary conditions where u_0 and θ_0 are the boundary values of $u(x)$ and $\theta(x)$. The last term is a regularizer which can be tailored to the application. There are

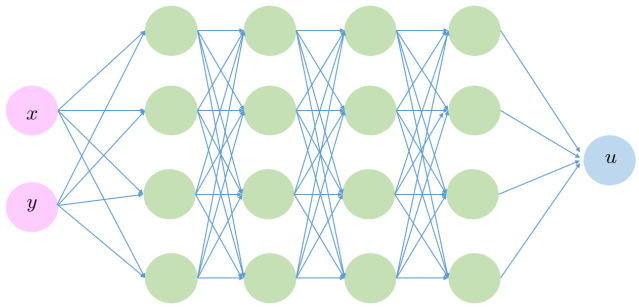


Figure 1: Network architecture: the point $(x, y) \in \mathbb{R}^2$ serves as an input and u as the output.

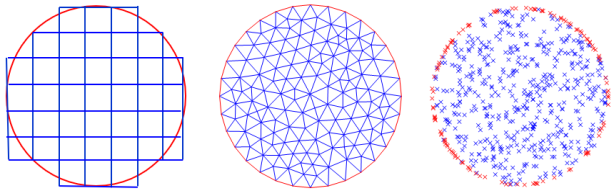


Figure 2: Left to right: finite differences grid, finite elements mesh and random points samples used in the proposed algorithm.

few advantages of this setting. First, the solutions are smooth analytic functions and are therefore *analytically differentiable*. It was recently proven that for such functions, e.g. Lipchitz with constant ν, the approximation of the function by a fully connected network Θ is given by

$$\|u(x) - \Theta\|_{L^p([0,1]^d)} \leq 40\nu\sqrt{d}N^{-2/d}L^{-2/d},$$

where $p \in [0, \infty)$, N number of neurons in a layer and L number of layers Shen et al. (2019). In addition, this framework enables setting a prior knowledge on the solution by designing the regularizers $\mathcal{R}^F$ and $\mathcal{R}^I$. Moreover, the training procedure is mesh free. Unlike finite differences or finite elements methods, we use *random* points in the domain and its boundary in the course of the optimization of equation 3 and equation 4, see Figure 2. This means that the solution does not depend upon a coordinate mesh and we can also define an arbitrary regular domain Ω.

3 Application to Electrical Impedance Tomography

Let us address a special case of equation 1,

$$\begin{aligned} &\nabla \cdot \Big(\sigma(x)\nabla u(x)\Big) = 0, && x \in \Omega \subset \mathbb{R}^2 \\ &u(x) = u_0(x), && x \in \partial\Omega. \end{aligned} \tag{5}$$

We assume that $0 < \sigma(x) \in C^1(\Omega)$, which guarantees existence and uniqueness of a solution $u \in C^2(\Omega)$ Evans (2010).

The elliptical system equation 5 was addressed by Siltanen et al. (2000) in the context of Electrical Impedance Tomography (EIT) which is a reconstruction method for the inverse conductivity problem. The function σ indicates the electrical conductivity density, and u is the electrical potential. An electrical current

$$\psi_{n,\varphi} = \sigma \frac{\partial u_n}{\partial \nu}\Big|_{\partial\Omega} = \frac{1}{\sqrt{2\pi}} \cos(n\kappa + \varphi), \;\; n \in \mathbb{Z}$$

is applied on electrodes on the surface $\partial\Omega$, where κ is the angle in polar coordinate system along the domain boundary, φ is the phase and ν is the normal unit. The resulting voltage $u|_{\partial\Omega} = u_0$

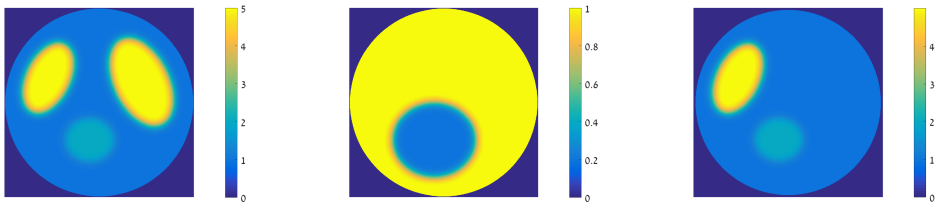


Figure 3: Left to right: phantom 1, phantom 2 and phantom 3

is measured through the electrodes. The conductivity σ is determined from the knowledge of the Dirichlet-to-Neumann map or voltage-to-current map

$$\Lambda_\gamma : u|_{\partial\Omega} \to \sigma \frac{\partial u_n}{\partial \nu}\Big|_{\partial_\Omega},$$

Mueller & Siltanen (2012); Alsaker & Mueller (2018); Bera et al. (2011); Fan & Ying (2019).

