Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 33(12): 7637–7675.
DOI: 10.3934/era.2025338
Received: 25 June 2025
Revised: 22 September 2025
Accepted: 04 December 2025
Published: 19 December 2025

Research article

Image reconstruction in electrical impedance tomography using deep neural networks with differential evolution

Margaret Esther C. Cruz[1], **Renier G. Mendoza**[1,2] **and Rhudaina Z. Mohammad**[1,2,*]

[1] Data Science Program, College of Science, University of the Philippines Diliman, Quezon City 1101, Philippines

[2] Institute of Mathematics, College of Science, University of the Philippines Diliman, Quezon City 1101, Philippines

* **Correspondence:** Email: rmohammad@math.upd.edu.ph; Tel: +63289818500.

Abstract: Electrical impedance tomography (EIT) is a noninvasive imaging technique that reconstructs the internal conductivity distribution of an object by measuring electric potentials on its surface. Mathematically, EIT is twofold: the forward problem solves a generalized Laplace equation with conductivity coefficients to determine the electric potential, while the inverse problem requires estimating conductivity from noisy and incomplete boundary measurements—an ill-posed task highly sensitive to noise. This paper proposed a hybrid deep learning–evolutionary framework for EIT image reconstruction. The fully-connected feedforward neural networks (FNNs) and convolutional neural networks (CNNs) were trained to approximate the forward map from conductivity to electric potential, eliminating the need for repeated finite element simulations during inversion. For the inverse problem, we formulate the recovery of conductivity distributions as a global optimization task using differential evolution and its variants. Among them, success-history based adaptive differential evolution (SHADE) achieved the most accurate and robust results. While the proposed CNN-SHADE algorithm demonstrated competitive reconstruction performance, the FNN-SHADE approach provided a favorable trade-off between accuracy and computational efficiency. Furthermore, the FNN-SHADE framework outperformed the traditional finite element-based method in computational speed, while maintaining accuracy. By embedding a neural forward operator into the differential evolution loop, the framework offered a scalable, data-driven alternative to traditional EIT reconstruction methods.

Keywords: electrical impedance tomography; generalized Laplace equation; image reconstruction; fully-connected neural network; convolutional neural network; inverse problem; differential evolution

1. Introduction

Electrical impedance tomography (EIT) is a noninvasive imaging technique that reconstructs the internal conductivity distribution of a subject by injecting small electrical currents through surface electrodes and measuring the resulting voltages. This requires solving both a forward and an inverse problem governed by a generalized Laplace equation with conductivity coefficients. The forward problem involves computing the electric potential given a known conductivity distribution, while the inverse problem involves recovering conductivity from boundary voltage measurements. EIT arises in a wide range of applications, particularly in biomedical imaging where it offers a low-cost portable alternative [1], facilitates early breast cancer diagnosis [2], and aids in detecting pulmonary emboli and abdominal lesions [3]. In geophysical imaging, it is used to analyze rock resistivity based on porosity and water content [4], as well as to locate underground mineral deposits and leaks [5]. Industrial applications are equally diverse, ranging from damage detection in self-sensing composite tubes [6] to monitoring pharmaceutical crystallization processes [7].

Due to rapid technological advances and increasing accuracy in data collection and processing, there is a demand for EIT image reconstruction algorithms to improve imaging resolution. However, reconstructing high-quality images from EIT data remains a challenge due to the ill-posed nature of the inverse problem [8]. This is because, in the Hadamard sense, the inverse conductivity problem only meets existence and uniqueness, but not stability (continuous dependence on data), as the inverse map is discontinuous [9]. It has been demonstrated that because the forward map lacks a continuous inverse, the problem requires a regularization strategy based on a family of continuous mappings [10]. To be precise, the EIT inverse problem admits at best a logarithmic stability estimate [11], highlighting its severe ill-posedness. Consequently, small measurement noise can cause large errors in the reconstructed conductivity, necessitating regularization techniques in classical numerical methods for solving partial differential equations (PDEs) such as finite element method (FEM), finite volume method, and spectral element method. For instance, [12] developed a software package for EIT utilizing FEM by discretizing the conductivity domain, formulating the governing PDEs, and assessing its accuracy, computational efficiency, and practical application in reconstructing resistivity distributions from boundary voltage measurements. However, they are computationally expensive, particularly for complex geometries and high-resolution reconstructions, due to the need for fine mesh discretization and iterative solvers. They also scale poorly with domain size, as larger problems lead to increasingly large system matrices that hinder real-time applications. Even small discretization errors can impact accuracy due to their sensitivity to mesh quality and boundary conditions. Moreover, they may struggle to maintain accuracy when the true conductivity distribution exhibits structural irregularities, such as sharp transitions or discontinuous inclusions, which exacerbate instability. Robustness to such irregularities is therefore a key consideration in modern EIT reconstruction methods, as highlighted in related work on discontinuous PDE inputs [13].

Machine learning techniques, particularly data-driven neural networks, have shown promising results in EIT image reconstruction. Artificial neural networks can learn complex mappings between input voltages and output conductivity distributions, offering a faster and more efficient approach than classical PDE-based methods. Among these, multilayer perceptrons (MLPs) have been explored extensively in EIT. For instance, [14] systematically investigated the performance of different MLP architectures for solving the EIT inverse problem, focusing on anomaly localization on a sensing

surface. MLPs rely on fully-connected layers and treat input features independently, which limits their ability to exploit spatial relationships. In contrast, convolutional neural networks (CNNs) use convolutional filters to capture local spatial patterns, making them more effective for modeling the structured nature of EIT data. In particular, CNNs have demonstrated robust performance in capturing spatial dependencies within EIT data, thereby improving image quality [15].

Alongside neural networks, global optimization algorithms such as differential evolution [16], genetic algorithm, particle swarm optimization, and simulated annealing have been explored to fine-tune neural network parameters and enhance model generalizability. Unlike gradient-based optimization techniques, which can be sensitive to local minima, these algorithms leverage metaheuristic strategies to efficiently search high-dimensional parameter spaces and escape local optima. In the context of standard neural networks, this capability was demonstrated by using various metaheuristic optimization algorithms to effectively train MLPs [17]. Furthermore, this approach has been extended to deep learning, where algorithms such as simulated annealing and differential evolution were applied to CNNs to improve classification accuracy compared to traditional methods [18]. Compared to other global optimization algorithms, differential evolution is known for its simple yet powerful self-adaptive mutation and crossover mechanisms, making it well-suited for challenging search landscapes, particularly those where gradients are very small, such as those arising from inverse problems like EIT. For instance, [19] demonstrated that differential evolution's mutation scheme allows for significantly faster convergence than genetic algorithm while using fewer control parameters. Moreover, [20] provides a comparative analysis showing that differential evolution offers superior stability and versatility compared to particle swarm optimization, particularly when navigating the complex, multimodal environments often encountered in such inverse problems. Differential evolution is gradient-free and relies solely on function evaluations, allowing it to progress without explicit regularization or carefully-chosen initial guesses, while its self-adaptive and population-based nature enhances robustness in exploring the search space. As such, differential evolution has been used in several EIT algorithms. For example, [21] investigated the performance of differential evolution and other evolutionary algorithms in reconstructing irregular insulating objects. More recently, [22] proposed an approach that combines differential evolution with a modified Newton-Raphson algorithm (MNR), which allows each individual in the population to be optimized using the MNR algorithm during the mutation stage. Both works, however, rely on repeatedly solving the forward problem via traditional numerical solvers such as FEM, which can be computationally expensive during optimization.

This paper introduces a deep learning-evolutionary framework to enhance EIT image reconstruction. Specifically, we propose using neural networks to solve the forward problem and differential evolution to solve the inverse problem by minimizing the discrepancy between simulated and measured boundary voltages. In this work, we primarily focus on deep fully-connected feedforward neural networks (FNNs) due to their simplicity and ease of implementation. FNNs require minimal architectural tuning and make no assumptions about input structure, such as spatial locality or sequential patterns. This makes them suitable for low-dimensional, structured inputs like parameterized conductivity values. While FNNs serve as the baseline architecture in this study, we also evaluate CNNs to assess the benefits of spatial feature extraction in solving the forward and inverse problems. Moreover, we explore different differential evolution variants to assess their efficiency and accuracy in recovering conductivity distributions under various conditions.

Our work is motivated by recent progress in data-driven PDE solvers [14, 15] and evolutionary optimization techniques [21, 22] for ill-posed inverse problems. However, most prior studies apply these two methodologies independently. Our contribution is distinct in embedding a pretrained neural network forward operator directly within a global optimization loop. In prior hybrid approaches, differential evolution or other global optimization algorithms are used to train or fine-tune the neural network itself. Early investigations applied the standard version of this technique to train radial basis function networks [23] or optimize back propagation networks [24]. Similarly, [25] utilized it to train artificial neural networks for nonlinear reconstruction. Advancing this methodology, [26] later employed a hybrid variant of particle swarm optimization to optimize radial basis function neural networks. In contrast, in our work, differential evolution optimizes only the conductivity parameters of the EIT inverse problem, with the neural network acting as a fast surrogate PDE solver. Moreover, this approach differs from standard FEM-based iterative methods. Such algorithms have been implemented specifically for imaging human brain function [27], while a general finite element model has been formulated for impedance computed tomography [28]. These conventional techniques typically optimize over the full conductivity field with strong regularization. Instead, under a piecewise-conductivity assumption we recover a low-dimensional parameter vector, substantially reducing the search space. To our knowledge, this explicit use of a neural network as a forward model inside a global inverse solver—rather than using global optimizers to train the network—has not been demonstrated for EIT.

By using the data-driven learning capabilities of deep neural networks to approximate the nonlinear mapping from conductivity to boundary voltage, and the optimization power of differential evolution to explore the conductivity space globally, our proposed method reduces computational cost by eliminating the need to repeatedly solve large PDE systems during reconstruction. Instead of relying on mesh-based solvers at every iteration, this approach uses a pretrained model to perform forward evaluations without solving the PDE or assembling matrices on a mesh, thereby reducing computation time. Beyond improving speed, embedding the neural forward operator within the differential evolution loop also enables much broader global exploration than is feasible with FEM-based inversions, allowing the optimizer to test a larger and more diverse set of candidate conductivity distributions. While this achieves reconstruction accuracy comparable to FEM-based techniques, it does so with significantly improved computational efficiency, making our proposed method more practical for real-time scenarios and expanding applicability of EIT across various domains.

2. Problem formulation

The mathematical foundation of EIT dates back to the Calderón problem [29], originally posed by the mathematician Alberto Calderón in 1980, which focuses on determining the internal conductivity of a domain $\Omega \subseteq \mathbb{R}^d$ by measuring voltage on its boundary $\partial\Omega$. To be precise, EIT is divided into two: a forward problem and an inverse problem [30].

The forward problem solves for the electric potential $\varphi \in H^1(\Omega)$ and the corresponding boundary voltage $V = \varphi|_{\partial\Omega}$ in the generalized Laplace equation with conductivity coefficients $\sigma \in L^\infty(\Omega)$

given by

$$\begin{cases} \nabla \cdot (\sigma \nabla \varphi) = 0 & \text{in } \Omega, \\ \sigma \dfrac{\partial \varphi}{\partial n} = f & \text{on } \partial\Omega, \end{cases} \tag{2.1}$$

where n denotes the unit outer normal to $\partial\Omega$. The boundary current f is chosen so that the law of conservation of charge is preserved, that is,

$$\int_{\partial\Omega} f = 0. \tag{2.2}$$

To ensure the uniqueness of the solution, the electric potential φ must also satisfy

$$\int_{\partial\Omega} \varphi = 0, \tag{2.3}$$

which amounts to choosing the reference voltage. In this work, we adopt the same assumption of piecewise constant conductivity as in [31], motivated by the fact that many real-world materials, such as biological tissues and industrial composites, naturally exhibit distinct, nearly homogeneous conductivity in separate regions. This assumption aligns with the discrete nature of materials, where sharp transitions between regions are more common than smoothly varying conductivity. As such, we assume that a body $\Omega = \bigcup_{i=0}^{k-1} \Omega_i$ consists of disjoint subdomains Ω_i, corresponding to $k-1$ inclusions on a medium Ω_0. The conductivity σ of Ω can then be expressed as a piecewise constant function

$$\sigma(x) = \sum_{i=0}^{k-1} \sigma_i \chi_i(x), \tag{2.4}$$

where $\sigma_i \in \mathbb{R}$ are distinct conductivity parameters and χ_i is a characteristic function of Ω_i.

The inverse problem, or Calderón problem, on the other hand, involves recovering the conductivity σ throughout Ω based solely on measurements of φ on $\partial\Omega$. This is typically accomplished by introducing a known current on $\partial\Omega$ and measuring the corresponding voltage response, or vice versa. This process is captured by the Dirichlet-to-Neumann (DtN) map $\Lambda_\sigma : \varphi|_{\partial\Omega} \mapsto \sigma \partial_n \varphi|_{\partial\Omega}$, which maps each boundary voltage distribution to the corresponding boundary current flux. For essentially bounded conductivities $\sigma \in L^\infty(\Omega)$, [32] showed that in two dimensions, the DtN map uniquely determines σ, even when σ is merely bounded and possibly discontinuous. This ensures that in such configurations involving discontinuous or rough conductivities, the inverse conductivity problem is well-defined in the sense that a unique solution exists. However, it remains ill-posed in the Hadamard sense, meaning small errors in boundary measurements can still lead to large reconstruction errors. Hence, practical reconstruction still requires careful modeling and regularization.

To illustrate the setup, consider a simple configuration shown in Figure 1, where a domain Ω contains an inclusion Ω_1 with different conductivity σ_1 from the conductivity σ_0 of the surrounding medium Ω_0. A current is applied at the boundary of Ω and the resulting voltage is influenced by the varying conductivities. The primary objective of the inverse problem is to determine the unknown conductivities inside the domain based on the applied current and the observed boundary voltage.

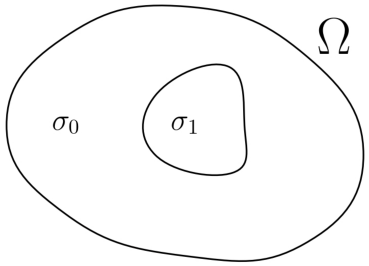

Figure 1. A representation of a body Ω containing an irregularly shaped inclusion Ω_1 with conductivity σ_1 in a medium Ω_0 with conductivity σ_0.

Mathematically, we define a forward operator $F : L^2(\Omega) \to \overline{L}^2(\partial\Omega)$ that maps the internal conductivity distribution σ to boundary voltages $\varphi|_{\partial\Omega}$, i.e., $F(\sigma) = \varphi|_{\partial\Omega}$. Here, $\overline{L}^2(\partial\Omega)$ denotes the set of all functions $f \in L^2(\partial\Omega)$ such that (2.2) holds. If the measured boundary voltage $V_* = F(\sigma_*)$ is known, then the inverse problem recovers the unknown conductivity σ_*. This leads to the following optimization strategy

$$\sigma_* = \underset{\sigma}{\operatorname{argmin}} \|F(\sigma) - V_*\|_2,$$

where $\|\cdot\|_2$ denotes the L^2-norm. In practice, however, there is a finite number, say m, of measured boundary voltages $\overline{V}_i$ at $x_i \in \partial\Omega$, for $i = 1, 2, \ldots, m$, that is, $V_* \approx \overline{V} := (\overline{V}_1, \overline{V}_2, \ldots, \overline{V}_m)$. This translates to minimizing a mean squared error (MSE) given by

$$\mathrm{MSE}(\sigma) = \frac{1}{m} \sum_{i=1}^{m} \left(F(\sigma)_i - \overline{V}_i\right)^2, \tag{2.5}$$

where $F(\sigma)_i = \varphi(x_i)$ denotes the forward evaluation of the voltage at boundary node $x_i \in \partial\Omega$. Minimizing MSE, which is equivalent to minimizing a squared ℓ^2-norm up to a constant factor, is preferred in numerical optimization due to its smoothness, differentiability, and clear interpretation as the average squared prediction error.

3. Methods

In this section, we present our proposed algorithm and describe the datasets and numerical methods used to solve the EIT forward and inverse problems.

3.1. Data acquisition

While experimental EIT datasets exist, we opted to use simulated data in this study to maintain full control over the ground-truth conductivity distributions and ensure consistency across experiments. This controlled setting allowed us to systematically evaluate accuracy, stability, and the relative performance of different differential evolution variants without confounding factors such as electrode placement errors or physiological variability.

