Supplementary Information: A unified physics-informed generative operator framework for general inverse problems

Gang Bao[1] and Yaohua Zang[2*]

[1]Center for Interdisciplinary Applied Mathematics, Zhejiang University, Yuhangtang Road 886, Hang Zhou, 310058, China.
[2*]School of Engineering and Design, Technical University of Munich, Boltzmannstraße 15, Garching, 85748, Germany.

*Corresponding author(s). E-mail(s): yaohua.zang@tum.de;
Contributing authors: baog@zju.edu.cn;

Contents

1 Supplementary Note 1: The MultiONet Architecture **3**

2 Supplementary Note 2: Normalizing Flow for Initialization **3**

3 Supplementary Note 3: Training and Inversion Algorithms **4**

4 Supplementary Note 4: Baseline Method Details **5**

5 Supplementary Methods **6**
5.1 General Implementation Details 6
5.2 Continuous Coefficient Recovery with Solution-based Measurements 6
5.2.1 Problem Setup 6
5.2.2 Model Setup 7
5.2.3 Latent-space Optimization Details 8
5.3 Piecewise-Constant Coefficient Recovery with Solution-based Measurements 8
5.3.1 Problem Setup 8
5.3.2 Model Setup 9
5.3.3 Latent-space Optimization Details 10
5.4 The EIT Problem with Operator-based Measurements 10
5.4.1 Problem Setup 10

5.4.2 Model Setup 11
5.4.3 Latent-space Optimization Details 11

6 Supplementary Tables **12**
6.1 Continuous Coefficient Recovery with Solution-based Measurements 12
6.1.1 Table 1: In-distribution case 12
6.1.2 Table 2: Out-of-distribution case 12
6.2 Piecewise-Constant Coefficient Recovery with Solution-based Measuremen 12
6.2.1 Table 3: In-distribution case 12
6.2.2 Table 4: Out-of-distribution case 13
6.3 The EIT Problem with Operator-based Measurements 13
6.4 Table 5: the EIT problem 13

7 Supplementary Figures **14**
7.1 Figure 1: The MultiONet architecture 14
7.2 Figure 2: Illustration of operator-based measurements in the EIT problem 15

1 Supplementary Note 1: The MultiONet Architecture

The MultiONet architecture extends the standard DeepONet by aggregating information from multiple intermediate layers of both branch and trunk networks, enriching the learned representation without increasing trainable parameters. The architecture, illustrated in Fig. 1, comprises two subnetworks:

- **Trunk network**: Encodes the spatial coordinates $\boldsymbol{x} \in \Omega$ of the output field.
- **Branch network**: Encodes a latent vector $\boldsymbol{\beta}$ representing a compact descriptor of the input functions.

Unlike the original DeepONet, MultiONet computes a weighted average of inner products between layer-wise features from multiple network depths rather than relying solely on last-layer features. Formally, the MultiONet mapping $\mathcal{G}(\boldsymbol{\beta})(\boldsymbol{x})$ is expressed as:

$$\mathcal{G}(\boldsymbol{\beta})(\boldsymbol{x}) = \frac{1}{l}\sum_{k=1}^{l} w^{(k)} \left(b^{(k)}(\boldsymbol{\beta}) \odot t^{(k)}(\boldsymbol{x})\right) + b_0, \tag{1}$$

where $b^{(k)}(\boldsymbol{\beta})$ and $t^{(k)}(\boldsymbol{x})$ denote the outputs from the k-th layers of the branch and trunk networks, l represents the total number of layers, $w^{(k)}$ indicates trainable weights, b_0 is the bias term, and $\odot$ represents the inner product operation.