We demonstrate our framework by solving the forward and inverse problem of equation 5 which is a first step towards a full tomography. Following Mueller & Siltanen (2012), we simulate the voltage measurement $u|_{\partial\Omega}$ by the Finite Elements Method (FEM) given three variants of a conductivity phantom $\sigma(x)$ depicted in Figure 3. We calculate the FEM solution with different triangle mesh densities and select as ground truth the one such that finer meshes do not improve the numerical solution. With our suggested method, the forward problem determines the electrical potential u in the whole domain Ω given σ, while the inverse problem uses the approximated u and calculates the conductivity σ given that $\sigma|_{\partial\Omega} = \sigma_0$.

4 Forward Problem

In the forward problem the conductivity $\sigma(x_i)$ and boundary conditions $u_0(x_b)$ are given for random points set $\{x_i\} \in \Omega \subset \mathbb{R}^2$, $\{x_b\} \in \partial\Omega \subset \mathbb{R}^2$ with sets size of N_s and N_b respectively. A neural network having the architecture shown in Figure 1 approximates $u(x)$. Let

$$\mathcal{L}_i := \nabla \cdot \Big(\sigma(x_i)\nabla u(x_i)\Big). \tag{6}$$

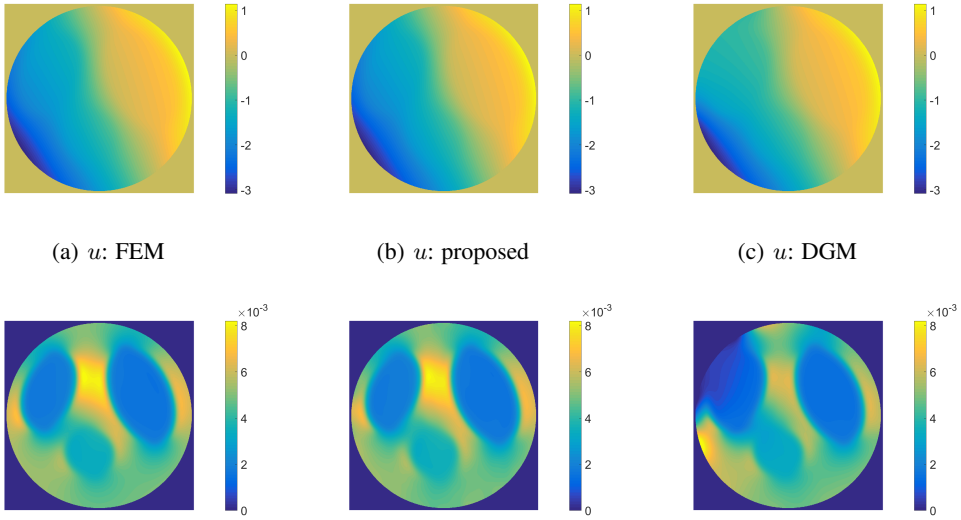
The cost function equation 3 is then rewritten as

$$\mathcal{F}\Big(u(x; w_u) = \frac{\lambda}{N_s}\sum_{i=1}^{N_s}|\mathcal{L}_i|^2 + \frac{\mu}{K}\sum_{k\in \text{top}_K(|\mathcal{L}_i|)}|\mathcal{L}_k| + \frac{1}{N_b}\sum_{b=1}^{N_b}\Big|u(x_b) - u_0(x_b)\Big| + \alpha\|w_u\|_2^2. \tag{7}$$

The first term is the L_2 norm of the differential operator, the second term is a relaxed version of the infinity norm where we take the mean value of the top-K values of $|\mathcal{L}_i|$. The third term imposes the boundary conditions and the last term serves as a regularizer of the network parameters.

The first phantom is shown in Figure 3 left. The impedance values associated with the background ellipses and circle were 1, 5 and 2 respectively. The original piecewise constant function σ was slightly smoothed by a Gaussian kernel.