The datasets were generated using simulated EIT measurements. Conductivity distributions were created within specified domains, with each distribution incorporating random inclusions of varying shapes, sizes, and conductivity values sampled from a predefined range. These inclusions were

embedded within a homogeneous background conductivity to simulate realistic scenarios encountered in practical applications.

To obtain the corresponding electric potential distributions, forward simulations were performed using FEM as follows. We obtain the variational formulation of (2.1), where we seek $\varphi \in H^1(\Omega)$ such that for all $v \in H^1(\Omega)$, we have

$$a(\varphi, v) = \int_{\partial\Omega} fv, \quad \text{where } a(\varphi, v) := \int_{\Omega} \sigma \nabla\varphi \cdot \nabla v.$$

To incorporate normalization condition (2.3), we introduce a Lagrange multiplier $\lambda \in \mathbb{R}$ in the variational formulation. The resulting problem seeks $(\varphi, \lambda) \in H^1(\Omega) \times \mathbb{R}$ such that for all $(v, \mu) \in H^1(\Omega) \times \mathbb{R}$, we have

$$a(\varphi, v) + \lambda \int_{\partial\Omega} v + \mu \int_{\partial\Omega} \varphi = \int_{\partial\Omega} fv. \tag{3.1}$$

To numerically solve this problem, we discretize domain Ω into a finite set of nonoverlapping triangles, forming a triangular mesh. The vertices of these triangles are called nodes, with the total number of nodes in the mesh denoted by M, and the total number of boundary nodes by m. This enables us to construct a finite-dimensional subspace $S \subset H^1(\Omega)$, using piecewise linear (P1) basis functions $\{\psi_j\}_{j=1}^M$ associated with the mesh nodes. We then approximate the solution φ using the finite-dimensional basis to get $\varphi = \sum_{j=1}^M \tilde{\varphi}_j \psi_j$, where $\tilde{\varphi}_j$ are unknown coefficients. Thus, we have $\varphi(x_j) \approx \tilde{\varphi}_j$ for all $x_j \in \Omega$. This reduces (3.1) to solving for $\tilde{\varphi} := [\tilde{\varphi}_j] \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}$ in the linear system

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} \tilde{\varphi} \\ \lambda \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{3.2}$$

where $A := [a(\psi_i, \psi_j)] \in \mathbb{R}^{M \times M}$, $B := [\int_{\partial\Omega} \psi_j] \in \mathbb{R}^M$, and $b := [\int_{\partial\Omega} f\psi_j] \in \mathbb{R}^M$. We implement this in `scikit-fem`, a Python library for finite element analysis where matrices are assembled in Compressed Sparse Row format—allowing us to solve (3.2) using sparse lower-upper (LU) factorization from `scipy.sparse.linalg` module.

All datasets were initially stored in `csv` files and were converted to and accessed from the `HDF5` (`.h5`) format using `dask` which enabled efficient parallel processing and facilitated better storage, organization, and access to large numerical datasets [33]. Each dataset was split into 70% for training, 30% for validation to provide an unbiased evaluation of the model during development, and an additional 10% of the dataset size was generated for testing to assess the generalization capability of the model after training.

3.1.1. Configurations based on geometry and conductivity

Four datasets were generated based on the geometry and conductivity of configurations of inclusions within a body.

The first configuration considers a domain with a fixed geometry based on anatomical structures obtained from CT scans, as detailed in [31]. Specifically, we model the thorax, including the heart and lungs as inclusions, where only the conductivity values remain unknown. Estimating these values is crucial for real-time physiological monitoring, as conductivity variations can indicate significant

pathological or physiological changes. For instance, fluctuations in conductivity may signal fluid accumulation in the lungs, cardiac activity, or tissue abnormalities. In this configuration, the medium conductivity outside the inclusions is fixed at $\sigma_0 = 6.7$. The conductivity values σ_1 and σ_2 of the two inclusions are uniformly sampled from $[0, 10]$. Furthermore, the function f that appears in (2.2) was the same as in [31]. For the domain discretization, we use the mesh generated in [31], where the anatomical boundaries of the lungs, heart, and thoracic wall were first approximated using parametric curves fitted via Fourier series expansions. The Fourier coefficients were estimated by fitting the series to sampled boundary points extracted from a CT scan. This parametric representation enabled accurate reconstruction of anatomical contours and allowed the generation of a high-resolution finite element mesh of size $h = 0.014$ consisting of $25,309$ triangular elements and a total of $M = 12,845$ nodes, which includes $m = 379$ boundary nodes. The geometric model of the thorax is illustrated in Figure 2.

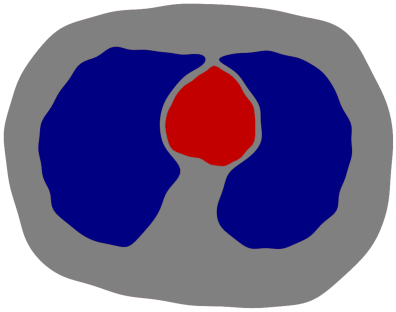

Figure 2. Domain geometry used in the first configuration, reconstructed from CT scan data as described in [31]. The thoracic region includes the heart (red) and lungs (blue) as embedded inclusions.

The remaining three configurations consider domain $\Omega = [0, 1] \times [0, 1]$, discretized into a mesh of size $h = 0.01$ consisting of $M = 20,000$ triangular elements and a total of $m = 10,201$ nodes, of which $m = 400$ nodes are at the boundary. As in [30], we take $f : \Omega \to \mathbb{R}, x := (x^{(1)}, x^{(2)}) \mapsto 2x^{(1)}x^{(2)} - x^{(1)} - x^{(2)}$ in (2.2). Each configuration is defined by specific geometric and conductivity parameters. The medium conductivity outside the inclusions is fixed at $\sigma_0 = 0.01$ to simulate a clear contrast between the inclusions and the surrounding medium. Each circular or elliptical inclusion Ω_i, $i > 0$, is characterized by its center coordinates (h_i, k_i) within the unit square. The size of the inclusion is determined by either a single radius r_i for circular inclusions or semi-axes $r_{i,a}$ and $r_{i,b}$ for ellipses, with values carefully chosen to ensure the inclusions remain entirely within the domain. For an elliptical inclusion, an additional parameter θ_i defines the tilt angle, representing the counterclockwise rotation relative to the $x^{(1)}-$axis. The electrical property of the inclusion are represented by its conductivity value σ_i, which define the contrast in electrical properties relative to the medium. To ensure a diverse dataset, the geometry and conductivity parameters are uniformly sampled from a predefined interval for a number of samples, as listed in Table 1. These interval values follow those used in [31] to ensure that the parameter bounds were physically meaningful and consistent with established practice in the literature.

Table 1. Range of parameters and number of samples for each dataset configuration.

Configuration	Parameters	Interval	Sample Size
1). Thorax	Conductivity σ_1, σ_2	[0,10], [0,10]	30,000
2). One circular inclusion	Center (h_1, k_1)	$[0,1] \times [0,1]$	60,000
	Radius r_1	[0,1]	
	Conductivity σ_1	[5,9]	
3). One elliptical inclusion	Center (h_1, k_1)	$[0,1] \times [0,1]$	90,000
	Semi-axes r_{1a}, r_{1b}	[0,1], [0,1]	
	Tilt angle θ_1	$[0, \pi]$	
	Conductivity σ_1	[5,9]	
4). Two inclusions (one circular and one elliptical)	Center 1 (h_1, k_1)	$[0,1] \times [0,1]$	150,000
	Radius r_1	[0,1]	
	Center 2 (h_2, k_2)	$[0,1] \times [0,1]$	
	Semi-axes r_{2a}, r_{2b}	[0,1], [0,1]	
	Tilt angle θ_2	$[0, \pi]$	
	Conductivities σ_1, σ_2	[5,9], [5,9]	

3.2. Neural networks for EIT forward problem

To solve the forward problem, we consider two network architectures: FNNs and CNNs. We train the model using the mean absolute error (MAE) as the loss function, which weighs all data equally—making it robust to the presence of outliers or noisy measurements. In EIT terms, the goal is to minimize absolute differences between predicted and true electric potential values—crucial in applications like medical imaging, where outliers from noise or artifacts should not dominate training.

3.2.1. Fully-connected neural network

We design an FNN architecture with several fully-connected layers, where each layer, except the output layer, was followed by a rectified linear unit (ReLU) activation function to introduce non-linearity. Here, the forward FNN model takes as input a configuration parameter vector, denoted by Θ, consisting of geometry and conductivity parameters, and outputs the corresponding electric potential distribution $\varphi_{\text{FNN}}(x_j; \Theta)$ for all $x_j \in \Omega$, $j = 1, 2, \ldots, M$. Architectural choices—such as the number of layers and neurons—are tuned for accuracy and stability using grid search cross-validation (GridSearchCV), a systematic hyperparameter tuning method provided in the Python `scikit-learn` library. This method performs an exhaustive search over a specified parameter grid using cross-validation to evaluate each combination, ensuring robust selection of hyperparameters [34]. The hyperparameters tuned included the number of layers, the number of neurons per layer, and the batch size. The range of values for these hyperparameters are shown in Table 2. We selected these ranges based on common practice in EIT deep-learning studies, which typically use multilayer architectures with tens to a few hundreds batch sizes and neurons per layer [14, 35]. Our upper limits represent the practical ceiling of what our hardware can handle while still capturing the nonlinear mapping between conductivity and boundary data. During initial tuning, performance peaked at moderate sizes. Because no clear gains appeared at the upper limits, larger models would have incurred longer training times and higher risk of overfitting without clear benefits. Table 2 therefore focuses on the

ranges where improvements were actually observed. We supplemented the grid search method with trial-and-error adjustments based on the computational resources available. This hybrid approach allowed us to fine-tune the model more effectively and select the most appropriate configurations.

Table 2. Hyperparameters with their range and the selected values.

Hyperparameters	Range	Value
Number of Hidden Layers	1 to 100	10
Number of Neurons in each Hidden Layer	32 to 256	64
Batch Size	32 to 256	128

Considering the configuration with one circular inclusion, we ran GridSearchCV ten times, with each run testing different hyperparameter configurations. This process enabled us to evaluate the impact of various combinations of hyperparameters. After evaluating the results, we determined that a network architecture with 10 layers—each containing 64 neurons, trained with a batch size of 128 using the default learning rate—offered the optimal balance between computational efficiency and accuracy. This configuration achieved a high level of precision in reconstructing conductivity distributions while maintaining reasonable training times and memory usage. Given the success of this architecture in the circular inclusion case, we applied the same architecture across all configurations for simplicity and consistency. This approach minimized the need for further exhaustive hyperparameter searches and allowed us to streamline the experimentation process without sacrificing model performance, ensuring a high degree of generalization to other problem configurations.

3.2.2. Convolutional neural network

In addition to FNN, we considered a CNN model to approximate the EIT forward solution using a spatially structured representation of the input. Unlike FNN, which operates on low-dimensional parameter vectors, the forward CNN model takes as input a conductivity distribution $\sigma(x_j)$ for all $x_j \in \Omega$, $j = 1, 2, \ldots, M$, which is represented as a two-dimensional image on a grid. This outputs the corresponding electric potential field $\varphi_{\text{CNN}}(x_j)$, also represented on a grid of the same resolution. Here, the spatial conductivity distribution σ over the domain Ω is generated using the geometry and conductivity parameters of the configuration to evaluate (2.4).

The CNN model used in this study follows a U-Net architecture [36], where each layer consists of a 3×3 convolution followed by batch normalization and a ReLU activation. Downsampling is performed using a 1×1 convolution, batch normalization, ReLU, and 2×2 max pooling, while upsampling consists of a 1×1 convolution, batch normalization, ReLU, and transposed convolution. We also adjust the number of channels and spatial dimensions to match the resolution of our input domain. In particular, the number of convolutional kernels or output channels is progressively increased in the encoder, starting from two in the first layer and doubling at each downsampling stage up to 32 in the bottleneck. This pattern is mirrored in the decoder through upsampling and skip connections. The use of only two kernels in the initial layer provides a compact feature representation, significantly reducing the number of trainable parameters while still enabling the model to capture complex patterns through deeper layers. The resulting architecture is illustrated in Figure 3.

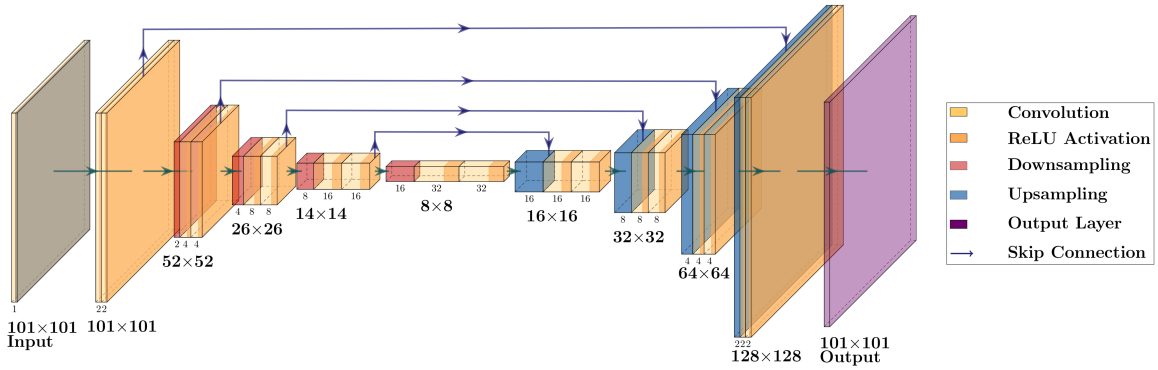

Figure 3. U-net architecture of the CNN model.

3.3. Global optimization algorithms for EIT inverse problem

The goal of the inverse problem is to identify the configuration parameters that define the geometry and conductivity of inclusions in a medium, given measured (observed) voltage measurements at the boundary. In this work, we consider boundary voltage data $\tilde{V}$ simulated using our pretrained neural network model using the configuration parameter values in Table 3.

Table 3. Input variables and their corresponding values.

Configuration Parameter Vector	Values
$\Theta_1 := (\sigma_1, \sigma_2)$	$(1, 6.3)$
$\Theta_2 := (h_1, k_1, r_1, \sigma_1)$	$(0.3, 0.7, 0.1, 7)$
$\Theta_3 := (h, k, r_{1a}, r_{1b}, \theta_1, \sigma_1)$	$(0.3, 0.7, 0.1, 0.2, \pi/4, 7)$
$\Theta_4 := (h_1, k_1, r_1, h_2, k_2, r_{2a}, r_{2b}, \theta_2, \sigma_1, \sigma_2)$	$(0.3, 0.7, 0.1, 0.7, 0.3, 0.1, 0.2, \pi/4, 6, 8)$

To formulate the inverse problem in our deep-learning framework, we first need to address the so-called inverse crime [37], which occurs when the same model and discretization scheme are used for both generating synthetic data and solving the inverse problem. This can lead to overly accurate results, as the inverse solver may recover parameters that fit the synthetic data too well, due to lack of model mismatch or measurement uncertainty. To avoid this, we introduce noise to the simulated boundary voltage data to allow realistic measurement errors and ensure a more robust and honest evaluation of the model performance under practical conditions. In particular, we assume a noisy boundary voltage data $\overline{V} = (\overline{V}_1, \overline{V}_2, \ldots, \overline{V}_m)$ such that $\overline{V}_i = \tilde{V}_i + z_i$, where z_i is the Gaussian noise with a mean of 0 and a standard deviation of 0.01 times the standard deviation of the true solution $\tilde{V}$. The actual solutions to the inverse problem that we aim to achieve are those shown in Table 3.

So far, we have a neural network model of the forward problem, which may be FNN or CNN, trained to predict electric potential distributions. In view of (2.5), we define a forward neural network operator $\mathcal{F}_{\text{NN}}$, which maps the configuration parameter vector $\Theta = (h_i, k_i, r_{i,a}, r_{i,b}, \theta_i, \sigma_i)$ to boundary voltages, that is, $\mathcal{F}_{\text{NN}}(\Theta) = \varphi_{\text{NN}}|_{\partial\Omega}$ where $\varphi_{\text{NN}}(x; \Theta)$, $x \in \Omega$, is the output of the pretrained neural network model. Given a measured (observed) boundary voltage data $\overline{V}$, we find the configuration parameter vector Θ_*, containing geometry and conductivity information, that minimizes the mean squared error (MSE) between the predicted and measured boundary voltage, which we denote by $\mathcal{J}_{\text{NN}}$. The minimization

problem can then be written as follows:

$$\Theta_* = \underset{\Theta}{\operatorname{argmin}}\, \mathcal{J}_{\mathrm{NN}}(\Theta) := \underset{\Theta}{\operatorname{argmin}}\, \frac{1}{m} \sum_{i=1}^{m} \left(\mathcal{F}_{\mathrm{NN}}(\Theta)_i - \overline{V}_i\right)^2, \tag{3.3}$$

where $\mathcal{F}_{\mathrm{NN}}(\Theta)_i = \varphi_{\mathrm{NN}}(x_i)$ denotes the predicted voltage at boundary node $x_i \in \partial\Omega$.