2 Supplementary Note 2: Normalizing Flow for Initialization

In gradient-based inversion, initial guess quality for the latent variable affects both convergence speed and optimization quality. Random initialization (e.g., from uniform or Gaussian distributions) ignores the trained latent space structure and may lead to inefficient optimization, particularly for highly nonlinear PDE inverse problems. We employ a normalizing flow (NF) model to learn a bijective mapping between the latent space and a standard multivariate normal distribution, enabling informed and probabilistically meaningful initialization. Specifically, we use the RealNVP architecture [1], which consists of invertible coupling layers. Each coupling layer splits the input $\boldsymbol{x} \in \mathbb{R}^d$ into $\boldsymbol{z}_1 \in \mathbb{R}^{d/2}$ and $\boldsymbol{z}_z \in \mathbb{R}^{d/2}$, then applies:

- Forward: $\boldsymbol{y}_1 = \boldsymbol{z}_1$, $\boldsymbol{y}_2 = \boldsymbol{z}_2 \odot \exp(s(\boldsymbol{z}_1)) + t(\boldsymbol{z}_1)$
- Inverse: $\boldsymbol{z}_1 = \boldsymbol{y}_1$, $\boldsymbol{z}_2 = (\boldsymbol{y}_2 - t(\boldsymbol{y}_1)) \odot \exp(-s(\boldsymbol{y}_1))$

where $s(\cdot)$ and $t(\cdot)$ are scale and translation networks parameterized by fully connected neural networks (FCNNs).

3 Supplementary Note 3: Training and Inversion Algorithms

Algorithm 1 Training the IGNO Framework

Inputs:

Training data $\mathcal{D} = \{a^{(n)}, g^{(n)}\}_{n=1}^{N}$; loss weights λ_{pde}, λ_{bd}, λ_{rec}.

Initialize:

Model parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_u, \boldsymbol{\theta}_a, \boldsymbol{\theta}_{\boldsymbol{\beta}_1}, \boldsymbol{\theta}_{\boldsymbol{\beta}_2})$; learning rate lr.

while Convergence or maximum number of iterations not reached **do**

Obtain latent representations:

$$\boldsymbol{\beta}_1^{(n)} \leftarrow E_{\boldsymbol{\beta}_1}(\boldsymbol{a}^{(n)}), \quad \boldsymbol{\beta}_2^{(n)} \leftarrow E_{\boldsymbol{\beta}_2}(\boldsymbol{g}^{(n)}). \quad n = 1, \cdots, N$$

Obtain the predicted PDE solutions and recovered coefficients:

$$u_{\text{pred}}^{(n)} \leftarrow \mathcal{G}_{\boldsymbol{\theta}_u}(\boldsymbol{\beta}^{(n)}), \quad a_{\text{pred}}^{(n)} \leftarrow \mathcal{G}_{\boldsymbol{\theta}_a}(\boldsymbol{\beta}_1^{(n)}). \quad n = 1, \cdots, N$$

Compute the loss $\mathcal{L}(\boldsymbol{\theta})$ and update model parameters with SGD:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - lr \odot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}).$$

if at every 2500th epoch **then**

Update the learning rate to $lr \leftarrow lr/2$.

end if

end while

Outputs:

Trained encoders $E_{\boldsymbol{\theta}^*_{\beta_1}}$, $E_{\boldsymbol{\theta}^*_{\beta_2}}$, and decoders $\mathcal{G}_{\boldsymbol{\theta}^*_u}$, $\mathcal{G}_{\boldsymbol{\theta}^*_a}$.

Algorithm 2 Inversion via Latent Optimization

Inputs:
Trained networks $E_{\boldsymbol{\theta}^*_{\beta_1}}$, $E_{\boldsymbol{\theta}^*_{\beta_2}}$, $\mathcal{G}_{\boldsymbol{\theta}^*_u}$, $\mathcal{G}_{\boldsymbol{\theta}^*_a}$; Trained normalizing flow $F_{\boldsymbol{\theta}^*_{NF}}$; measurements $\mathcal{M}$
Initialize:
Sample $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$; Set $\boldsymbol{\beta}_1 \leftarrow F^{-1}_{\boldsymbol{\theta}^*_{NF}}(\boldsymbol{z})$; learning rate lr
while Convergence or maximum number of iterations not reached **do**
    Compute optimization objective $\mathcal{F}(\boldsymbol{\beta}_1)$.
    Update $\boldsymbol{\beta}_1$ with gradient descent:

$$\boldsymbol{\beta}_1 \leftarrow \boldsymbol{\beta}_1 - lr \cdot \nabla_{\boldsymbol{\beta}_1} \mathcal{F}(\boldsymbol{\beta}_1)$$

    if at every 250th epoch **then**
        Update the learning rate to $lr \leftarrow lr/2$.
    end if
end while
Outputs:
Recovered coefficient $a^{rec} = \mathcal{G}_{\boldsymbol{\theta}^*_a}(\boldsymbol{\beta}_1)$.