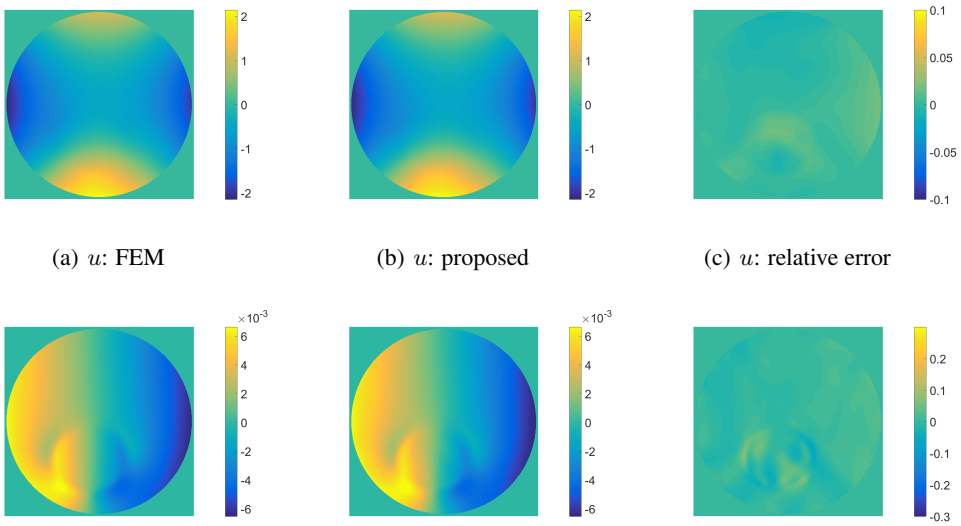
Figure 4 shows the forward problem results for current ψ with $n = 1$ and $\varphi = \pi/8$. The left column is the FEM solution which is referred to as ground truth, where the top row indicates the solution u and the bottom row the derivative of u with respect to x calculated as the finite difference approximation of the FEM result. The middle column depicts the outcome of the proposed method where $\partial u/\partial x$ is an analytical derivative of our result. The right column shows the outcome of the DGM method Sirignano & Spiliopoulos (2017) which is a special case of equation 3 with $\lambda = 1$ and $\mu = 0$. Quantitative results of the Mean square error (MSE) and PSNR are summarized in Table 1. Clearly, the proposed method outperforms the DGM method since the weighting parameters and the L_∞ norm play a significant role in the loss function.



(a) u: FEM (b) u: proposed (c) u: DGM

(d) $\partial u/\partial x$: FEM (e) $\partial u/\partial x$: proposed (f) $\partial u/\partial x$: DGM

Figure 4: Forward problem results of u and $\partial u/\partial x$ for $n = 1$, $\varphi = \pi/8$ given phantom 1. Left column: ground truth (FEM). Middle column: proposed method. Right column: DGM method. MSE and PSNR are noted in line 2 of Table 1



(a) u: FEM (b) u: proposed (c) u: relative error

(d) $\partial u/\partial x$: FEM (e) $\partial u/\partial x$: proposed (f) $\partial u/\partial x$: relative error

Figure 5: Forward problem results of u and $\partial u/\partial x$ for $n = 2$, $\varphi = \pi/2$ given phantom 2. Left column: ground truth (FEM). Middle column: proposed method. Right column: relative error. MSE and PSNR are reported in line 4 of Table 2

Table 1: Forward problem results for phantom 1 compared with the DGM method Sirignano & Spiliopoulos (2017)

	n	φ	u: GDM		u: Proposed		u_x: GDM		u_x: Proposed	
			MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR
1	1	0	2.50e-1	9.02	**2.38e-4**	**39.23**	6.01e-6	11.17	**7.11e-9**	**40.43**
2	1	$\pi/8$	4.20e-2	14.90	**1.13e-4**	**40.60**	7.95e-7	19.27	**5.92e-9**	**40.55**
3	1	$\pi/4$	5.46e-2	13.54	**1.23e-4**	**40.03**	1.87e-6	13.36	**5.62e-9**	**38.58**
4	2	0	5.12e-2	16.90	**4.50e-5**	**47.46**	1.64e-6	14.28	**2.81e-9**	**41.94**
5	2	$\pi/4$	6.11e-2	6.51	**8.31e-5**	**35.18**	1.99e-6	11.08	**2.74e-9**	**39.69**