To numerically solve this inverse problem, we propose using differential evolution (DE) for minimizing the objective function $\mathcal{J}_{\mathrm{NN}}(\Theta)$. While the EIT inverse problem is fundamentally ill-posed, it can be formulated as a well-defined optimization task. The objective function is typically non-convex and may exhibit large flat regions where the gradient is extremely small, even when the current estimate is far from the true conductivity distribution. In such cases, small variations in the cost function do not reflect corresponding improvements in the reconstruction error. This poses a major challenge for iterative gradient-based algorithms, which tend to stagnate in nearly flat regions and, thus, rely heavily on a good initial guess to approach the global solution. DE, in contrast, is a global optimization algorithm known for its straightforward implementation, relatively few control parameters, and robust convergence in exploring complex search spaces, even in the face of ill-posedness [16]. It is a population-based optimization technique designed to optimize real-valued multidimensional objective functions where individuals in the population are perturbed by the difference between two randomly chosen individuals, creating a new candidate solution. This perturbation helps DE avoid local optima and increases its robustness for complex search spaces. Using DE, we aim to find Θ such that the predicted boundary voltages $\mathcal{F}_{\mathrm{NN}}(\Theta)$ best match the measured voltages $\overline{V}$, in terms of MSE. This begins with a population of candidate solutions $\{\tilde{\Theta}_1, \ldots, \tilde{\Theta}_K\}$, randomly generated from a given set of bounds for each configuration parameter. Here, $\tilde{\Theta}_i$ represents a possible parameter configuration. During mutation, a new candidate vector Φ_i is generated for each individual by perturbing three randomly selected population members, say $\tilde{\Theta}_a$, $\tilde{\Theta}_b$, and $\tilde{\Theta}_c$, according to the rule:

$$\Phi_i = \tilde{\Theta}_a + s(\tilde{\Theta}_b - \tilde{\Theta}_c),$$

where s is a user-defined scaling factor. In the crossover step, this mutant vector Φ_i is combined with the current target vector $\tilde{\Theta}_i$ to form a trial vector using

$$\tilde{\Phi}_i = c\Phi_i + (1 - c)\tilde{\Theta}_i,$$

where c is the crossover probability. Finally, during selection, the trial vector $\tilde{\Phi}_i$ is evaluated using the objective function $\mathcal{J}_{\mathrm{NN}}$, and it replaces $\tilde{\Theta}_i$ in the next generation if it yields a lower objective value. After the maximum number of function evaluations is reached, the best-performing individual in the final population is selected as the solution. In our implementation, we used a population size $K = 50$, mutation scaling factor $s = 0.1$, and crossover rate $c = 0.9$. However, the parameters and learning strategies are highly problem-dependent.

To address this limitation, we explore four DE-variant algorithms: self-adaptive differential evolution (SADE), adaptive differential evolution with optional external archive (JADE), success-history based adaptive differential evolution (SHADE), and self-adaptive population-based differential evolution (SAPDE). SADE adaptively adjusts the scaling factor s and the crossover rate c based on the performance of the population during optimization, eliminating the need for manual tuning [38]. In our implementation, SADE was initialized with a population size $K = 50$, with s and c

updated based on success-history. Building on SADE, JADE introduces an adaptive parameter mechanism that allows the algorithm to adjust to different optimization stages dynamically [39]. Additionally, JADE incorporates an optional external archive that stores promising solutions, preserving diversity and guiding the search more effectively. We used initial adaptive scaling factor $s = 0.5$ and initial adaptive crossover rate $c = 0.5$. SHADE further refines parameter adaptation by maintaining a historical record of successful parameter settings, computing mean values across generations to enhance convergence [40]. In SHADE, we used the initial scaling factor $s = 0.5$ and initial crossover rate $c = 0.5$. While these DE variants focus on tuning mutation and crossover parameters, they keep the population size fixed. In contrast, SAPDE introduces a dynamic population size adjustment mechanism, removing the need for pre-defining an optimal population size [41], which allows the population size to evolve throughout the optimization process based on performance indicators such as diversity and fitness improvement trends. We initialized with a population of $K = 50$, and used Gaussian method instead of uniform method.

For comparison, we also consider three population-based optimization algorithms: genetic algorithm (GA), particle swarm optimization (PSO), and swarm simulated annealing (SSA). These methods are selected due to their widespread use in inverse problems and global optimization. They are also among the simplest and most commonly used baseline algorithms in the literature, and are often compared with DE. For example, in [42], GA, PSO, DE, and simulated annealing (SA) were directly compared in solving a nonconvex dynamic economic dispatch problem, highlighting their general applicability and effectiveness in complex optimization tasks. While DE was recently found to outperform PSO in most real-world and benchmark problems [20], we are interested in evaluating their relative performance in the context of our proposed deep-learning framework for the EIT inverse problem.

GA is inspired by the principles of natural evolution, where the fittest individuals are selected for reproduction to produce the offspring of the next generation. It operates on a population of individuals, each representing a potential solution to the optimization problem. The main operators in GA include selection where individuals are selected based on their fitness, crossover where portions of two parent chromosomes are exchanged to create offspring, and mutation where a gene within a chromosome is randomly altered to introduce diversity. As such, GA is well-suited for both combinatorial and continuous optimization problems and is particularly effective for high-dimensional and complex search spaces. However, GA can be computationally expensive and may require careful tuning of parameters. In our implementation, GA used a population size of 50, a mutation probability of 0.05, and uniform crossover with probability 0.9.

PSO is a population-based optimization algorithm inspired by the social behavior of birds flocking or fish schooling. In PSO, candidate solutions are represented as particles within a search space. Each particle has a position and velocity, which corresponds to a potential solution and the rate of movement toward new solutions. The particles adjust their positions based on their own best-known position and the best-known position of the entire swarm. We used local coefficient $c_\ell = 2.05$, global coefficient $c_g = 2.05$, inertia weight $\omega = 0.4$, and a swarm size of 50.

Finally, SA is an optimization technique inspired by the annealing process in metallurgy, where the temperature of a material is slowly lowered to minimize defects and achieve a low-energy state. SA operates by iteratively generating new solutions in the search space and accepting them based on a probability function that depends on the difference in energy or objective function value between the

current and new solution, and the current "temperature". Traditional SA, while effective in avoiding local minima through its probabilistic acceptance of worse solutions, operates using a single candidate solution that updates iteratively. This single-agent approach often results in slow convergence, as the algorithm relies on stochastic movements to explore the search space. SSA is a hybrid optimization algorithm that integrates swarm intelligence techniques with the probabilistic nature of SA to efficiently search for global optima in complex optimization problems. By incorporating the cooperative behavior of swarm-based algorithms and the temperature-based exploration mechanism of SA, SSA enhances both global exploration and local exploitation, making it a robust method for solving high-dimensional optimization challenges. For SSA, we used initial temperature $t_0 = 1000$, final temperature $t_f = 1$, maximum sub-iterations per temperature cycle 5, move count per individual 5, mutation rate 0.1, mutation step size 0.1, and mutation step size damp 0.99.

These methodological choices were designed to enhance the stability and reliability of the inverse reconstruction. By reducing the dimensionality of the problem through a parametric inclusion model, employing diversity-preserving global optimizers to avoid premature convergence, introducing controlled noise into the boundary measurements during inversion to improve robustness, and performing multiple stochastic reconstructions to quantify uncertainty, we established a framework that mitigates the inherent ill-posedness of EIT.

3.4. Proposed algorithm and its implementation

In summary, we proposed two numerical methods for EIT image reconstruction: FNN-DE and CNN-DE algorithms.

We first present the details of the proposed FNN-DE algorithm as follows.

1) *Forward problem:* Train an FNN model that takes as input a set of geometry and conductivity parameters to predict the corresponding electric potential distribution φ_{FNN}.

2) *Inverse problem:* Given a boundary voltage data $\overline{V}$, solve minimization problem using differential evolution. The procedure is as follows:

   (a) Specify lower and upper bounds for all geometric and conductivity parameters to define the feasible search space.

   (b) Randomly generate an initial population of K candidate parameter vectors $\{\tilde{\Theta}_1, \ldots, \tilde{\Theta}_K\}$, where each $\tilde{\Theta}_i$ represents a possible configuration of inclusion geometry and conductivity.

   (c) Apply differential evolution operators to generate K trial solution vectors $\{\tilde{\Phi}_1, \ldots, \tilde{\Phi}_K\}$.

   (d) For each candidate vector $\tilde{\Theta}_i$ and trial vector $\tilde{\Phi}_i$, compute the corresponding electric potential distributions $\varphi_{\text{FNN}}(x; \tilde{\Theta}_i)$ and $\varphi_{\text{FNN}}(x; \tilde{\Phi}_i)$ using the pretrained feedforward neural network model.

   (e) Extract the predicted boundary voltages $\mathcal{F}_{\text{FNN}}(\tilde{\Theta}_i)$ and $\mathcal{F}_{\text{FNN}}(\tilde{\Phi}_i)$ at boundary nodes.

   (f) Compute the objective function values $\mathcal{J}_{\text{NN}}(\tilde{\Theta}_i)$ and $\mathcal{J}_{\text{NN}}(\tilde{\Phi}_i)$ and retain the better-performing vector for the next generation.

   (g) Repeat steps (c) to (f) and stop after $T(\tau)$ function evaluations, where $\tau = \dim(\Theta)$ is the number of unknown parameters.

(h) The final output is the best-performing individual in the final population, which is $\text{argmin}_i \mathcal{J}_{\text{FNN}}(\tilde{\Theta}_i)$.

The flowchart of the proposed FNN-DE algorithm is shown in Figure 4. Moreover, we implemented variations of the proposed FNN-DE framework using the four DE-variants and three non-DE algorithms for the inverse problem. In particular, we compare the performance of FNN-SADE, FNN-JADE, FNN-SHADE, FNN-SAPDE, FNN-GA, FNN-PSO, and FNN-SSA with that of the base FNN-DE algorithm. The best-performing algorithm among these was then further compared with a CNN-based model and with the classical FEM-based method, both using the same optimization algorithm.

The CNN-DE algorithm follows the same structure as the FNN-DE approach, with the key difference being the use of a CNN for the forward model instead of an FNN. One additional step in this framework is the explicit computation of the conductivity distribution $\sigma(x)$ from the geometric and conductivity parameters Θ. This involves evaluating a linear combination of inclusion indicator functions to produce a 2D matrix representation of $\sigma(x)$. The boundary voltage is then extracted and compared with the measured data, and the parameters Θ are iteratively updated using differential evolution to minimize the objective function $\mathcal{J}_{\text{NN}}(\Theta)$, as in the FNN-DE implementation.

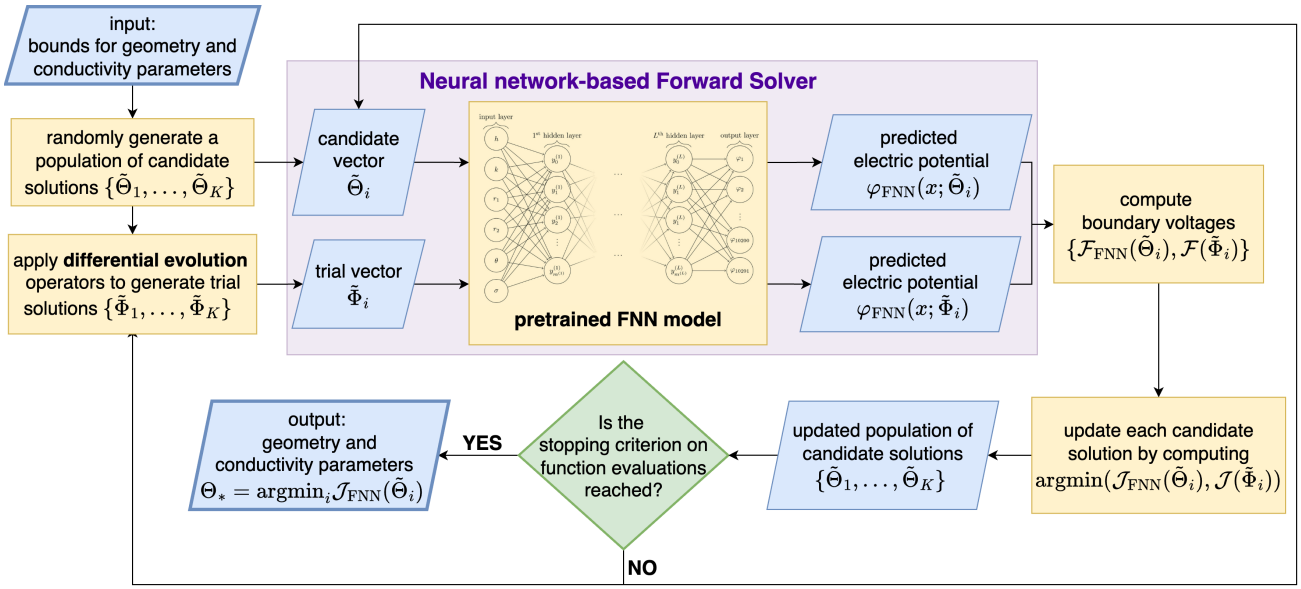

Figure 4. Flowchart of the proposed FNN-DE algorithm for the EIT problem.

We note here that direct numerical comparisons with other typical EIT reconstruction algorithms, such as those listed in the review paper [43], may not be meaningful due to different problem setups, cost functions, parameterizations, and assumptions. In practice, the major algorithm families vary in three main ways: the type of unknowns or parameterization (full-field conductivity, evolving boundaries, or low-dimensional parametric vectors), the priors or regularization imposed (global smoothness, total variation, edge-preserving, or hybrid penalties), and the optimization technique

(gradient-based versus gradient-free global search). Representative examples include methods minimizing penalized objectives over the full conductivity field using gradients, such as the foundational Newton's One-Step Error Reconstructor (NOSER) algorithm [44], improved Tikhonov regularization for lung monitoring [45], and homotopic mapping techniques [46]. Focusing on sparsity and edge preservation, other works employ total fractional-order variation [47] or modified orthogonal matching pursuit [48], while specialized solvers address fast iterative shrinkage [49] and region-of-interest estimation [50]. Distinct from these are Gauss–Newton approaches, which linearize the forward map and update the full field with Jacobians and regularization [51]. D-bar methods form another distinct group, solving a nonlinear transform with distinct regularization and no explicit misfit minimization. Specific studies have established regularized frameworks for this direct reconstruction method [52] and provided comprehensive guides to demystify its theoretical and practical application [53]. Shape reconstruction approaches alternatively evolve boundaries under piecewise-constant assumptions. Common implementations utilize level set representations combined with total variational regularization [54], or apply these techniques to experimental data for simultaneous conductivity and permittivity recovery [55]. Alternative geometric strategies employ monotonicity principles, particularly for lung imaging applications [56]. Finally, deep learning-based methods map measurements directly to images and are tied to the training distribution [24–26]. Recent surveys have comprehensively reviewed modern deep learning architectures and their implementations in this domain [57]. The implementation differences in these studies are not minor, but represent fundamentally different problem statements. These references illustrate that while qualitative comparisons can be made, e.g., the general accuracy of conductivity values or the ability to capture shapes, a fair comparison between the numerical methods is difficult to establish because each family of methods solves a different version of the EIT inverse problem under different modeling assumptions. For this reason, we focus our baseline on FEM-DE. As is common in the literature, FEM is used for solving the forward problem, while heuristic algorithms, such as DE, are employed when treating the inverse problem as an optimization task, as in [31].

3.4.1. Training

All geometry and conductivity parameters Θ were min-max normalized using the training set. For the CNN model, conductivity fields defined on the FEM mesh were rasterized to a grid using a piecewise-constant per-triangle assignment. Dataset splits were 70% training, 30% validation, with an additional independent test set comprising 10% of the total dataset.

To investigate the effect of data availability on model performance, models were trained using five different dataset sizes: 20%, 40%, 60%, 80%, and 100% of the generated dataset for each test configuration. For each dataset size, five random seeds $\{0, 1, 42, 100, 1234\}$ were used, and we report median performance across seeds.