4 Supplementary Note 4: Baseline Method Details

We compare IGNO against physics-informed deep inverse operator networks (PI-DIONs) [2], a recent state-of-the-art method for PDE inverse problems with solution-based measurements. This method employs the DeepONet architecture to parameterize both the unknown coefficient a and the PDE solution u. Coordinates $\boldsymbol{x}$ are treated as inputs to the trunk networks, while solution-based measurements $\mathcal{M}_{sol}$ are input to the branch networks. The two resulting models are denoted $\mathcal{G}_{\boldsymbol{\theta}_u}$ and $\mathcal{G}_{\boldsymbol{\theta}_a}$, parameterized by $\boldsymbol{\theta}_u$ and $\boldsymbol{\theta}_a$, respectively. Predictions at location $\boldsymbol{x}$ are given by $\mathcal{G}_{\boldsymbol{\theta}_u}(\mathcal{M}_{sol})(\boldsymbol{x})$ for the solution and $\mathcal{G}_{\boldsymbol{\theta}_a}(\mathcal{M}_{sol})(\boldsymbol{x})$ for the coefficient.

Training of PI-DIONs relies on a physics-informed composite loss that combines (i) a physics loss, consisting of strong-form PDE residuals ($\mathcal{L}_{pde}$) and boundary mismatches ($\mathcal{L}_{bd}$), and (ii) a data loss ($\mathcal{L}_{data}$) that penalizes deviations between predicted and observed solutions at fixed sensors. For N training samples with solution-based measurements $\{\mathcal{M}^{(n)}_{sol}\}_{n=1}^N$ collected on sensors $\boldsymbol{X}_m = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \subset \Omega$, the total loss is defined as:

$$\begin{aligned}
\mathcal{L} &= \lambda_{physics}(\mathcal{L}_{pde} + \mathcal{L}_{bd}) + \lambda_{data}\mathcal{L}_{data} \\
&= \frac{\lambda_{physics}}{N} \sum_{n=1}^{N} \left(\|\boldsymbol{\mathcal{R}}^{(n)}(\mathcal{G}_{\boldsymbol{\theta}_u}(\mathcal{M}^{(n)}_{sol}), \mathcal{G}_{\boldsymbol{\theta}_a}(\mathcal{M}^{(n)}_{sol}))\|_2^2 + \|\boldsymbol{\mathcal{B}}^{(n)}(\mathcal{G}_{\boldsymbol{\theta}_u}(\mathcal{M}^{(n)}_{sol})(\Xi_{bd})) - \boldsymbol{g}^{(n)}\|_2^2 \right) \\
&\quad + \frac{\lambda_{data}}{N} \sum_{n=1}^{N} \|\mathcal{G}_{\boldsymbol{\theta}_u}(\mathcal{M}^{(n)}_{sol})(\boldsymbol{X}_m) - \mathcal{M}^{(n)}_{sol}\|_2^2,
\end{aligned} \tag{2}$$

where $\boldsymbol{\mathcal{R}}^{(n)} = (r^{(n)}_{w_1}, \ldots, r^{(n)}_{w_K}) \in \mathbb{R}^K$ denotes the strong-form residual vector. The terms $\lambda_{physics}$ and λ_{data} are loss weights. Once trained, the recovered coefficient a and solution u for test measurements

$\mathcal{M}_{sol}^{test}$ are directly obtained via $\mathcal{G}_{\boldsymbol{\theta}_a^*}(\mathcal{M}_{sol}^{test})$ and $\mathcal{G}_{\boldsymbol{\theta}_u^*}(\mathcal{M}_{sol}^{test})$, respectively. PI-DIONs is not applicable to operator-based inverse problems and serves as a baseline only for solution-based cases.