Table 2: Forward problem results of phantom 2 given a circular domain Ω, and phantom 3 with domains Ω_1, Ω_2 and Ω_3 as defined in Figure 7

	phantom	n	φ	u		u_x	
				MSE	PSNR	MSE	PSNR
1	$2, \Omega$	1	0	2.86e-4	47.43	1.70e-8	40.61
2	$2, \Omega$	1	$\pi/2$	1.74e-3	38.90	1.03e-8	33.79
3	$2, \Omega$	2	0	1.29e-4	45.52	3.26e-9	41.18
4	$2, \Omega$	2	$\pi/2$	1.30e-4	45.49	8.63e-4	37.07
5	$3, \Omega_1$	1	$\pi/4$	6.42e-5	47.16	5.64e-9	39.43
6	$3, \Omega_1$	2	$\pi/4$	1.08e-4	34.03	2.61e-9	41.32
7	$3, \Omega_2$	1	$\pi/4$	1.08e-4	44.91	4.51e-9	40.39
8	$3, \Omega_2$	2	$\pi/4$	5.93e-5	36.64	2.74e-9	41.11
9	$3, \Omega_3$	1	$\pi/4$	1.22e-4	44.37	9.17e-9	37.31
10	$3, \Omega_3$	2	$\pi/4$	1.74e-4	31.96	2.24e-9	41.99

The forward problem was repeated using phantom 2 where the background and circle conductivities were 1 and 0.2 respectively (Figure 3, middle). Four different current combinations were applied. Quantitative results are summarized in lines 1-4 of Table 2. Figure 5 shows the results for both u and $\partial u/\partial x$ for $n = 2$ and $\varphi = \pi/2$. The right column presents the relative error defined as $e(x, y) = (u_{\text{fem}}(x, y) - u(x, y))/\max(u_{\text{fem}})$.

5 Inverse Problem for EIT

In the inverse problem, the electrical potential $u(x)$ is known while $\sigma(x)$ is unknown. Since we have a network which approximates $u(x)$, we can evaluate it at any point x. The objective function equation 4 then takes the form

$$\begin{aligned}\mathcal{I}\Big(\sigma(x; w_\sigma)\Big) =& \frac{\lambda}{N_s}\sum_{i=1}^{N_s}|\mathcal{L}_i|^2 + \frac{\mu}{K}\sum_{k\in\text{top}_K(|\mathcal{L}_i|)}|\mathcal{L}_k| \\ &+\frac{1}{N_b}\sum_{b=1}^{N_b}\Big|\sigma(x_b) - \sigma_0(x_b)\Big| + \alpha\|w_\sigma\|_2^2 + \frac{\beta}{N_s}\sum_{i=1}^{N_s}|\nabla\sigma(x_i)|^p.\end{aligned} \tag{8}$$

As in the forward problem, the first two terms enforce σ to satisfy the PDE, where $\mathcal{L}_i$ is defined in equation 6. The third term imposes the boundary conditions, and the fourth regularizes the network parameters. The last term is the total variation regularization ($p = 1$) which promotes the solution towards a piecewise constant function.

Additional inverse problem generalization may exploit multiple u approximations for several currents ψ_j. The σ calculation thus, simultaneously relies on all $\{u_j\}$, resulting a more stable solution,

$$\mathcal{L}_{ij} := \nabla\cdot\Big(\sigma(x_i)\nabla u_j(x_i)\Big). \tag{9}$$

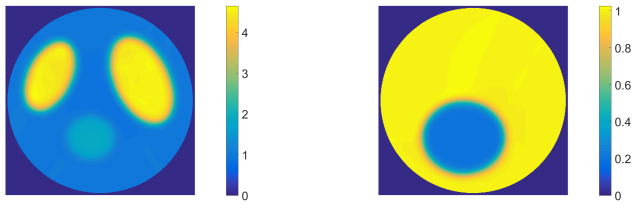


Figure 6: Reconstructed σ by the generalized inverse problem. Left: phantom 1, Right: phantom 2. MSE and PSNR are summed up in lines 1-2 of Table 3.