All models were optimized using the Adam optimizer with a learning rate of 0.01 and `eps` set to 1×10^{-7}. Each model was trained for up to 500 epochs with a batch size of 128. A learning rate scheduler (`ReduceLROnPlateau`) was applied to automatically reduce the learning rate by a factor of 0.5 whenever the validation loss plateaued for five consecutive epochs. Early stopping was also implemented with a patience of 20 epochs, based on validation loss. The best-performing model on the validation set was saved and restored after training.

3.4.2. Optimization

For the inverse problem, each algorithm was executed independently 100 times with default parameters and a stopping criterion of $3000 \cdot \tau$ function evaluations, where τ denotes the dimensionality of the problem given by the number of parameters in the configuration vector Θ. Since each run uses a newly generated noisy dataset, this procedure effectively performs bootstrapping, and we record statistical measures across runs to quantify the associated uncertainties. Each run incorporates a randomly-generated noise with a mean of 0 and a standard deviation equal to 0.01 times the standard deviation of the true electric potential. In addition, we investigate varying the mesh size to verify that the method remains effective under different discretizations, and we perturb the injected currents, recording the corresponding estimated parameters as another form of bootstrapping. All global optimizers were implemented using the `mealpy` package [58].

3.4.3. Evaluation metrics

To assess the forward model performance, the predicted electric potential values were compared against reference solutions in the test data. Since EIT is an ill-posed inverse problem, small errors in the predicted electric potential values can lead to noticeable distortions when used to reconstruct the images. For the forward model evaluation, we consider four performance metrics: test MSE, which measures the average squared difference between predicted and true values; coefficient of determination (R^2), which indicates how well the predictions explain the variance in the true data; peak signal-to-noise ratio (PSNR), which assesses the quality of the predicted signal relative to noise; and structural similarity index (SSIM), which evaluates the perceived similarity between predicted and true images based on luminance, contrast, and structure. These metrics provide a comprehensive quantitative assessment of prediction accuracy and perceptual similarity. The reported metrics represent the median values across five independent runs with different random seeds (0, 1, 42, 100, 1234). Lower MSE and higher values of R^2, PSNR, and SSIM indicate better model performance. We also compare neural network model predictions of electric potential distribution with that of the FEM-based reference solutions, by computing the relative error, across all test configurations.

For the inverse problem, evaluation was based on MSE and intersection over union (IoU), measuring both numerical accuracy and geometric consistency in reconstructing conductivity distributions. Lower MSE and higher values of IoU indicate better performance. We also present overlay visualizations of the results along with the relative error of the best, average, and worst estimated conductivities obtained by each algorithm.

In addition to model performance, we also evaluate the computational cost of each optimization algorithm in terms of runtime, measured as the total time required to reach the maximum number of function evaluations. All runtimes were recorded using Python's built-in time module, capturing the duration of each run from initialization to termination. Each experiment was repeated 100 times, and the reported runtime reflects the average across these runs.

3.4.4. Hardware and software

The forward simulations were conducted on Windows 11 Home (Version 24H2), using an AMD Ryzen™ 9 8945HS CPU with Radeon™ 780M Graphics @ 4.00 GHz and 32 GB RAM. The implementation used Python 3.13 and PyTorch 2.6, with NumPy and Matplotlib for numerical

computation and visualization, respectively. On the other hand, the inverse simulations were run on Ubuntu 22.04, using an Intel(R) Xeon(R) Gold 6248 CPU @ 2.50 GHz (8 cores) and 64 GB RAM. Optimization was performed using the `mealpy` package [58], which supports population-based metaheuristic algorithms.

4. Results and discussion

This section presents the results of the proposed algorithms for solving the EIT forward and inverse problems. We report the performance of the FNN and CNN models in predicting electric potential distribution, as well as the reconstruction quality achieved using the pretrained neural network model with DE, its four variants (SADE, JADE, SHADE, SAPDE), and three other global optimization algorithms (GOAs): GA, PSO, and SSA. Comparisons are made against the FEM-based solutions to evaluate accuracy and consistency.

4.1. Forward problem

The loss curves of the forward FNN and CNN models during training and validation across four test configurations are illustrated in Figure 5. For both architectures, the training and validation losses rapidly decrease within the first few epochs and then stabilize, indicating convergence. The FNN shows consistently low validation loss across all cases with minimal overfitting. The CNN exhibits similar trends, although with slightly higher variance in the validation loss. Nonetheless, both models generalize well, with close alignment between training and validation losses.

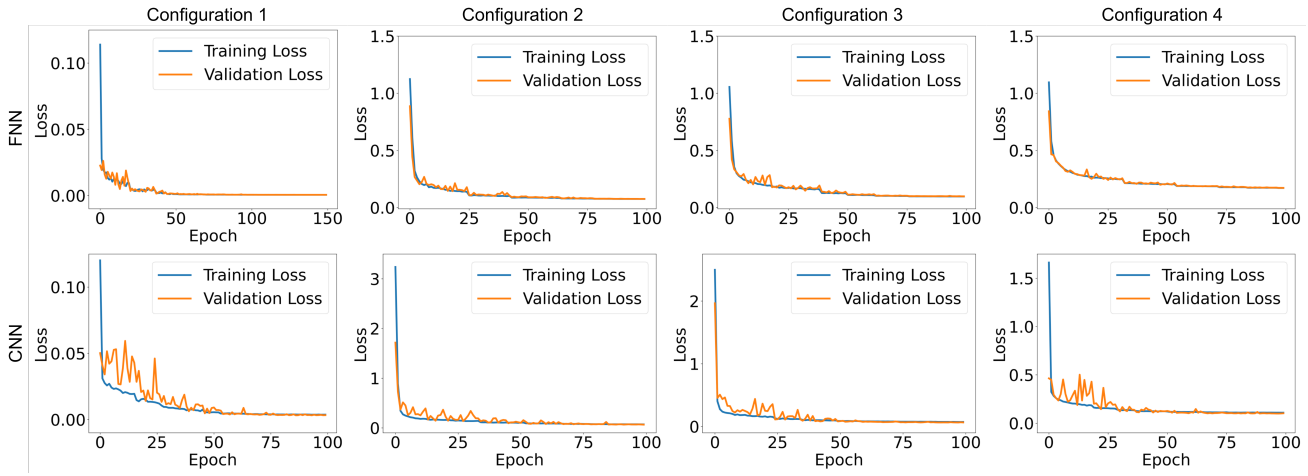

Figure 5. Training (blue) and validation (orange) loss curves of the forward FNN (top) and CNN (bottom) model across configurations.

Across all four configurations, Table 4 presents the median performance metrics obtained from five independent training runs with different random seeds. In general, increasing the size of the training dataset results in improved model performance, as evidenced by lower test loss, lower MSE, and higher

R^2, PSNR, and SSIM values. This trend highlights the benefit of larger datasets for generalization in the forward EIT problem.

Table 4. Performance metrics of the FNN model for the forward problem.

Config	Sample Size	Test Loss	MSE	R^2	PSNR	SSIM
1	30,000	6.56000×10^{-4}	1.76813×10^{-6}	0.99999958	67.87612	0.9999739
	24,000	6.96822×10^{-4}	1.79552×10^{-6}	0.99999952	67.18670	0.9999715
	18,000	7.30892×10^{-4}	2.26116×10^{-6}	0.99999928	67.11420	0.9999639
	12,000	8.59385×10^{-4}	3.01011×10^{-6}	0.99999893	65.54156	0.9999591
	6,000	12.70026×10^{-4}	5.43267×10^{-6}	0.99999583	63.28205	0.9999214
2	60,000	6.95620×10^{-2}	3.07537×10^{-2}	0.99945702	50.07821	0.9880054
	48,000	7.33158×10^{-2}	3.28295×10^{-2}	0.99942551	49.79371	0.9877824
	36,000	7.62693×10^{-2}	3.63851×10^{-2}	0.99936439	49.31757	0.9864775
	24,000	8.72567×10^{-2}	4.53511×10^{-2}	0.99920222	48.35665	0.9840954
	12,000	1.03628×10^{-1}	5.99776×10^{-2}	0.99894044	47.14246	0.9808856
3	90,000	9.41736×10^{-2}	3.78197×10^{-2}	0.99934579	49.12356	0.9876026
	72,000	1.00151×10^{-1}	4.20971×10^{-2}	0.99927461	48.66441	0.9863467
	54,000	1.08883×10^{-1}	4.82666×10^{-2}	0.99916137	48.08865	0.9845991
	36,000	1.12694×10^{-1}	5.17437×10^{-2}	0.99910799	47.77948	0.9839454
	18,000	1.44113×10^{-1}	7.74754×10^{-2}	0.99866457	46.01739	0.9779998
4	150,000	1.84685×10^{-1}	1.00296×10^{-1}	0.99817581	44.94544	0.9753957
	120,000	1.77801×10^{-1}	9.58364×10^{-2}	0.99825978	45.15244	0.9764413
	90,000	1.86633×10^{-1}	1.01305×10^{-1}	0.99816151	44.89740	0.9752029
	60,000	1.97430×10^{-1}	1.14836×10^{-1}	0.99790603	44.36085	0.9728754
	30,000	2.29354×10^{-1}	1.47003×10^{-1}	0.99731425	43.29089	0.9668762

However, the improvement in metrics is not strictly monotonic as the training set size increases. Minor fluctuations are observed in certain cases. This behavior can be attributed to the inherent stochasticity of neural network training, arising from random initialization and data shuffling, which can still lead to variability in final outcomes. Moreover, as the dataset grows, the optimization landscape may shift, which may cause the model to converge to different local minima, especially if the network's capacity is not optimally matched to the data complexity. As a result, the models may encounter difficulty in generalizing as more varied samples are introduced in more complex cases. This observation suggests that expanding the range of hyperparameters considered during tuning may be necessary to fully adapt the model to the increased data complexity and to achieve optimal performance at larger dataset sizes.

Overall, while the general trend affirms the advantage of larger datasets for improved predictive accuracy, there are observed minor inconsistencies due to the combined effects of stochastic optimization and increasing data complexity.

The performance of the CNN model for the forward problem is summarized in Table 5. Across all four geometric cases, the improvement in performance metrics with increasing dataset size was strictly monotonic. This consistent trend suggests that a CNN model with a U-net architecture is particularly well-suited for learning spatial features and benefits directly from the availability of more training

data. When compared to the FNN model performance in Table 4, CNNs generally achieve lower test loss and comparable or higher PSNR values across most configurations. Although FNNs sometimes attain marginally better R^2 scores due to their direct approximation of FEM coefficients, CNNs provide reconstructions with higher structural fidelity, as reflected in their superior SSIM values.

Table 5. Performance metrics of the CNN model for the forward problem.

Config	Sample Size	Test Loss	MSE	R^2	PSNR	SSIM
1	30,000	3.38994×10^{-3}	4.78114×10^{-5}	0.99990	51.55488	0.99946
	24,000	3.94193×10^{-3}	6.50955×10^{-5}	0.99988	50.98160	0.99937
	18,000	4.18450×10^{-3}	8.32644×10^{-5}	0.99981	49.45207	0.99933
	12,000	5.85120×10^{-3}	25.25968×10^{-5}	0.99954	46.65135	0.99903
	6,000	9.66159×10^{-3}	63.74875×10^{-5}	0.99850	40.88564	0.99645
2	60,000	4.90948×10^{-2}	1.01127×10^{-2}	0.99985	55.72820	0.99692
	48,000	5.34322×10^{-2}	1.05303×10^{-2}	0.99984	55.20704	0.99676
	36,000	5.62540×10^{-2}	1.25015×10^{-2}	0.99981	54.79769	0.99609
	24,000	6.12718×10^{-2}	1.44612×10^{-2}	0.99977	54.07781	0.99560
	12,000	7.97138×10^{-2}	2.31482×10^{-2}	0.99961	52.22923	0.99378
3	90,000	6.90441×10^{-2}	1.70230×10^{-2}	0.99975	53.45060	0.99516
	72,000	6.97484×10^{-2}	1.72943×10^{-2}	0.99975	53.43304	0.99519
	54,000	7.47295×10^{-2}	1.95871×10^{-2}	0.99970	52.78146	0.99435
	36,000	8.38660×10^{-2}	2.30337×10^{-2}	0.99964	52.11940	0.99394
	18,000	9.90166×10^{-2}	3.11384×10^{-2}	0.99949	50.58530	0.99194
4	150,000	9.46501×10^{-2}	2.93625×10^{-2}	0.99952	50.73349	0.99304
	120,000	9.60641×10^{-2}	2.98090×10^{-2}	0.99951	50.59544	0.99267
	90,000	10.09884×10^{-2}	3.26450×10^{-2}	0.99945	50.24608	0.99221
	60,000	11.20994×10^{-2}	3.90168×10^{-2}	0.99932	49.58626	0.99101
	30,000	13.19443×10^{-2}	5.13409×10^{-2}	0.99908	48.27336	0.98836

Overall, CNNs outperform FNNs in the configurations 2–4, achieving lower test loss, higher PSNR, and better SSIM scores across all dataset sizes. These configurations involve simple or moderately complex geometries, where CNNs trained on structured grids effectively capture spatial features. In contrast, FNNs perform better in the first configuration, where the domain is highly irregular. Here, the use of FEM basis functions in FNNs provides a natural alignment with the unstructured mesh, while the CNNs rely on interpolated data mapped onto a regular grid, which can introduce artifacts and reduce accuracy near complex boundaries.

To further evaluate the accuracy of the predictions, Figure 6 shows the pointwise relative error between the neural network predictions and the FEM-based reference solutions across all configurations. For both FNN and CNN models, the relative error is low across most of the domain, indicating strong agreement with the FEM solution. However, error is typically higher near interfaces between regions with differing conductivities. These are the areas where both architectures face challenges due to sharp transitions.

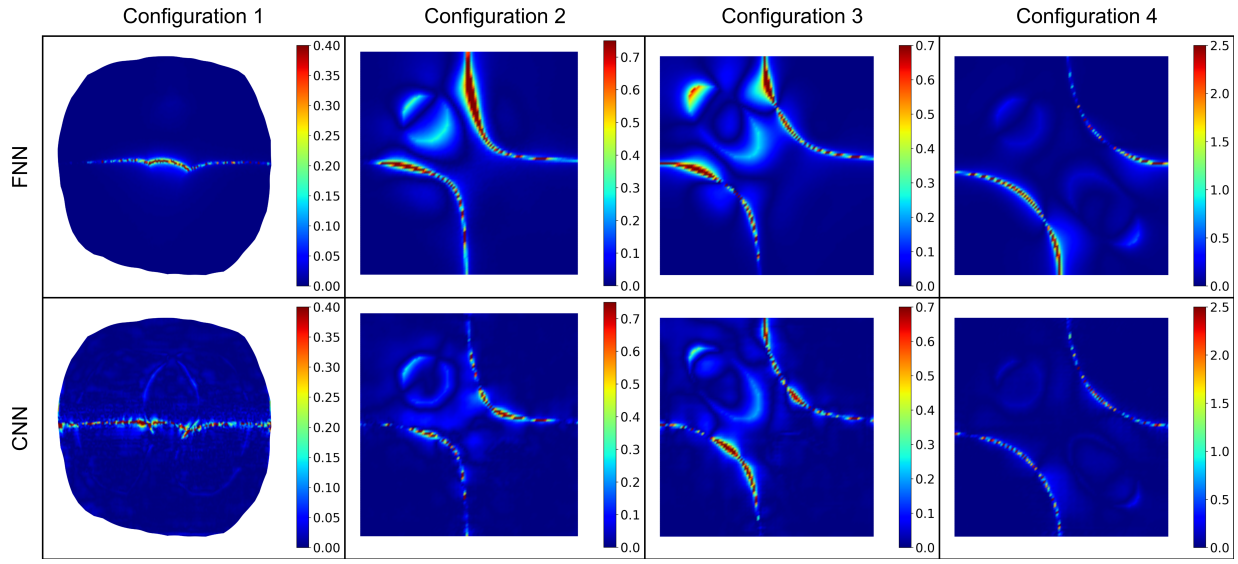

Figure 6. Pointwise relative errors of forward solutions obtained using FNN (top) and CNN (bottom) against FEM solutions for all configurations.

In the first configuration, which represents a thorax model with lung and heart inclusions, the agreement between the FEM and FNN results is particularly strong. The FNN model captures the details with high accuracy, resulting in almost identical outputs. This suggests that FNN can effectively model complex anatomical structures, highlighting its potential for biomedical applications. The CNN, in contrast, shows visibly higher errors across the domain, particularly near the center. This is likely due to interpolation artifacts introduced by mapping data onto a structured grid.