5 Supplementary Methods

5.1 General Implementation Details

Unless otherwise specified, all models are trained using the ADAM optimizer with an initial learning rate of 5×10^{-4}, which is halved every 400 epochs. Training is performed for 2,000 epochs to ensure convergence. The batch sizes are set to 50, 25, and 100 for the Darcy flow problem with continuous coefficients, the Darcy flow problem with piecewise-constant coefficients, and the EIT problem, respectively. For the proposed IGNO framework, the loss weights are assigned as $\lambda_{\text{pde}} = \lambda_{\text{bd}} = \lambda_{\text{rec}} = 1$ and the normalizing flow loss weight is set to 0.05. For the PI-DIONs method, loss weights are set as $\lambda_{physics} = 1$ and $\lambda_{data} = 100$ for best performance. All experiments are conducted under identical hardware conditions using a 64-core AMD Ryzen CPU paired with an NVIDIA RTX 4090 GPU.

5.2 Continuous Coefficient Recovery with Solution-based Measurements

5.2.1 Problem Setup

The governing PDE is the following Darcy's flow equation:

$$\begin{aligned} -\nabla(k(x,y)\nabla p(x,y)) &= f(x,y), \quad \text{in } \Omega = [0,1]^2, \\ p(x,y) &= 0, \quad \text{in } \partial\Omega, \end{aligned} \tag{3}$$

where k is the permeability field, p is the pressure field, and $f = 10$ is the source term. Measurements are collected at $m = 100$ sensor locations $\boldsymbol{X}_m = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$, with additive Gaussian noise contamination:

$$\mathcal{M}_{sol} = \left(\hat{p}(\boldsymbol{x}_1), \ldots, \hat{p}(\boldsymbol{x}_m)\right), \text{ where } \hat{p}(\boldsymbol{x}_i) = p(\boldsymbol{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2). \tag{4}$$

Noise levels are characterized by the signal-to-noise ratio (SNR), which is defined as:

$$\text{SNR} = 10 \log 10 \left(\frac{\frac{1}{m}\sum_{i=1}^{m} p^2(\boldsymbol{x}_i)}{\sigma^2} \right). \tag{5}$$

In particular, we examine three scenarios: low noise (SNR=50), medium noise (SNR=25), and high noise (SNR=15).

For training IGNO, $N = 1000$ synthetic permeability fields are generated as $k(x,y) = 2.1 + \sin(\omega_1 x) + \cos(\omega_2 y)$ with ω_1, ω_2 sampled independently from the uniform distribution $\text{U}(0, 7\pi/4)^2$. Since the boundary condition is fixed (i.e., $g = 0$) across all samples, only the permeability encoder $E_{\boldsymbol{\beta}_1}$ is required. For training PI-DIONs, pressure fields $p^{(n)}$ are computed by solving (3) via the finite element method (FEM) for each permeability sample $k^{(n)}$. Then, training measurements $\mathcal{M}_{sol}^{(n)}$ are formulated by sampling the pressure values at the same sensor locations $\boldsymbol{X}_m$ without added noise.

To evaluate the performance of both methods on this problem, we consider two classes of test targets:

- **In-distribution target:** permeability fields k generated with $(\omega_1, \omega_2) \sim \mathrm{U}(0, 7\pi/4)$ (with different random seed), consistent with the training distribution.
- **Out-of-distribution target:** permeability fields k generated with $(\omega_1, \omega_2) \sim \mathrm{U}(7\pi/4, 2\pi)$, which lie entirely outside the training distribution.

In both cases, noisy test measurements $\mathcal{M}_{sol}^{test}$ are generated by solving (3) with FEM and perturbing the pressure values with Gaussian noise at the specified SNR levels.