Table 3: σ reconstruction by the generalized inverse problem

	phantom	MSE	PSNR
1	$1, \Omega$	4.06e-2	27.90
2	$2, \Omega$	9.04e-5	40.44
3	$3, \Omega_1$	1.07e-2	33.67
4	$3, \Omega_2$	5.9e-3	36.27
5	$3, \Omega_3$	1.12e-2	33.51

Then equation 8 is generalized to

$$\mathcal{J}_j\big(\sigma(x;w_\sigma)\big) = \frac{\lambda}{N_s}\sum_{i=1}^{N_s}|\mathcal{L}_{ij}|^2 + \frac{\mu}{K}\sum_{k\in \text{top}_K(|\mathcal{L}_{ij}|)}|\mathcal{L}_{kj}|, \tag{10}$$

and

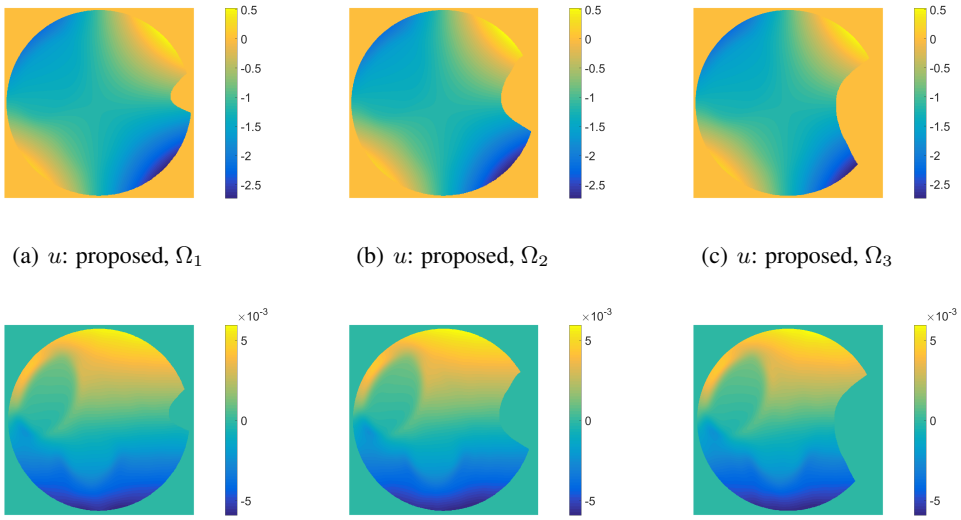
$$\mathcal{I}\Big(\sigma(x;w_\sigma\Big) = \sum_j \mathcal{J}_j\Big(\sigma(x;w_\sigma)\Big) + \frac{1}{N_b}\sum_{b=1}^{N_b}\Big|\sigma(x_b)-\sigma_0(x_b)\Big| + \alpha\|w_\sigma\|_2^2 + \frac{\beta}{N_s}\sum_{i=1}^{N_s}|\nabla\sigma(x_i)|^p. \tag{11}$$

Reconstruction results by the generalized inverse problem are shown in Figure 6 and lines 1-2 in Table 3. In both phantoms we used four current combinations $\psi_{n,\varphi}$: $(1,0)$, $(1,\pi/4)$, $(2,0)$, $(2,\pi/4)$, and $\sigma_0 = 1$.

6 Free Shape Geometry

We applied the proposed method to arbitrary domains Ω_1, Ω_2 and Ω_3, see Figure 7. The random sample points within the domain and along its boundary can be easily obtained as can be seen in Figure 7 right. The forward problem results applied on phantom 3 are presented in lines 5-10 of Table 2. Figure 8 shows the results for $n = 2$ and $\varphi = \pi/4$ for the three domains. The outcome of the generalized inverse problem equation 11 is shown in Figure 12 and lines 3-5 of Table 3.

Figure 7: Free shapes. Left to right: Ω_1, Ω_2, Ω_3 and sample points of Ω_3.



(a) u: proposed, Ω_1 (b) u: proposed, Ω_2 (c) u: proposed, Ω_3

(d) $\partial u/\partial x$: proposed, Ω_1 (e) $\partial u/\partial x$: proposed, Ω_2 (f) $\partial u/\partial x$: proposed, Ω_3

Figure 8: Forward problem results of u and $\partial u/\partial x$ for $n = 2$, $\varphi = \pi/4$ given phantom 3 applied on different domains. Left to right: Ω_1, Ω_2, Ω_3. MSE and PSNR are summed up in lines 6, 8 and 10 of Table 2.

7 Additional Inverse Problem Examples

In this section we exemplify the proposed algorithm with additional inverse problem setting introduced by Xu & Darve (2019). Unknown parameters which are parameterized by neural networks are inferred out of few measurements. The differential operators are approximated in finite differences schemes discretized by Δt and Δh, the temporal and spatial grid spacings.