In the remaining test configurations, however, the CNN outperforms the FNN, producing lower errors. These configurations show that CNNs effectively learn spatial patterns from grid-aligned data. The FNNs, while still generally accurate, exhibit higher errors near conductivity transitions. This suggests that CNNs are well-suited for structured domains, whereas FNNs maintain an advantage in domains with irregular geometries, as their outputs are expressed in the FEM basis, eliminating the need for interpolation onto a structured grid.

Overall, both models provide accurate forward predictions, with CNNs excelling in structured domains and FNNs showing advantages in irregular domains. The primary limitation observed in both architectures is related to accuracy at conductivity interfaces and geometrically complex regions, areas where further refinement in network training or architecture could potentially enhance performance.

4.2. Comparison of global optimization algorithms for the inverse EIT

The goal of the inverse problem is to estimate both the geometric parameters (e.g., location, size, orientation) and the conductivity values of internal inclusions from boundary voltage measurements. In this section, we first focus on evaluating how accurately each GOA recovers the true conductivity values across the four test configurations.

To assess this, we report the relative errors between the true and estimated conductivity values obtained using each GOA in combination with the pretrained FNN forward model.

Table 6. Relative error of conductivity values for test configuration 1. For the average and best-case errors, blue values indicate the top three per configuration; the best is also **bold**.

	σ_1-Relative Error			σ_2-Relative Error		
Algorithm	Average	Best	Worst	Average	Best	Worst
FNN-DE	1.38×10^{-2}	2.56×10^{-4}	$\mathbf{3.76 \times 10^{-2}}$	3.20×10^{-2}	$\mathbf{1.65 \times 10^{-4}}$	$\mathbf{9.27 \times 10^{-2}}$
FNN-SADE	1.22×10^{-2}	1.81×10^{-4}	4.48×10^{-2}	2.88×10^{-2}	4.63×10^{-4}	1.05×10^{-1}
FNN-JADE	1.46×10^{-2}	3.54×10^{-4}	4.81×10^{-2}	3.46×10^{-2}	2.66×10^{-4}	1.29×10^{-1}
FNN-SHADE	$\mathbf{1.17 \times 10^{-2}}$	1.41×10^{-4}	3.91×10^{-2}	$\mathbf{2.79 \times 10^{-2}}$	8.67×10^{-4}	1.06×10^{-1}
FNN-SAPDE	1.10×10^{-1}	1.31×10^{-4}	1.88×10^{-1}	2.38×10^{-1}	5.36×10^{-3}	5.28×10^{-1}
FNN-GA	8.98×10^{-2}	7.43×10^{-2}	9.27×10^{-2}	2.35×10^{-1}	1.88×10^{-1}	2.73×10^{-1}
FNN-PSO	2.06×10^{-2}	$\mathbf{5.53 \times 10^{-5}}$	7.61×10^{-2}	5.25×10^{-2}	6.80×10^{-4}	2.30×10^{-1}
FNN-SSA	1.50×10^{-1}	1.96×10^{-2}	1.85×10^{-1}	5.29×10^{-1}	3.96×10^{-2}	5.87×10^{-1}

Table 6 reports the relative errors for test configuration 1, where the true conductivities are $\sigma_1 = 1$ and $\sigma_2 = 6.3$. Among the proposed algorithms, FNN-SADE, FNN-JADE, FNN-SHADE, and FNN-PSO consistently achieved near-perfect values across all metrics, indicating stable performance. FNN-DE also performed well but showed minor deviations, while FNN-SAPDE, FNN-GA, and FNN-SSA resulted in larger errors.

For configuration 2, where the true conductivity is $\sigma_1 = 7$, Table 7 shows that FNN-DE achieved the lowest average and worst-case relative errors, indicating stable performance. While FNN-SSA attained the lowest best-case error, it has a higher variability across trials. FNN-SAPDE followed closely, with competitive best and average errors but slightly less consistency in the worst case. FNN-GA produced a strong best-case result, but its higher average error suggests occasional poor convergence. In contrast, FNN-SADE, FNN-JADE, FNN-SHADE, and FNN-PSO all converged to nearly identical values close to the lower bound of the search range, resulting in uniformly high relative errors across trials, indicating poor exploration of the solution space.

Table 7. Relative error of conductivity values in test configurations 2 and 3. For the average and best-case errors, blue values indicate the top three per configuration; the best is also **bold**.

	Configuration 2 σ_1-Relative Error			Configuration 3 σ_1-Relative Error		
Algorithm	Average	Best	Worst	Average	Best	Worst
FNN-DE	2.50×10^{-1}	2.80×10^{-2}	2.86×10^{-1}	1.92×10^{-1}	6.84×10^{-4}	2.86×10^{-1}
FNN-SADE	1.41×10^{-1}	2.53×10^{-3}	2.86×10^{-1}	1.39×10^{-1}	6.20×10^{-3}	2.86×10^{-1}
FNN-JADE	1.61×10^{-1}	1.95×10^{-3}	2.86×10^{-1}	1.32×10^{-1}	2.41×10^{-3}	2.86×10^{-1}
FNN-SHADE	1.54×10^{-1}	1.84×10^{-3}	2.86×10^{-1}	1.37×10^{-1}	$\mathbf{1.86 \times 10^{-4}}$	2.86×10^{-1}
FNN-SAPDE	1.78×10^{-1}	8.18×10^{-3}	2.86×10^{-1}	2.03×10^{-1}	6.83×10^{-3}	2.86×10^{-1}
FNN-GA	1.50×10^{-1}	4.91×10^{-3}	2.80×10^{-1}	1.54×10^{-1}	8.97×10^{-3}	2.86×10^{-1}
FNN-PSO	1.17×10^{-1}	9.77×10^{-5}	2.76×10^{-1}	$\mathbf{1.31 \times 10^{-1}}$	8.03×10^{-3}	$\mathbf{2.81 \times 10^{-1}}$
FNN-SSA	2.79×10^{-1}	1.84×10^{-1}	2.86×10^{-1}	2.86×10^{-1}	2.86×10^{-1}	2.86×10^{-1}

Table 8. Relative error of conductivity values in test configuration 4. For the average and best-case errors, blue values indicate the top three per configuration; the best is also **bold**.

	σ_1-Relative Error			σ_2-Relative Error		
Algorithm	Average	Best	Worst	Average	Best	Worst
FNN-DE	1.42×10^{-1}	5.14×10^{-3}	5.00×10^{-1}	9.87×10^{-2}	2.68×10^{-2}	1.99×10^{-1}
FNN-SADE	2.30×10^{-1}	2.84×10^{-3}	5.00×10^{-1}	1.29×10^{-1}	$\mathbf{2.00 \times 10^{-4}}$	3.75×10^{-1}
FNN-JADE	1.99×10^{-1}	$\mathbf{4.19 \times 10^{-4}}$	5.00×10^{-1}	1.46×10^{-1}	1.50×10^{-3}	3.72×10^{-1}
FNN-SHADE	2.13×10^{-1}	1.35×10^{-3}	$\mathbf{4.83 \times 10^{-1}}$	1.14×10^{-1}	2.02×10^{-4}	3.70×10^{-1}
FNN-SAPDE	2.73×10^{-1}	1.32×10^{-3}	5.00×10^{-1}	1.83×10^{-1}	2.44×10^{-3}	3.75×10^{-1}
FNN-GA	2.29×10^{-1}	8.38×10^{-3}	4.99×10^{-1}	1.53×10^{-1}	2.58×10^{-3}	3.75×10^{-1}
FNN-PSO	1.64×10^{-1}	7.27×10^{-4}	4.90×10^{-1}	1.22×10^{-1}	2.90×10^{-3}	3.41×10^{-1}
FNN-SSA	$\mathbf{1.18 \times 10^{-1}}$	1.10×10^{-1}	5.00×10^{-1}	$\mathbf{4.16 \times 10^{-2}}$	3.99×10^{-2}	$\mathbf{1.25 \times 10^{-1}}$

For configuration 3 (Table 7), with a single inclusion of true conductivity $\sigma_1 = 7$, FNN-DE achieved the most accurate average and best-case estimates but showed variability in the worst case. FNN-JADE and FNN-SAPDE also achieved accurate best cases but had large worst-case errors. On the other hand, FNN-SADE, FNN-SHADE, FNN-GA, FNN-PSO, and FNN-SSA frequently converged to boundary values, yielding high average errors.

In configuration 4 (Table 8), where the goal is to simultaneously recover $\sigma_1 = 6$ and $\sigma_2 = 8$, the best conductivity estimates were close to the true solution for most algorithms. However, the worst estimates often hit the endpoints of the search space. The average conductivity pairs from the algorithms deviated slightly from the true values, reflecting the difficulty in consistently recovering both conductivity values accurately.

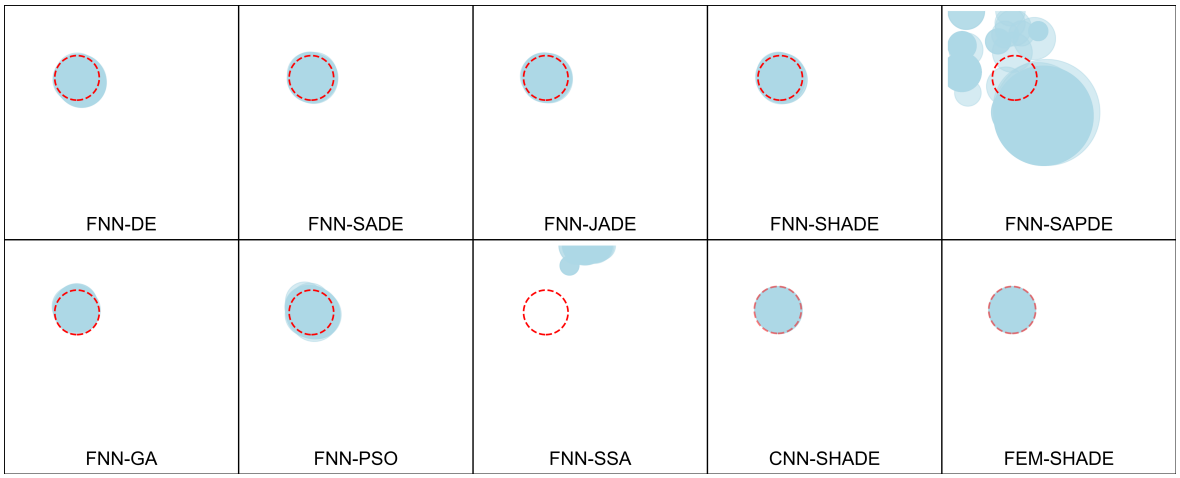

Figure 7. Overlay of predicted inclusions for configuration 2. The red dashed lines represent the true location and size.

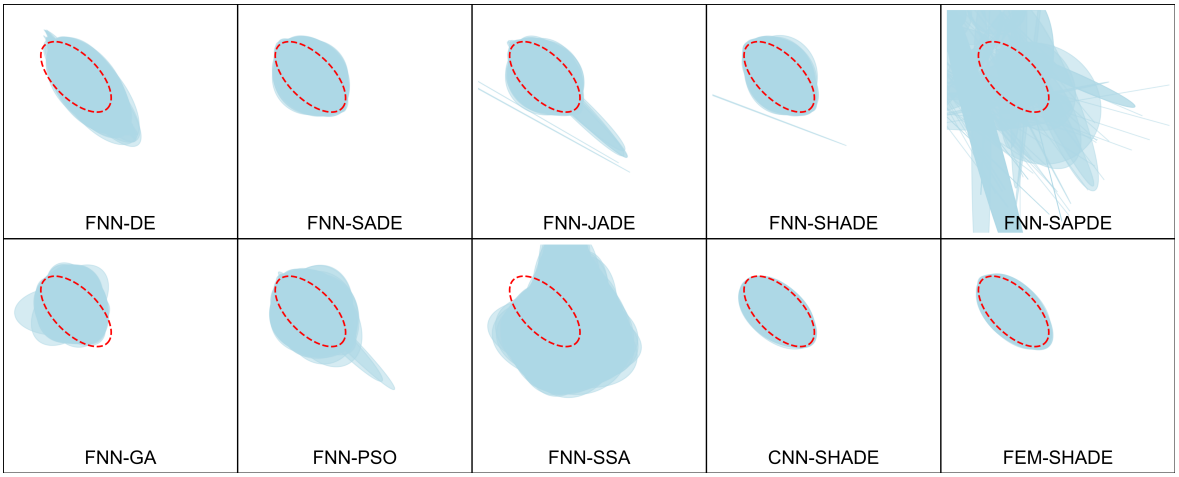

Figure 8. Overlay of predicted inclusions for configuration 3. The red dashed lines represent the true location and size.

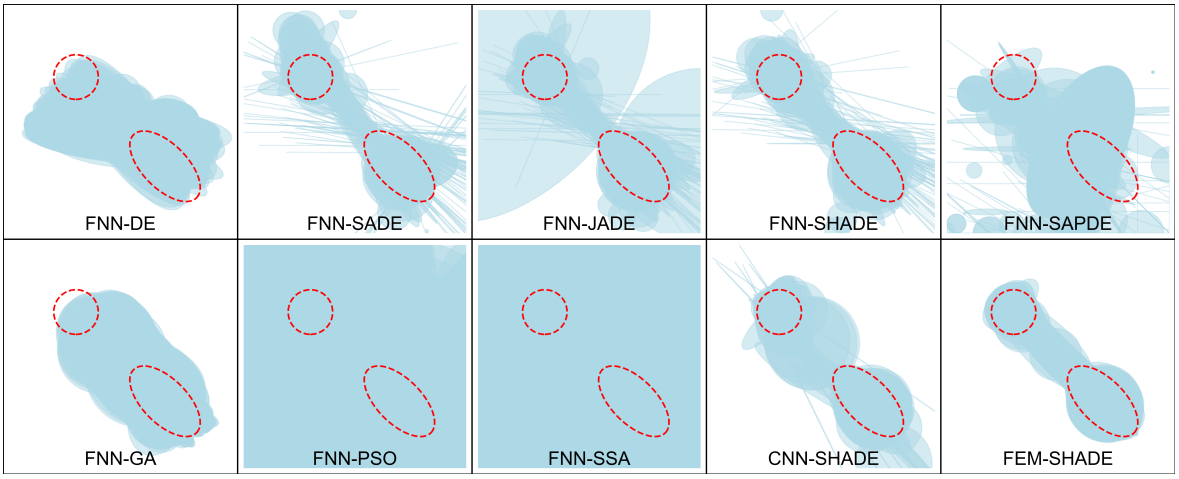

Figure 9. Overlay of predicted inclusions for configuration 4. The red dashed lines represent the true location and size.

While these tables show how accurately the proposed algorithms can capture the conductivity values, these metrics do not fully capture the spatial quality of reconstructions. The geometry of the inclusions also plays a crucial, if not more significant, role. The overlay plots further illustrate how well the algorithms recover the geometric features of the true conductivity distribution. In first configuration, the geometry was fixed, and only the conductivity values varied. Figures 7–9 show the overlay of the predicted geometries for each algorithm across the last three test configurations with the red dashed lines representing the true inclusions. These overlays provide insight into how well each GOA, coupled with the pretrained FNN model, can recover not only the conductivity values but also the geometric features of the inclusions.

In general, FNN-JADE, FNN-SADE, and FNN-SHADE produced the most accurate geometric reconstructions, with the predicted shapes closely matching the true boundaries. The overlays for FNN-

PSO also show reasonable agreement but with some runs experiencing more geometric distortions. FNN-DE and FNN-GA showed more spread and less precision in shape recovery, while FNN-SAPDE and FNN-SSA consistently produced scattered and inaccurate geometric reconstructions, highlighting their poor performance.

For the second configuration, the inclusions recovered by most algorithms were relatively accurate, with FNN-JADE, FNN-SADE, and FNN-SHADE yielding the cleanest match to the ground truth. However, as the geometric complexity increased, differences among the algorithms became more pronounced. In the most complex case, FNN-SHADE remained the most consistent, while FNN-SSA and FNN-SAPDE struggled, producing significant deviations and noise.

Table 9 presents the average MSEs between the true and predicted conductivity distributions for each algorithm across four test cases. In the first configuration, FNN-SADE achieved the lowest MSE, followed by FNN-SHADE and FNN-PSO. All three produced near machine-precision errors, indicating highly accurate reconstructions. FNN-JADE also performed well but ranked lower. FNN-DE and FNN-SAPDE had moderate errors, while FNN-GA and FNN-SSA showed the highest.