5.2.2 Model Setup

To extract latent representations from the input coefficients with the encoder $E_{\boldsymbol{\theta}_{\boldsymbol{\beta}_1}}$, we set $\Xi_a = (\xi_{a,1}, \ldots, \xi_{a,N_a}) \subset \Omega$ as a 29×29 uniform grid, so that the encoder input is $\boldsymbol{a} = a(\Xi_a)$. Since the boundary condition is fixed ($g = 0$) across all samples, no boundary input is needed (i.e., the boundary encoder vanishes). Moreover, a mollifier $f(x, y) = \sin(\pi x)\sin(\pi y)$ is applied to the output of the solution decoder, i.e.,

$$u_{pred}(x, y) = \mathcal{G}_{\boldsymbol{\theta}_u}(\boldsymbol{\beta})(x, y) \cdot f(x, y), \tag{6}$$

which enforces the boundary condition automatically and removes the need for an explicit boundary loss during training. Below, we provide details of the network architectures for both methods:

The IGNO method

- **Encoder $E_{\boldsymbol{\theta}_{\boldsymbol{\beta}_1}}$.** It consists of a Convolutional Neural Network (CNN) followed by a Feed-Forward Fully Connected Network (FFCN). The CNN has three hidden layers with 64 output channels each, kernel size $(3, 3)$, and stride 2. The FFCN has two hidden layers of 128 neurons each. The SiLU activation is applied to all hidden layers, while the output layer uses Tanh to constrain the latent variables to a bounded cubic region.
- **Solution Decoder $\mathcal{G}_{\boldsymbol{\theta}_u}$.** The solution decoder $\mathcal{G}_{\boldsymbol{\theta}_u}$ adopts the MultiONet architecture. Both branch and trunk networks are FFCNs with 6 hidden layers of 100 neurons each. A custom activation function, Tanh_Sin, is used in all hidden layers:

$$\mathrm{Tanh\_Sin}(x) = \tanh(\sin(\pi x + \pi)) + x. \tag{7}$$

- **Coefficient Decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$.** The coefficient decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$ adopts the same architecture as $\mathcal{G}_{\boldsymbol{\theta}_u}$.
- **NF model $F_{\boldsymbol{\theta}_{NF}}$.** The NF model consists of three flow steps, each parameterized by a fully connected network with two hidden layers of 64 neurons per layer and SiLU activations.

The PI-DIONs method

- **Decoder $\mathcal{G}_{\boldsymbol{\theta}_u}$.** For fair comparison, the PI-DIONs solution decoder adopts the same architecture as in IGNO. However, the input to the branch network is the solution-based measurement $\mathcal{M}_{sol}$, instead of the latent variable $\boldsymbol{\beta}$.
- **Decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$.** Similarly, the coefficient decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$ uses the same architecture as in IGNO, with the branch network taking $\mathcal{M}_{sol}$ as input instead of $\boldsymbol{\beta}_1$.

5.2.3 Latent-space Optimization Details

Gradients of the objective $\mathcal{F}$ with respect to $\boldsymbol{\beta}_1$ are computed via PyTorch's automatic differentiation. The weights ρ_{data} and ρ_{pde} are set to 50 and 1, respectively. Optimization is performed using ADAM with an initial learning rate of 0.01, decayed by a factor of 2/3 every 250 steps, for a total of 500 updates.

5.3 Piecewise-Constant Coefficient Recovery with Solution-based Measurements

5.3.1 Problem Setup

The governing equation remains Darcy's law (3) but with piecewise-constant coefficient:

$$k(x,y) = \begin{cases} 10, & (x,y) \in \Omega_1 \\ 5, & (x,y) \in \Omega_2 \end{cases} \tag{8}$$

where Ω_1 and Ω_2 represent two disjoint phases (i.e., phase 1 and phase 2) such that $\Omega_1 \cup \Omega_2 = \Omega$. Measurements $\mathcal{M}_{sol}$ are collected on $m = 100$ fixed sensors $\boldsymbol{X}_m$ randomly sampled in Ω. The measurements are contaminated with Gaussian noise, and three levels of noise are considered: low (SNR=50), medium (SNR=20), and high (SNR=15). The discontinuous nature of the coefficient field makes this problem particularly difficult for existing methods: (i) Gradient-based methods require differentiability of k, but in this case, k is discrete-valued, so gradients with respect to k are not well defined; (ii) Non-gradient methods such as evolutionary algorithms or MCMC may, in principle, handle discrete parameters, but they become computationally prohibitive when k lies in a high- or infinite-dimensional function space, restricting their applicability to only low-dimensional problems. The proposed IGNO framework naturally overcomes these challenges by introducing a low-dimensional and well-structured latent space representation of the original high-dimensional and discontinuous coefficient field. This transforms the otherwise intractable inverse problem into a tractable optimization problem in a continuous latent space, making it both efficient and robust.