7.1 Diffusion Equation

Diffusion PDE is given by

$$\partial_t u(\mathbf{x}, t) = C(\mathbf{x})\nabla^2 u(\mathbf{x}, t) + f(\mathbf{x}, t) \quad (\mathbf{x}, t) \in [-1, 1]^2 \times [0, T],$$

where the conductivity $C(\mathbf{x})$ is the unknown parameter. In the following we solve the equation in 2D where $\mathbf{x} := (x, y)$. Given two measurements at adjacent times $u(x, y, t_1)$ and $u(x, y, t_1 + \Delta t)$, the loss function then takes the form

$$\mathcal{L} = \frac{u(x, y, t_1 + \Delta t) - u(x, y, t_1)}{\Delta t} - C(x, y)\frac{D^2 u(x, y, t_1) + D^2 u(x, y, t_1 + \Delta t)}{2(\Delta h)^2} - f(x, y, t_1),$$

where D^2 is the central scheme approximation for the Laplacian on 3x3 stencil. The analytical 2D conductivity function is given by

$$C(x, y) = 1 + e^{-(12xy)^2},$$

and

$$u(x, y, t) = e^{-\pi^2 t} \sin \pi x \cos(\pi y + \pi/4),$$
$$f(x, y, t) = \pi^2 u(x, t)(2C(x) - 1).$$

The measurements were taken at $t_1 = 0.1$ with $\Delta t = 0.001$ and $\Delta h = 0.002$. The result is shown in Figure 9. As can be seen the smoothness regularizer and the L_∞ terms increase the reconstruction accuracy. In addition, we added few Gaussian noise levels to the measurements. Quantitative results are summarized in lines $1 - 4$ of Table 4.

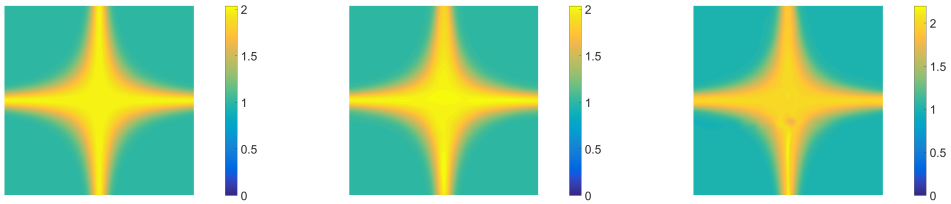


Figure 9: Reconstructed conductivity $C(x)$. Left: ground truth, Middle: proposed, Right: Xu & Darve (2019). MSE and PSNR are given in lines 1-4 of Table 4

7.2 Wave Equation

The second example is the wave equation given by

$$\partial_{tt}u(x,y,t) = \eta^2(x,y)\nabla^2 u(x,y,t) \quad ((x,y),t) \in [-1,1]^2 \times [0,T].$$

The unknown parameter is the velocity field $\eta(x,y)$. Its analytical value is given by

$$\eta(x,y) = 0.1 + e^{-\left((x-0.5)^2)+(y-0.1)^2\right)},$$

and the two initial conditions are

$$u(x,y,0) = e^{-10x^2-3y^2},$$

and

$$u(x,y,\Delta t) = e^{-10(x-0.001)^2-3(y-0.001)^2}.$$

Solutions for all time steps t are iteratively calculated using the finite differences scheme. The velocity field $\eta(x,y)$ is extracted using three measurements with the following loss function

$$\mathcal{L} = \frac{u(x,y,t_1+\Delta t) - 2u(x,y,t_1) + u(x,yt_1-\Delta t)}{(\Delta t)^2} - \eta^2(x,y)\frac{D^2u(x,y,t_1)}{(\Delta h)^2}.$$

Reconstruction result is depicted in Figure 10 showing the advantage of the proposed method. Additional quantitave results of noisy measurements are in lines 5-8 of Table 4.