Table 9. Average MSE between the true and predicted conductivity distributions for each algorithm across all configurations. Blue values indicate the top three per configuration; the best is also **bold**.

Algorithm	Configuration 1	Configuration 2	Configuration 3	Configuration 4
FNN-DE	4.36480×10^{-3}	5.12737×10^{-1}	1.65458×10^{0}	1.45083×10^{1}
FNN-SADE	3.72352×10^{-3}	2.33875×10^{-1}	9.24779×10^{-1}	4.51386×10^{0}
FNN-JADE	5.57131×10^{-3}	$\mathbf{2.29579 \times 10^{-1}}$	8.03804×10^{-1}	4.45917×10^{0}
FNN-SHADE	$\mathbf{3.28988 \times 10^{-3}}$	2.32966×10^{-1}	$\mathbf{6.99476 \times 10^{-1}}$	$\mathbf{4.32549 \times 10^{0}}$
FNN-SAPDE	1.96785×10^{-1}	2.52605×10^{0}	3.64952×10^{0}	6.40001×10^{0}
FNN-GA	1.50517×10^{-1}	4.42862×10^{-1}	1.33081×10^{0}	7.64654×10^{0}
FNN-PSO	1.34703×10^{-2}	3.58233×10^{-1}	1.42353×10^{0}	2.89318×10^{1}
FNN-SSA	8.09906×10^{-1}	2.19800×10^{0}	9.75441×10^{0}	5.57908×10^{1}
CNN-SHADE	6.51363×10^{-4}	1.66176×10^{-1}	4.15736×10^{-1}	2.93994×10^{0}
FEM-SHADE	2.32199×10^{-3}	2.28066×10^{-1}	3.85457×10^{-1}	2.81672×10^{0}

In the second configuration, FNN-JADE, FNN-SADE, and FNN-SHADE were the top performers. FNN-PSO and FNN-DE produced moderate errors. FNN-SAPDE, FNN-GA, and FNN-SSA had significantly higher MSEs, with FNN-SSA performing worst.

In the third configuration, FNN-SHADE ranked first, followed by FNN-JADE and FNN-SADE. FNN-PSO and FNN-DE produced acceptable results, while FNN-SAPDE and FNN-SSA had the largest errors. Finally, in the fourth configuration, FNN-SAPDE achieved the lowest MSE, followed by FNN-SHADE and FNN-SADE. FNN-JADE and FNN-PSO showed moderate accuracy. FNN-DE and FNN-GA had larger errors, and FNN-SSA again recorded the highest. It is important to note that FNN-SAPDE achieved the lowest MSE even though it failed to correctly identify several inclusions—most predicted radii were close to zero (i.e., $< 10^{-5}$), effectively omitting them. This apparent contradiction arises because the MSE metric evaluates the difference between predicted and true conductivity values across the domain, not the presence or absence of geometric structures. In this case, since inclusions occupy a relatively small area, omitting them can still result in a low

average error across the entire domain. However, this comes at the cost of geometric fidelity.

Overall, among all algorithms evaluated, FNN-SHADE consistently demonstrated strong performance across all test configurations. It ranked within the top three in every configuration and achieved the lowest MSE in configuration 3, and second-lowest in configurations 1 and 4. This consistency underscores the robustness of FNN-SHADE in handling a range of conductivity distributions, including both simple and complex scenarios. FNN-SADE also performed strongly, achieving the lowest MSE in configuration 1. FNN-JADE ranked in the top three in two out of four configurations, indicating good but slightly less consistent performance compared to FNN-SHADE and FNN-SADE. In contrast, FNN-SSA consistently recorded the highest MSE values in all test configurations, indicating limited reconstruction accuracy and suggesting poor adaptability to varying conductivity profiles. We attribute this weaker performance of SAPDE and SSA to the aggressive self-adaptation and cooling schedules used by these algorithms. When parameters such as population size or temperature are adapted too quickly, the resulting loss of population diversity in early generations can drive premature convergence and collapse to suboptimal or trivial solutions. This effect is amplified in EIT because the objective surface contains large, flat regions where insufficient diversity prevents the optimizer from escaping uninformative areas of the search space. Moreover, using a neural forward surrogate introduces small approximation errors relative to FEM. In variants that rely heavily on parameter adaptation, these perturbations can shift the search process toward regions of the solution space that contain low-quality or misleading solutions. In contrast, SADE, JADE, and SHADE combine success-based parameter adaptation with explicit diversity mechanisms, sustaining exploration on the broad plateaus of the EIT objective function. The other optimizers we tested—original DE, GA, and PSO—also maintain more robust diversity, which allows ongoing exploration even late in the search and makes them less prone to collapse on the highly flat landscape of the EIT inverse problem. These differences mainly reflect the stochastic nature of evolutionary optimization. Different global optimization algorithms balance exploration and exploitation differently, leading some to stable reconstruction and others to premature convergence.

While MSE provides insight into overall electric potential prediction accuracy, it does not fully capture how well the shape and location of internal inclusions are reconstructed. To address this, we evaluate each algorithm using the IoU metric. Table 10 presents the average IoU scores for each algorithm across configurations 2 to 4. These values quantify the geometric accuracy of the predicted inclusions relative to the ground truth, that is, how well each algorithm recovers the shape and location of the inclusions.

As shown in the Table 10, the average IoU for FNN-SAPDE for configuration 4 was only 0.03126, indicating that it failed to capture the inclusion shapes and locations. In contrast, FNN-SHADE achieved a higher MSE but correctly reconstructed the inclusion geometry, yielding the highest IoU score of 0.66668. This geometric accuracy is critical in many practical applications.

Among the FNN-based methods, FNN-SHADE achieved the highest IoU scores in configurations 3 and 4, and was nearly tied for best in configuration 2, confirming its superior capability in geometric reconstruction. In contrast, algorithms like FNN-SAPDE and FNN-SSA recorded significantly lower IoUs, indicating poor inclusion localization. While FNN-SADE and FNN-JADE also yielded competitive results, their performance was slightly less consistent across configurations.

Table 10. Average IoU scores for each algorithm across configurations 2–4. Higher IoU indicates better geometric accuracy in identifying inclusions. Blue values indicate the top three among the proposed FNN-based algorithms per configuration; the best is also **bold**.

Algorithm	Configuration 2	Configuration 3	Configuration 4
FNN-DE	0.83193	0.53134	0.14784
FNN-SADE	**0.89785**	0.78256	0.26304
FNN-JADE	0.89495	0.79737	**0.31893**
FNN-SHADE	0.89239	**0.82861**	0.29308
FNN-SAPDE	0.03160	0.03664	0.06336
FNN-GA	0.75788	0.65680	0.28147
FNN-PSO	0.80958	0.66987	0.15536
FNN-SSA	0.00000	0.27418	0.09397
CNN-SHADE	0.92415	0.89331	0.57278
FEM-SHADE	0.87885	0.90709	0.57317

As illustrative examples, Figures 10 to 13 present the reconstructions with the lowest MSEs for each test configuration. For the configuration 1 with fixed geometry, the conductivity has values that closely match the ground truth. For configurations 2 and 3, the recovered inclusion's location, size, shape, and conductivity from most algorithms agree closely with the targets, with residuals concentrated only along thin boundary interfaces. On the other hand, for configuration 4, only FNN-SADE, FNN-JADE, FNN-SHADE, CNN-SHADE, and FEM-SHADE were able to recover the approximate location, size, shape, and conductivity values of both the inclusions. These examples are consistent with the findings in Tables 9 and 10.

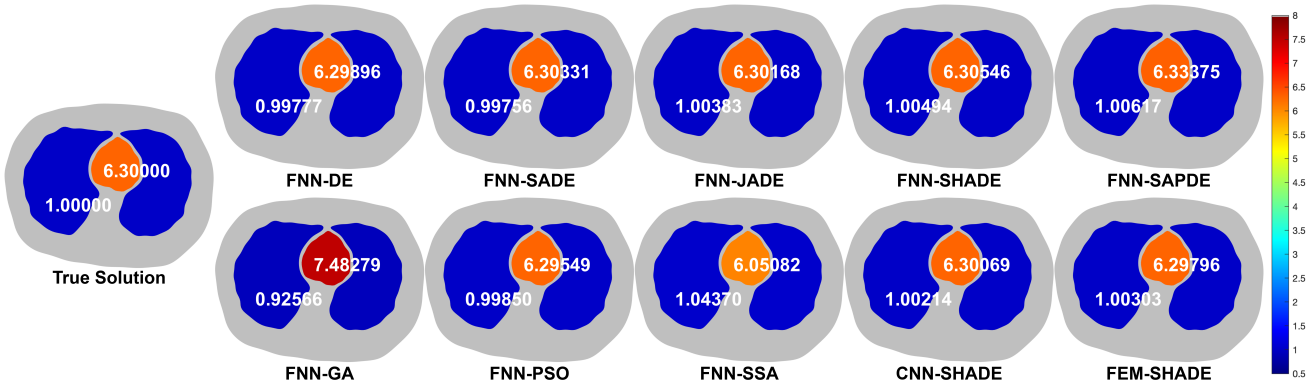

Figure 10. Reconstructions with the lowest MSE value for configuration 1.

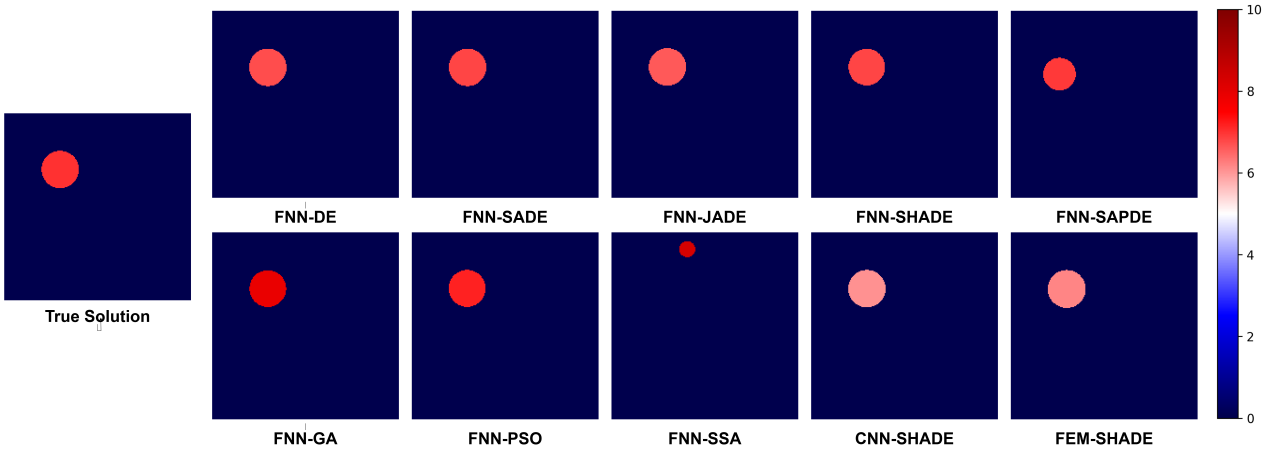

Figure 11. Reconstructions with the lowest MSE value for configuration 2.

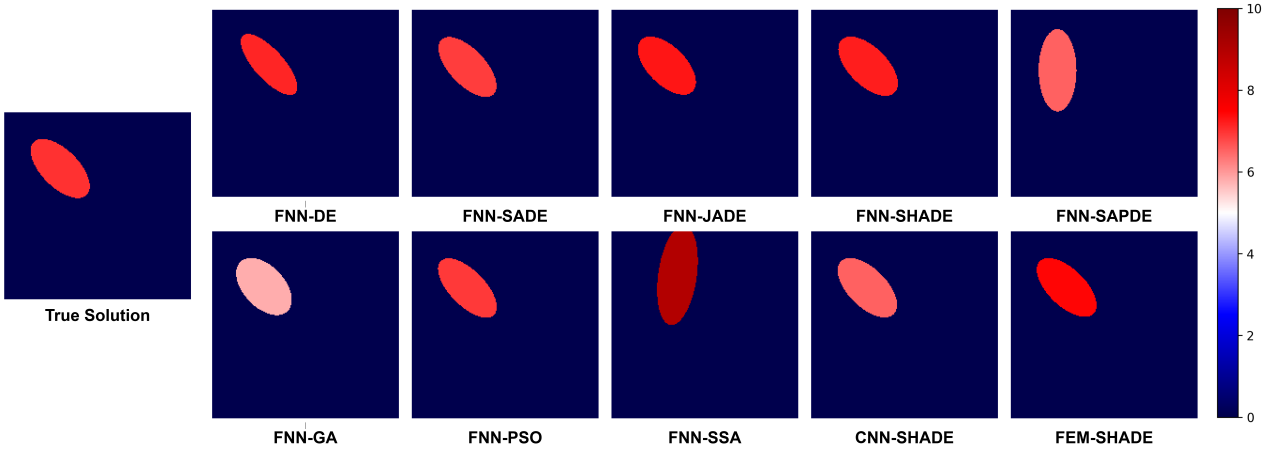

Figure 12. Reconstructions with the lowest MSE value for configuration 3.

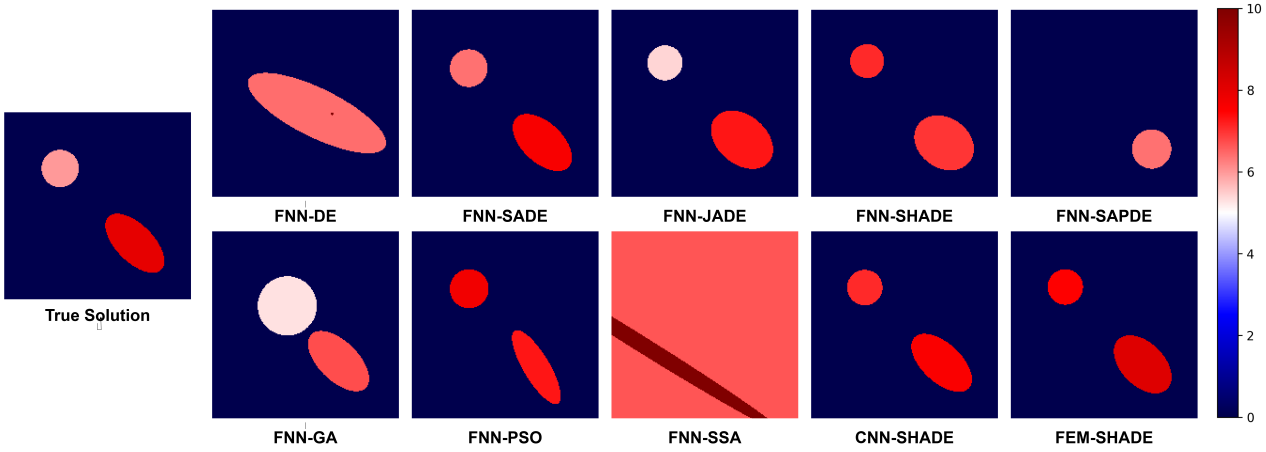

Figure 13. Reconstructions with the lowest MSE value for configuration 4.

Table 11 presents the computational time (in seconds) required to solve the EIT inverse problem using different methods across all configurations. Among these, FEM-SHADE consistently required the highest computational cost. The results clearly show that the neural network-based models, combined with various GOAs, offer significant computational advantages. Across all configurations, the FNN-based methods required only around 6 to 38 seconds depending on the complexity of the configuration and the specific GOA used. CNN-SHADE, which incorporates convolutional layers, was considerably slower—ranging from 69 to 125 seconds—due to the added computational overhead. While CNN-SHADE achieves good accuracy, it is consistently outpaced by FNN-SHADE in runtime. In the same vein, FEM-SHADE, which directly solves the forward problem using the finite element method, required substantially more time—reaching up to 1618 seconds in configuration 4.

Table 11. Runtimes (in seconds) taken in solving the EIT inverse problem using variants of the proposed algorithm. Blue values indicate the top three algorithms per configuration; the best is also **bold**.