To train IGNO, $10,000$ piecewise-constant permeability fields are generated by using a cutoff Gaussian Process $\mathcal{GP}(0, (-\Delta + 9I)^{-2})$ [3, 4]. For each GP realization, $k(x,y) = 10$ if the underlying GP-value is greater than 0 and $k(x,y) = 5$ otherwise. Again, only the permeability encoder $E_{\boldsymbol{\beta}_1}$ is required as the boundary condition is fixed across all samples. For training PI-DIONs, the pressure field $p^{(n)}$ corresponding to each permeability sample $k^{(n)}$ is obtained using FEM. Noise-free pressure values on the fixed sensors $\boldsymbol{X}_m$ are then used as training measurements $\mathcal{M}_{sol}^{(n)}$.

We evaluate both methods on two types of inverse targets:

- **In-distribution target:** The target permeability k is generated from the same distribution (with different random seed) used in training, i.e., the cutoff $\mathcal{GP}(0, (-\Delta + 9I)^{-2})$.
- **Out-of-distribution target:** The target k is generated from a different distribution, namely the cutoff $\mathcal{GP}(0, (-\Delta + 16I)^{-2})$. The higher-order correlation structure of k differs significantly from the training distribution, resulting in more complex geometries of the permeability field.

In both cases, test measurements $\mathcal{M}_{sol}^{test}$ are obtained by solving Darcy's equation with FEM, and then corrupted with Gaussian noise corresponding to the specified SNR levels.

5.3.2 Model Setup

The coefficient k is represented by an image of size 29×29, denoted by $\Xi_a = (\xi_{a,1}, \ldots, \xi_{a,N_a})$. Each pixel value corresponds to the permeability of the associated phase (either 10 or 5). To improve the performance of IGNO in this problem, the coefficient decoder is not trained to predict the coefficient values directly. Instead, it predicts the probability that a spatial location $\boldsymbol{x}$ belongs to phase 1. Accordingly, the recovery loss $\mathcal{L}_{rec}$ is replaced by a cross-entropy loss:

$$\mathcal{L}_{rec} = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{N_a} \left[z_i^{(n)} \log \sigma\big(\mathcal{G}_{\boldsymbol{\theta}_a}(\boldsymbol{\beta}_1^{(n)})(\xi_{a,i})\big) + (1 - z_i^{(n)}) \log\big(1 - \sigma(\mathcal{G}_{\boldsymbol{\theta}_a}(\boldsymbol{\beta}_1^{(n)})(\xi_{a,i}))\big) \right], \tag{9}$$

where $\sigma(\cdot)$ is the sigmoid function, and z_i is a binary label indicating the true phase at location $\xi_{a,i}$ ($z_i = 1$ for phase 1 and $z_i = 0$ for phase 2). At inference time, the recovered coefficient can be obtained either by sampling from the predicted probability field or, as done in this work, by applying a cut-off rule: values greater than 0.5 are assigned to phase 1 and others to phase 2. This strategy is not applicable to PI-DIONs, as the method relies on strong-form PDE residuals that require differentiability with respect to the coefficient field. Consequently, the coefficient decoder in PI-DIONs is designed to predict permeability values directly. To obtain binary reconstructions, a threshold of 7.5 is then applied to convert the continuous predictions into discrete phase labels. For IGNO, the 29×29 coefficient image is used as the coefficient encoder input. Since the boundary condition is fixed ($g = 0$) across all samples, the boundary encoder is omitted. To enforce boundary conditions automatically, a mollifier $f(x, y) = \sin(\pi x)\sin(\pi y)$ is applied to the solution decoder output:

$$u_{pred}(x, y) = \mathcal{G}_{\boldsymbol{\theta}_u}(\boldsymbol{\beta})(x, y) \cdot f(x, y). \tag{10}$$