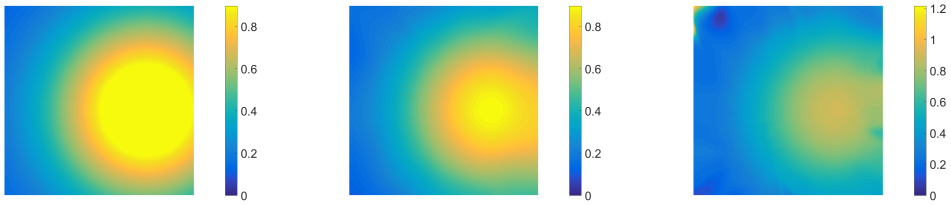


Figure 10: Reconstructed velocity field $\eta(x,y)$. Left: ground truth, Middle: proposed, Right: Xu & Darve (2019). MSE and PSNR are given in lines 5-8 of Table 4

8 Implementation Details

The network architecture has 4 layers having 26, 26, 26 and 10 neurons. The algorithm was implemented by TensorFlow Ten (2015) using the ADAM optimizer which is a variant of the SGD algorithm. We used the same hyper parameters set in almost all of our experiments. Batch size=1000, decaying learning rate starting at $5e-3$. The learning rate was factored by 0.8 every 200 epochs, $Ns = 80000$, $N_b = 1200$, $\lambda = 0.01$, $\alpha = 1e-8$, $K = 40$, $\mu = 0.01$, and $\beta = 0.01$. In the diffusion

Table 4: Inverse problem results for diffusion and wave equations

	Equation	σ_{noise}	Xu & Darve (2019)		Proposed	
			MSE	PSNR	MSE	PSNR
1	Diffusion	0	4.90e-4	39.12	2.04e-4	42.92
2	Diffusion	1e-7	0.259	11.89	0.249	12.06
3	Diffusion	5e-7	0.335	10.77	0.316	11.02
4	Diffusion	1e-6	0.448	9.50	0.438	9.60
5	Wave	0	0.015	19.01	0.013	19.53
6	Wave	3e-9	0.049	13.92	0.015	19.16
7	Wave	3.5e-9	0.056	13.34	0.018	18.32
8	Wave	4e-9	0.059	13.11	0.029	16.19

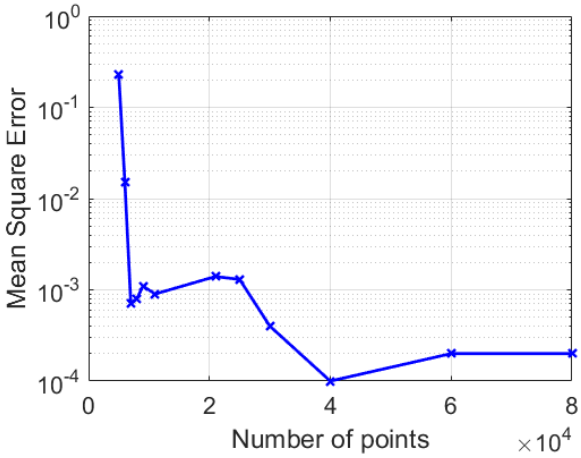


Figure 11: Mean Square Error as a function of number of sampled points

and wave equation case we set $\lambda = 1$, $\beta = 1e-4$, $\mu = 1e-4$ and $\alpha = 1e-5$. Running time, for the EIT, on Intel i7-8650u CPU was about 15 minutes for the forward problem and 13 minutes for the generalized inverse problem. For the diffusion and wave equation the running time was about 5 minutes.

Figure 11 presents the sensitivity of the algorithm to the number of sampled points. As expected, with large number of points we obtain a plateau while with small and intermediate number (up to 50000) the error is not monotonic due to local minima.

9 Discussion

Deep networks by their nature use compositions of simple functions such as matrix multiplication and non-linear activations like sigmoid or tanh. This structure (i) enables the approximation of an arbitrary function Hornik (1989) and (ii) is inherently differentiable. The network architecture dictates the number of degrees of freedom which in turn enables the expressibility of complex functions. In this work we present a unified framework for the solution of forward and inverse problems in partial differential equations in an arbitrary domain. The algorithm relies on direct approximation of the unknown function by a neural network which yields an *analytical* smooth solution. The network is trained to satisfy the PDE and boundary conditions in an unsupervised fashion by the minimization of a cost function. The optimization procedure depends on random points set within the domain and its boundary. The problem is therefore mesh free with free-form domain. We introduce a cost function which is composed of both L_2 and L_∞ fidelity terms, boundary conditions constraints and additional regularizers. The mathematical rationale of the L_∞ norm which controls possible

outliers was verified in our numerical experiments. We also stress the robustness of our approach exemplified by having the same hyper parameters set for all our experiments.