Algorithm	Configuration 1	Configuration 2	Configuration 3	Configuration 4
FNN-DE	6.59874	14.51166	23.82091	34.88486
FNN-SADE	6.72153	14.54692	23.88585	35.82462
FNN-JADE	7.59279	15.74757	25.22329	37.75088
FNN-SHADE	7.99827	16.08827	25.46233	37.69826
FNN-SAPDE	7.33459	15.14424	25.00183	36.85576
FNN-GA	7.95353	14.85899	24.25379	35.51946
FNN-PSO	**6.59445**	**14.16006**	**23.10202**	34.35080
FNN-SSA	6.96922	14.86165	24.04540	**34.02391**
CNN-SHADE	96.59271	69.17885	77.48984	124.02645
FEM-SHADE	1346.48680	942.14251	1560.33578	1618.48402

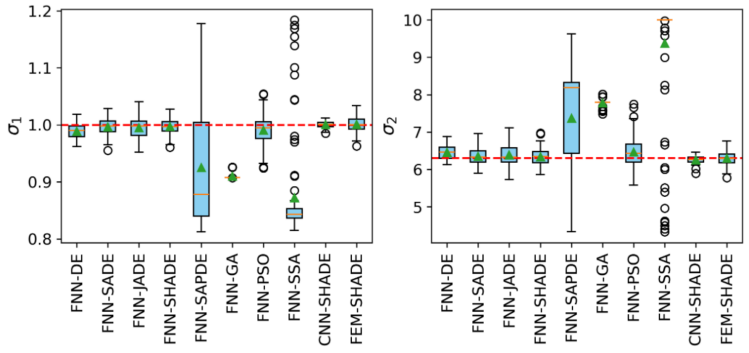

Figure 14. Reconstruction results of configuration 1 using different mesh sizes with matched forward and inverse meshes to evaluate discretization errors. The red dashed line represents the true solution, the green triangles indicate the mean, and the orange line marks the median.

This stark difference in computational cost highlights the strength of using FNN models for the EIT problem. The FNN-based approach drastically reduces runtime while maintaining an accuracy

comparable to FEM, especially beneficial when solving large-scale or computationally demanding problems. The reduced time is crucial for practical applications where repeated runs and real-time performance are necessary. Meanwhile, the slight variation in runtime among the neural network-based GOAs reflects their inherent stochastic nature and convergence behavior, but all remain significantly faster than FEM-SHADE. This efficiency gain, combined with acceptable accuracy, makes FNN-SHADE a practical and powerful tool for EIT.

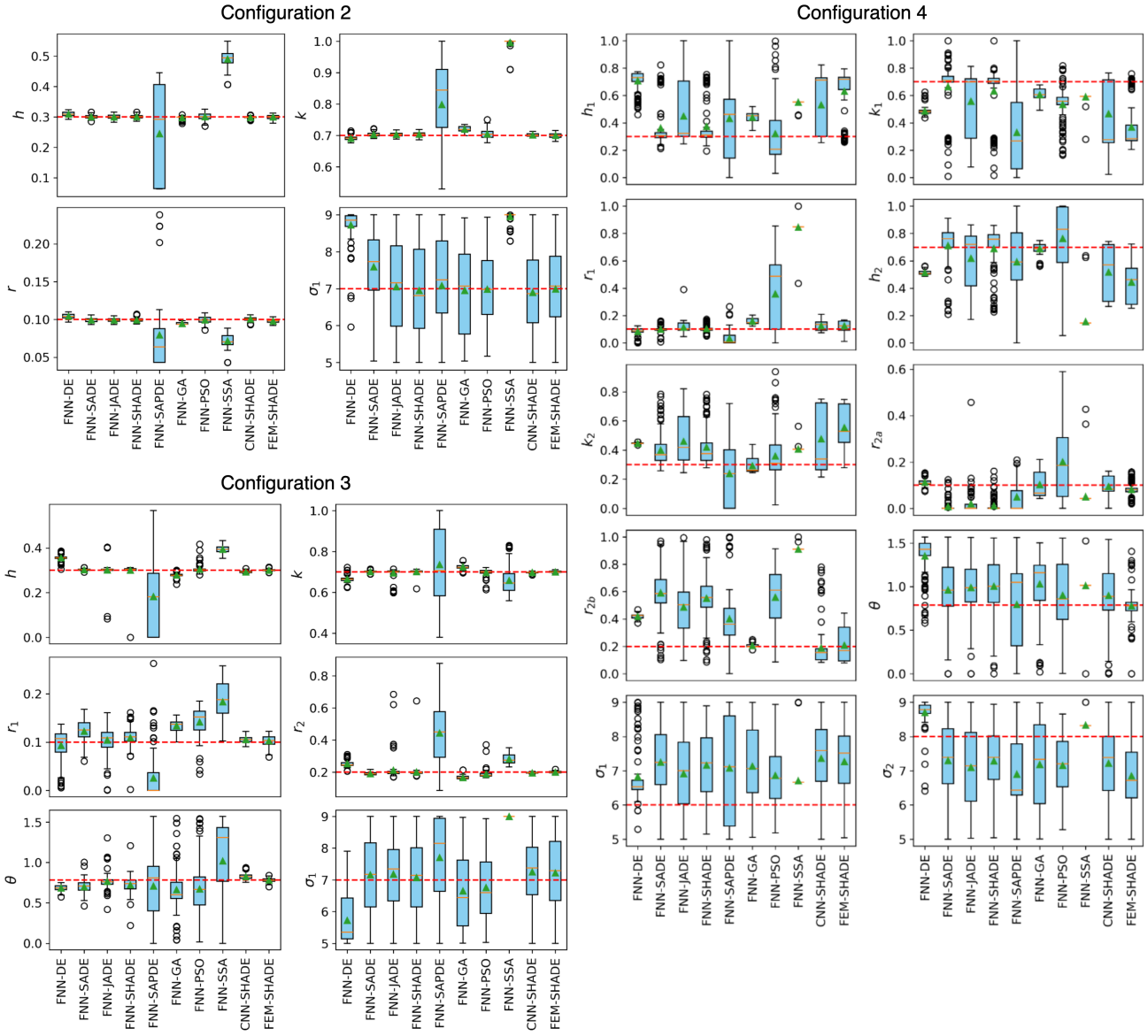

Figure 15. Reconstruction results of configurations 2, 3, and 4 using different mesh sizes with matched forward and inverse meshes to evaluate discretization errors. The red dashed line represents the true solution, the green triangles indicate the mean, and the orange line marks the median.

In addition to the MSE and IoU, we generated box plot visualizations to illustrate in Figures 14 and 15 the spread of the recovered parameters associated to the reconstructed inclusions. This allows us to assess accuracy (proximity of means to true values) and stability (width of confidence intervals and presence of outliers).

In configuration 1, the box plots reveal that almost all algorithms produce parameter estimates very close to the true solution, with narrow confidence intervals across both conductivity parameters. This indicates that the reconstructions are highly stable and consistent regardless of the optimizer used. Only a few isolated outliers appear across the algorithms, but their effect on the median and mean values is minimal.

In configuration 2, the box plots show that the geometric parameters h, k, and r have very tight confidence intervals around the true values for almost all algorithms, indicating stable reconstructions across runs and minimal sensitivity to stochastic effects. On the other hand, the spread for σ_1 is noticeably larger. This is likely due to the boundary electric potential being less sensitive to conductivity than to geometric parameters, causing the global optimization algorithms to shift their focus toward optimizing the latter. That said, FNN-SHADE, CNN-SHADE, and FEM-SHADE all maintain mean values that remain very close to the true solution. This pattern demonstrates that, for this configuration, all methods converge reliably on the geometric parameters and that the best-performing optimizers also achieve accurate conductivity estimate.

In configuration 3, the mean estimates of h, k, r_1, and r_2 obtained by FNN–SHADE, CNN–SHADE, and FEM–SHADE were very close to the desired values. The confidence intervals around these means were narrow with only a few outliers, indicating consistent performance across runs even under varying noise. For the rotation angle θ, FNN–SHADE produced a mean still close to the true value but exhibited a few farther outliers compared with CNN–SHADE and FEM–SHADE. For the conductivity parameter σ_1, all three methods performed similarly, with comparable means and spreads.

In configuration 4, which involves two inclusions but with different geometry and placement, h_1, k_1, and r_1 again had means close to the true values with narrow confidence intervals, confirming stable recovery of the primary inclusion. However, for h_2, k_2, r_{2a}, and r_{2b}, the means were somewhat farther from the desired values and the distributions showed more outliers, reflecting the increased difficulty of estimating the second inclusion under noise. The rotation angle θ had a slightly biased mean, with FEM–SHADE achieving the closest alignment to the true value, suggesting its advantage in angular estimation. For the conductivities σ_1 and σ_2, both inclusions yielded mean values near seven. This convergence, as well as the inaccuracies in the geometry and placement, likely results from occasional switching of the location of the two inclusions and associated conductivity labels across runs.

It is important to note that for some parameters and configurations, solutions tend to be toward the endpoints of the bounds. We attribute this to `mealpy`'s default handling of boundary conditions, which clips any candidate outside the search limits to the nearest bound.

The reconstruction metrics (MSE and IoU) and box plot visualizations show how reconstruction uncertainty grows with problem complexity, yet the results remain generally stable, highlighting where the proposed methods are most reliable and where additional modeling or regularization may be beneficial.

Overall, FNN-SHADE demonstrated superior accuracy and robustness in both conductivity recovery and geometric reconstruction, making it the most reliable algorithm for solving the EIT inverse problem in this study. While FNN-SADE and FNN-JADE also performed well, FNN-SHADE

consistently delivered the more accurate and stable reconstructions.

In terms of computation time, FNN-SHADE achieved substantial speedups compared to CNN-SHADE and FEM-SHADE across all four configurations. In Configuration 1, it was about 12 times faster than CNN-SHADE and more than 168 times faster than FEM-SHADE. In the second configuration, the gains remained significant, with FNN-SHADE being over 4 times faster than CNN-SHADE and nearly 59 times faster than FEM-SHADE. For Configuration 3, it ran about 3 times faster than CNN-SHADE and over 61 times faster than FEM-SHADE. Finally, in the fourth configuration, FNN-SHADE was more than 3 times faster than CNN-SHADE and about 43 times faster than FEM-SHADE. These results highlight the considerable computational advantage of FNN-SHADE in solving the inverse EIT problem.

Given its consistent performance and substantial computational advantage across all configurations, we focus on FNN-SHADE in the subsequent experiments.

4.3. Varying mesh size

To assess the influence of discretization error, model mismatch, and electrode uncertainties, we trained and tested our inverse models on four distinct mesh resolutions (26×26, 51×51, 101×101, 201×201) using the corresponding meshes for the inverse problem.

In Figure 16, the boxplots for h, k, and r across the four mesh resolutions reveal a clear and consistent pattern. There is a slightly larger spread at the coarsest mesh, tighter interquartile ranges at intermediate and fine meshes, and stable central tendencies very close to the true values at all mesh sizes. This convergence indicates that geometric parameters become more accurate and stable as discretization error diminishes. In contrast, σ_1 retains a larger absolute variance but maintains its mean near the true value across all meshes. In terms of computational costs, the forward model training became progressively slower with increasing mesh resolution, whereas inverse reconstructions across all tested mesh sizes required a similar average runtime of about 16 seconds per run.

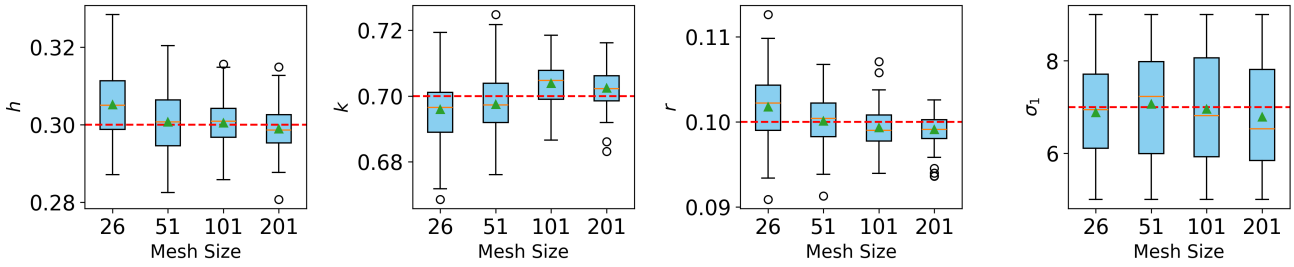

Figure 16. Reconstruction results using different mesh sizes with matched forward and inverse meshes to evaluate discretization errors. The red dashed line represents the true solution, the green triangles indicate the mean, and the orange line marks the median.

To examine potential inverse crime effects, we also tested our inverse solvers on voltage data generated on a finer grid. Specifically, the true simulated voltage data came from a 101×101 grid, while the inverse solvers operated on coarser grids (either 51×51 or 26×26). This setup allowed us

to observe how differences between the data generation mesh and the inversion mesh influenced reconstruction stability and accuracy. The results in Figure 17 showed that, despite the mismatch between meshes, the reconstructed geometric parameters remained stable and close to the true values, indicating that the approach is reasonably robust to mesh discrepancies.

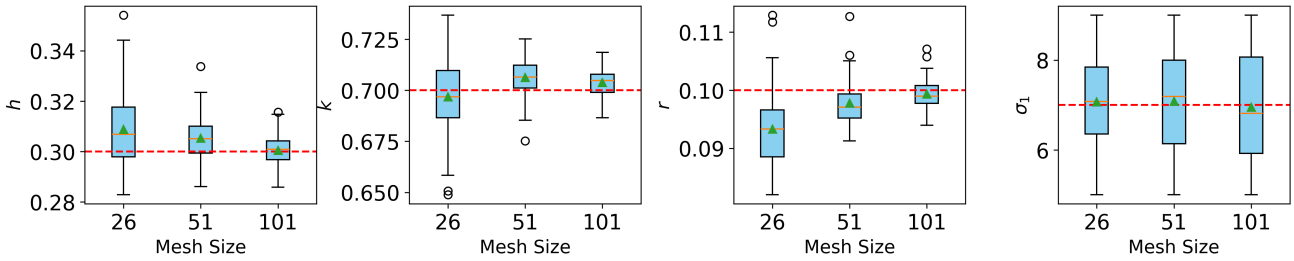

Figure 17. Comparison of reconstruction results when using coarser inverse meshes on data simulated on a finer mesh to evaluate inverse crime effects. The red dashed line represents the true solution, the green triangles indicate the mean, and the orange line marks the median.

4.4. Uncertainty quantification in the applied current

In Figure 18, the box plots comparing the original and perturbed current injection patterns serve as an additional validation step and provide an uncertainty analysis by showing the effect of current perturbations on the estimated parameters. For h, k, and r, the central tendencies remain essentially unchanged between the two current configurations, indicating that the model generalizes well to unseen boundary excitations. Variability increases slightly under the perturbed pattern—especially for r—but the spread remains small and the estimates continue to cluster near the true values. Similarly, σ_1 shows the largest absolute variance but retains its mean and median across both current patterns. These findings show that introducing perturbed current injection patterns produced no meaningful degradation in performance compared to the unperturbed case, suggesting that the reconstructions are robust to electrode-related uncertainties such as placement shifts, contact impedance, and imperfect current delivery, and also partially simulate model mismatch between training and testing forward operators.

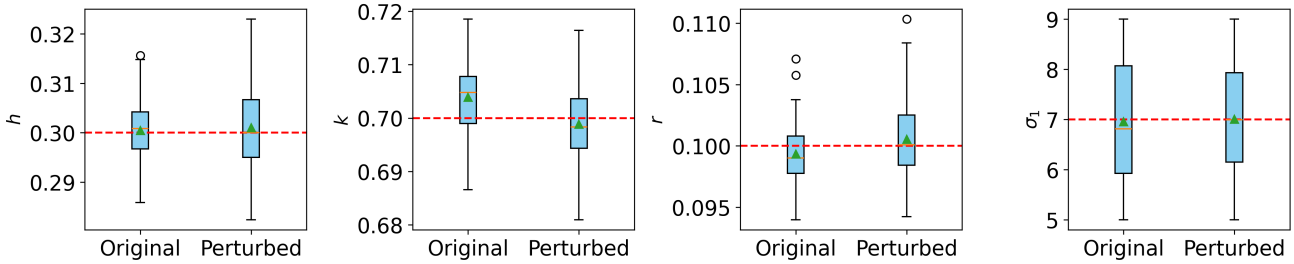

Figure 18. Comparison of results using the original current pattern and the perturbed current pattern. The red dashed line represents the true solution, the green triangles indicate the mean, and the orange line marks the median.

Overall, the combined results from mesh variation, perturbed excitations, and added Gaussian noise show that our inverse model is robust to discretization error, electrode-related variability, and differences in boundary conditions. The method consistently recovers geometric parameters across diverse conditions and maintains reasonable stability for conductivity, indicating strong generalization beyond the specific forward configurations used in training.

5. Conclusions

This paper presents a hybrid framework for solving the forward and inverse problems in EIT, combining deep neural networks and DE. The neural networks were trained to approximate solutions for the forward problem across diverse geometric configurations, while DE, its variants, and other optimization algorithms were used for reconstructing the internal conductivity distribution in the inverse problem. This integrated framework addresses the ill-posedness of the EIT inverse problem, with the global search capability of DE overcoming the limitations of gradient-based methods that tend to stagnate in flat regions of the neural network-based objective function. Among the optimization techniques explored, SHADE outperformed the others.

FEM served as a benchmark for SHADE's performance in solving the inverse problem. A comparison of the FNN, CNN, and FEM approaches revealed key insights into their respective strengths and limitations, highlighting the trade-offs between efficiency and accuracy. Although FEM and CNN excelled in terms of accuracy, their slower implementation made it less suitable for real-time inverse problem solving. The FNN-SHADE framework has the potential for accurate and efficient reconstructions, emphasizing the promise of integrating machine learning with evolutionary optimization in EIT applications.

There are several directions for future research and improvement in this area of study. One potential avenue is the refinement of hyperparameter tuning specifically tailored to each configuration. By conducting more targeted and configuration-specific hyperparameter optimization, it may be possible to further improve the performance of optimization algorithms and the accuracy of neural network predictions. Future work may also explore alternative activation functions beyond ReLU to better capture sharp conductivity transitions and potentially reduce estimation errors near discontinuities. Furthermore, hyperparameter tuning for the optimization algorithms could further improve reconstruction accuracy and reduce computational costs. In addition, evaluating a broader range of optimization algorithms, including other SHADE variants such as L-SHADE, and experimenting with different Python optimization packages as well as boundary constraint handling strategies may provide further improvement in performance and robustness.

Another important area for future work involves extending the study to more complex scenarios, such as those involving more than two inclusions. These configurations introduce even more complexities of the conductivity distribution. A more diverse dataset along with improved optimization techniques will be required to effectively handle these configurations.

Another key direction is to validate the proposed framework on experimental and clinical EIT datasets. While the present study relied on simulated data to maintain full control over ground-truth conductivity distributions and ensure consistency across experiments, future work will extend the framework to real-world data to demonstrate robustness and practical applicability. In addition, establishing a standardized benchmark across different EIT reconstruction methods would enable fair

comparisons between fundamentally different frameworks. Designing such a benchmark is beyond the scope of the present study but represents a natural and important next step.

Lastly, it would be worth considering using physics-informed neural networks (PINNs) or their variations, as opposed to traditional artificial neural networks. These incorporate physical laws directly into the neural network structure and have shown great promise in solving inverse problems, particularly when dealing with differential equations and boundary conditions. Employing PINNs or similar approaches could potentially lead to more accurate and robust solutions, as they are designed to leverage already-known physics alongside data-driven learning.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors acknowledge the Office of the Chancellor of the University of the Philippines Diliman, through the Office of the Vice Chancellor for Research and Development, for funding support through the Open Grant (242401 OG).

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. T. K. Bera, Applications of electrical impedance tomography (EIT): A short review, in *3rd International Conference on Communication Systems (ICCS-2017)*, (2017), 14–16. https://doi.org/10.1088/1757-899X/331/1/012004
2. S. Mansouri, S. Chabchoub, Y. Alharbi, A. Alshrouf, EIT 40-electrodes breast cancer detection and screening, *IEEJ Trans. Electr. Electron. Eng.*, **17** (2022), 1141–1147. https://doi.org/10.1002/tee.23605
3. W. Kuk, N. Wright, Bedside diagnosis of pulmonary embolism using electrical impedance tomography: A case report, *A & A Pract.*, **16** (2022), e01606. https://doi.org/10.1213/XAA.0000000000001606
4. M. Kamiński, P. Zientara, M. Krawczyk, Electrical resistivity tomography and digital aerial photogrammetry in the research of the "Bachledzki Hill" active landslide – in Podhale (Poland), *Eng. Geol.*, **285** (2021), 106004. https://doi.org/10.1016/j.enggeo.2021.106004
5. A. A. Konaté, O. B. Kaba, M. S. M. Conté, M. Zaheer, B. M. Thiam, F. Oularé, et al., Use of electrical resistivity tomography (ERT) for detecting underground voids on electrical pylon installation sites: Case studies from Labé Prefecture, Republic of Guinea, in *Proceedings of the 9th International Conference on Civil Engineering (ICCE)*, Springer, (2023), 611–620. https://doi.org/10.1007/978-981-99-2532-2_51

6. A. J. Thomas, J. J. Kim, T. N. Tallman, C. E. Bakis, Damage detection in self-sensing composite tubes via electrical impedance tomography, *Composites, Part B*, **177** (2019), 107276. https://doi.org/10.1016/j.compositesb.2019.107276
7. G. Rao, S. Aghajanian, Y. Zhang, L. Jackowska-Strumiłło, T. Koiranen, M. Fjeld, Monitoring and visualization of crystallization processes using electrical resistance tomography: CaCO3 and sucrose crystallization case studies, *Sensors*, **22** (2022), 4431. https://doi.org/10.3390/s22124431
8. M. Cheney, D. Isaacson, J. C. Newell, Electrical impedance tomography, *SIAM Rev.*, **41** (1999), 85–101. https://doi.org/10.1137/S0036144598333613
9. L. Borcea, Electrical impedance tomography, *Inverse Probl.*, **18** (2002), R99. http://doi.org/10.1088/0266-5611/18/6/201
10. S. J. Hamilton, S. Siltanen, Nonlinear inversion from partial EIT data: Computational experiments, *Contemp. Math.*, **615** (2014), 105–129.
11. G. Alessandrini, Stable determination of conductivity by boundary measurements, *Appl. Anal.*, **27** (1988), 153–172. https://doi.org/10.1080/00036818808839730
12. E. J. Woo, P. Hua, J. G. Webster, W. J. Tompkins, Finite-element method in electrical impedance tomography, *Med. Biol. Eng. Comput.*, **32** (1994), 530–536. https://doi.org/10.1007/BF02515311
13. H. Jo, K. Josić, J. K. Kim, Neural network-based parameter estimation for non-autonomous differential equations with discontinuous signals, preprint, arXiv:2507.06267.
14. T. Huuhtanen, A. Jung, Anomaly location detection with electrical impedance tomography using multilayer perceptrons, in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, (2020), 1–6. https://doi.org/10.1109/MLSP49062.2020.9231818
15. X. Li, R. Lu, Q. Wang, J. Wang, X. Duan, Y. Sun, et al., One-dimensional convolutional neural network (1D-CNN) image reconstruction for electrical impedance tomography, *Rev. Sci. Instrum.*, **91** (2020), 124704. https://doi.org/10.1063/5.0025881
16. R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.*, **11** (1997), 341–359. https://doi.org/10.1023/A:1008202821328
17. A. A. Kawam, N. Mansour, Metaheuristic optimization algorithms for training artificial neural networks, *Int. J. Comput. Inf. Technol.*, **1** (2012), 156–161.
18. L. M. R. Rere, M. I. Fanany, A. M. Arymurthy, Metaheuristic algorithms for convolution neural network, *Comput. Intell. Neurosci.*, **2016** (2016), 1537325. https://doi.org/10.1155/2016/1537325
19. D. Karaboğa, S. Ökdem, A simple and global optimization algorithm for engineering problems: Differential evolution algorithm, *Turk. J. Electr. Eng. Comput. Sci.*, **12** (2004), 53–60.
20. A. P. Piotrowski, J. J. Napiorkowski, A. E. Piotrowska, Particle swarm optimization or differential evolution – a comparison, *Eng. Appl. Artif. Intell.*, **121** (2023), 106008. https://doi.org/10.1016/j.engappai.2023.106008

21. R. R. Ribeiro, A. R. S. Feitosa, R. E. de Souza, W. P. dos Santos, A modified differential evolution algorithm for the reconstruction of electrical impedance tomography images, in *5th ISSNIP-IEEE Biosignals and Biorobotics Conference (2014): Biosignals and Robotics for Better and Safer Living (BRC)*, IEEE, (2014), 1–6. https://doi.org/10.1109/BRC.2014.6880982
22. R. H. Tan, C. Rossa, Electrical impedance tomography using differential evolution integrated with a modified Newton Raphson algorithm, in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, (2020), 2528–2534. https://doi.org/10.1109/SMC42975.2020.9282957
23. P. Wang, L. L. Xie, Y. C. Sun, Application of PSO algorithm and RBF neural network in electrical impedance tomography, in *2009 9th International Conference on Electronic Measurement & Instruments (ICEMI)*, IEEE, (2009), 2-517–2-521. https://doi.org/10.1109/ICEMI.2009.5274525
24. P. Wang, L. L. Xie, Y. C. Sun, Electrical impedance tomography based on BP neural network and improved PSO, in *2009 International Conference on Machine Learning and Cybernetics (ICMLC)*, IEEE, (2009), 1059–1064. https://doi.org/10.1109/ICMLC.2009.5212387
25. S. Martin, C. T. M. Choi, Nonlinear electrical impedance tomography reconstruction using artificial neural networks and particle swarm optimization, *IEEE Trans. Magn.*, **52** (2015), 1–4. https://doi.org/10.1109/TMAG.2015.2488901
26. H. Wang, K. Liu, Y. Wu, S. Wang, Z. Zhang, F. Li, et al., Image reconstruction for electrical impedance tomography using radial basis function neural network based on hybrid particle swarm optimization algorithm, *IEEE Sens. J.*, **21** (2020), 1926–1934. https://doi.org/10.1109/JSEN.2020.3019309
27. A. P. Bagshaw, A. D. Liston, R. H. Bayford, A. Tizzard, A. P. Gibson, A. T. Tidswell, et al., Electrical impedance tomography of human brain function using reconstruction algorithms based on the finite element method, *NeuroImage*, **20** (2003), 752–764. https://doi.org/10.1016/S1053-8119(03)00301-X
28. T. Murai, Y. Kagawa, Electrical impedance computed tomography based on a finite element model, *IEEE Trans. Biomed. Eng.*, **BME-32** (1985), 177–184. https://doi.org/10.1109/TBME.1985.325526
29. A. P. Calderón, On an inverse boundary value problem, *Comput. Appl. Math.*, **25** (2006), 133–138.
30. R. Mendoza, S. Keeling, A two-phase segmentation approach to the impedance tomography problem, *Inverse Probl.*, **33** (2017), 015001. https://doi.org/10.1088/0266-5611/33/1/015001
31. A. C. Velasco, M. Darbas, R. Mendoza, M. Bacon, J. C. de Leon, Comparative study of heuristic algorithms for electrical impedance tomography, *Philipp J. Sci.*, **149** (2020), 747–772.
32. K. Astala, L. Päivärinta, Calderón's inverse conductivity problem in the plane, *Ann. Math.*, **163** (2006), 265–299. https://doi.org/10.4007/annals.2006.163.265
33. Dask Development Team, Dask: Library for dynamic task scheduling, 2016. Available from: http://dask.pydata.org.
34. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al., Scikit-learn: Machine learning in Python, *J. Mach. Learn. Res.*, **12** (2011), 2825–2830.

35. D. P. Leins, C. Gibas, R. Brück, R. Haschke, Toward more robust hand gesture recognition on EIT data, *Front. Neurorobot.*, **15** (2021), 659311. https://doi.org/10.3389/fnbot.2021.659311

36. O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Springer, (2015), 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

37. A. Wirgin, The inverse crime, preprint, arXiv:math-ph/0401050.

38. A. K. Qin, P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in *2005 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, (2005), 1785–1791. https://doi.org/10.1109/CEC.2005.1554904

39. J. Zhang, A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, *IEEE Trans. Evol. Comput.*, **13** (2009), 945–958. https://doi.org/10.1109/TEVC.2009.2014613

40. R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in *2013 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, (2013), 71–78. https://doi.org/10.1109/CEC.2013.6557555

41. J. Teo, Exploring dynamic self-adaptive populations in differential evolution, *Soft Comput.*, **10** (2006), 673–686. https://doi.org/10.1007/s00500-005-0537-1

42. J. K. Pattanaik, M. Basu, D. P. Dash, Dynamic economic dispatch: A comparative study for differential evolution, particle swarm optimization, evolutionary programming, genetic algorithm, and simulated annealing, *J. Electr. Syst. Inf. Technol.*, **6** (2019), 1. https://doi.org/10.1186/s43067-019-0001-4

43. C. Dimas, V. Alimisis, N. Uzunoglu, P. P. Sotiriadis, Advances in electrical impedance tomography inverse problem solution methods: From traditional regularization to deep learning, *IEEE Access*, **12** (2024), 47797–47829. https://doi.org/10.1109/ACCESS.2024.3382939

44. M. Cheney, D. Isaacson, J. C. Newell, S. Simske, J. Goble, NOSER: An algorithm for solving the inverse conductivity problem, *Int. J. Imaging Syst. Technol.*, **2** (1990), 66–75. https://doi.org/10.1002/ima.1850020203

45. B. Sun, S. Yue, Z. Hao, Z. Cui, H. Wang, An improved Tikhonov regularization method for lung cancer monitoring using electrical impedance tomography, *IEEE Sens. J.*, **19** (2019), 3049–3057. https://doi.org/10.1109/JSEN.2019.2892179

46. S. Li, H. Wang, T. Liu, Z. Cui, J. N. Chen, Z. Xia, et al., A fast Tikhonov regularization method based on homotopic mapping for electrical resistance tomography, *Rev. Sci. Instrum.*, **93** (2022), 043709. https://doi.org/10.1063/5.0077483

47. Y. Shi, J. Liao, M. Wang, Y. Li, F. Fu, M. Soleimani, Total fractional-order variation regularization based image reconstruction method for capacitively coupled electrical resistance tomography, *Flow Meas. Instrum.*, **82** (2021), 102081. https://doi.org/10.1016/j.flowmeasinst.2021.102081

48. W. Zhang, C. Tan, Y. Xu, F. Dong, Electrical resistance tomography image reconstruction based on modified OMP algorithm, *IEEE Sens. J.*, **19** (2019), 5723–5731. https://doi.org/10.1109/JSEN.2019.2906264

49. A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM J. Imaging Sci.*, **2** (2009), 183–202. https://doi.org/10.1137/080716542
50. D. Liu, V. Kolehmainen, S. Siltanen, A. Laukkanen, A. Ursin, Estimation of conductivity changes in a region of interest with electrical impedance tomography, *Inverse Probl. Imaging*, **9** (2015), 211–229. https://doi.org/ 10.3934/ipi.2015.9.211
51. B. Liu, B. Yang, C. Xu, J. Xia, M. Dai, Z. Ji, et al., pyEIT: A python based framework for electrical impedance tomography, *SoftwareX*, **7** (2018), 304–308. https://doi.org/10.1016/j.softx.2018.09.005
52. K. Knudsen, M. Lassas, J. L. Mueller, S. Siltanen, Regularized D-bar method for the inverse conductivity problem, *Inverse Probl. Imaging*, **35** (2009), 599–624. https://doi.org/10.3934/ipi.2009.3.599
53. J. L. Mueller, S. Siltanen, The D-bar method for electrical impedance tomography—demystified, *Inverse Probl.*, **36** (2020), 093001. https://doi.org/10.1088/1361-6420/aba2f5
54. E. T. Chung, T. F. Chan, X. C. Tai, Electrical impedance tomography using level set representation and total variational regularization, *J. Comput. Phys.*, **205** (2005), 357–372. https://doi.org/10.1016/j.jcp.2004.11.022
55. M. Soleimani, W. R. B. Lionheart, O. Dorn, Level set reconstruction of conductivity and permittivity from boundary electrical measurements using experimental data, *Inverse Probl. Sci. Eng.*, **14** (2006), 193–210. https://doi.org/10.1080/17415970500264152
56. L. Zhou, B. Harrach, J. K. Seo, Monotonicity-based electrical impedance tomography for lung imaging, *Inverse Probl.*, **34** (2018), 045005. https://doi.org/10.1088/1361-6420/aaaf84
57. A. C. Carcavilla, M. Meribout, Algorithms on electrical impedance tomography, focusing on deep learning architectures and their implementations: A review, *IEEE Sens. J.*, **25** (2025), 34252–34274. https://doi.org/10.1109/JSEN.2025.3589157
58. N. Van Thieu, S. Mirjalili, MEALPY: An open-source library for latest meta-heuristic algorithms in Python, *J. Syst. Archit.*, **139** (2023), 102871. https://doi.org/10.1016/j.sysarc.2023.102871

AIMS Press