This framework alleviates several problems of the finite differences and finite elements methods. In particular meshing, discretization and derivatives approximation are solved in a simple and natural way. Numerical solutions of PDEs in an arbitrary domain are of extreme importance, in particular in medical imaging applications. The framework is demonstrated by an elliptic system in $\mathbb{R}^2$ applied to Electrical Impedance Tomography for both forward and inverse problems. Promising results were achieved for complex and non monotonic functions. Rigorous analysis of the approximation error and its relation to the network architecture and design are under current study and will be published elsewhere. Future research includes also full tomography solution, higher dimensional problems and other classes of PDEs such as dynamic non-linear equations (Burgers, Navier-Stokes etc.)

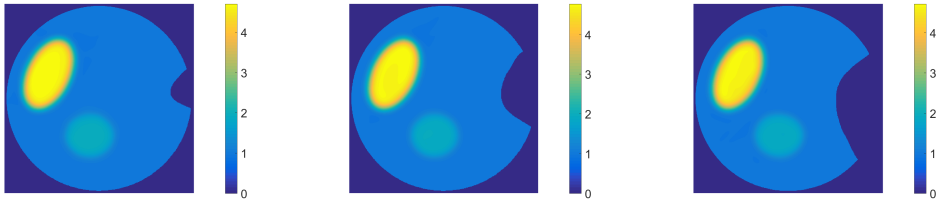


Figure 12: Reconstructed σ by the generalized inverse problem given phantom 3 applied on different domains. Left to right: Ω_1, Ω_2, Ω_3. MSE and PSNR are given in lines 3-5 of Table 3

REFERENCES

TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

M. Alsaker and J. L. Mueller. Use of an optimized spatial prior in D-bar reconstructions of EIT tank data. *Inverse Problems and Imaging*, 12:883–901, 2018.

T. K. Bera, S. K. Biswas, K. Rajan, and J. Nagaraju. Improving image quality in electrical impedance tomography (eit) using projection error propagation-based regularization (pepr) technique: A simulation stud. *Journal of Electrical Bioimpedance*, 2:2–12, 2011.

M. M. Chiaramonte and M. Kiener. *Solving differential equations using neural networks*, 2017. `http://cs229.stanford.edu/proj2013`.

L.C. Evans. *Patial Differential Equations*. the American Mathematical Society, 2010.

Y. Fan and L. Ying. Solving electrical impedance tomography with deep learning. *arXiv 1906.03944v1*, 2019.

M. Feigin, D. Freedman, and B. W. Anthony. A deep learning framework for single-sided sound speed inversion in medical ultrasound. *arXiv 1810.00322v3*, 2018.

J. Han, A. Jentzen, and E. Weinan. Solving high-dimensional partial differential equations using deep learning. *arXiv 1707.02568*, 2018.

K. Hornik. Multilayer feedforward networs are universal approximators. *Neural Networks*, 2:359–366, 1989.

Y. Khoo and L. Ying. Switchnet: A neural network model for forward and inverse scattering problems. *arXiv 1810.09676v1*, 2018.

H. Li, J. Schwab, S. Antholzer, and M. Haltmeier. NETT: Solving inverse problems with deep neural networks. *arXiv 1803.00092*, 2018.

Y. Li, J. Lu, and A. Mao. Variational training of neural network approximations of solution maps for physical models. *arXiv 1905.02789v1*, 2019.

A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos. Using deep neural networks for inverse problems in imaging. *IEEE Signal Processing Magazine*, 2018.

M. T. McCann, K. H. Jin, and M. Unser. A review of convolutional neural networks for inverse problems in imaging. *arXiv 1710.04011*, 2017.

J. Mueller and S. Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*. SIAM, 2012.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv 1711.10561v1*, 2017.

J.K. Seo, K. C. Kim, A. Jargal, K. Lee, and B. Harrach. A learning-based method for solving ill-posed nonlinear inverse problems: A simulation study of lung EIT. *SIAM journal on Imaging Sciences*, 12:1275–1295, 2019.

Z. Shen, H. Yang, and S. Zhang. Nonlinear approximation via compositions. *arXiv:1902.10170*, 2019.

S. Siltanen, J. Mueller, and D. Isaacson. An implementation of the reconstruction algorithm of a nachman for the 2d inverse conductivity problem. *Inverse Problems*, 16(3):681–699, 2000.

J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *arXiv 1708.07469v3*, 2017.

K. Xu and E. Darve. The neural network approach to inverse problems in differential equations. *arXiv 1901.07758v1*, 2019.