Below, we present the network architectures for both methods:

The IGNO method

- **Encoder $E_{\boldsymbol{\theta}_{\boldsymbol{\beta}_1}}$.** The encoder $E_{\boldsymbol{\theta}_{\boldsymbol{\beta}_1}}$ is selected as an FFCN. The input coefficient image is first flattened into a vector, then passed through two dense layers with 512 and 256 neurons, respectively. SiLU activations are applied in all hidden layers, and a Tanh activation is applied at the output.
- **Solution Decoder $\mathcal{G}_{\boldsymbol{\theta}_u}$.** The solution decoder $\mathcal{G}_{\boldsymbol{\theta}_u}$ adopts the MultiONet architecture. Both the branch and trunk networks are FFCNs with five hidden layers of 100 neurons each. The custom activation function Tanh_Sin (Eq. (7)) is applied to all hidden layers.
- **Coefficient Decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$.** The coefficient decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$ outputs the probability that a given spatial point belongs to phase 1. The recovered permeability field is obtained by thresholding the probability field at 0.5 and mapping to 10 (phase 1) or 5 (phase 2). Both branch and trunk networks follow the MultiONet design, with five hidden layers of 256 neurons each. The trunk employs the custom activation

  $$\text{SiLU\_Sin}(x) = \text{SiLU}(\sin(\pi x + \pi)) + x, \tag{11}$$

  while the branch employs

  $$\text{SiLU\_Id}(x) = \text{SiLU}(x) + x. \tag{12}$$

  A Sigmoid activation is applied at the output to ensure predictions lie in $[0, 1]$.
- **NF model $F_{\boldsymbol{\theta}_{NF}}$.** The NF model consists of three flow steps. Each step is parameterized by an FFCN with two hidden layers of 128 neurons and SiLU activations.

The PI-DIONs method

- **Decoder $\mathcal{G}_{\boldsymbol{\theta}_u}$.** The solution decoder of PI-DIONs adopts the same architecture as in IGNO. However, its branch network takes as input the solution-based measurements $\mathcal{M}_{sol}$ rather than the latent variable $\boldsymbol{\beta}$.
- **Decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$.** Unlike IGNO, PI-DIONs predict the coefficient field a directly, because probability-based cut-offs are non-differentiable and incompatible with strong-form residual training. For fairness, the decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$ adopts the same architecture as in IGNO, except that the branch network input is $\mathcal{M}_{sol}$, and no activation is applied at the output layer.

5.3.3 Latent-space Optimization Details

Gradients are computed via PyTorch automatic differentiation. The weights ρ_{data} and ρ_{pde} are set to 1 and 1, respectively. The ADAM optimizer is used with an initial learning rate of 0.1, halved every 50 steps. Each inverse problem is solved with 500 gradient updates.

5.4 The EIT Problem with Operator-based Measurements

5.4.1 Problem Setup

The EIT is governed by the elliptic PDE with Dirichlet boundary conditions:

$$\begin{aligned} -\nabla(\gamma(x,y)\nabla u(x,y)) = 0, \quad \text{in } \Omega = [0,1]^2, \\ u(x,y) = g(x,y), \quad \text{in } \partial\Omega, \end{aligned} \tag{13}$$

where $\gamma > 0$ denotes the unknown conductivity field, assumed to be smooth, and g is the prescribed Dirichlet voltage. The goal is to recover γ from the DtN map Λ_γ, defined as:

$$\Lambda_\gamma[g] : g \longrightarrow \gamma \frac{\partial u}{\partial \vec{n}}|_{\partial\Omega}, \tag{14}$$

which maps the input voltage g into the current $\gamma\frac{\partial u}{\partial \vec{n}} = \gamma\nabla u \cdot \vec{n}$ at boundary with $\vec{n}$ being the unit outward normal vector. In practice, the DtN map is approximated using a finite set of pre-defined voltage–current pair, defined as $\mathcal{M}_{op} = \left\{\left(g_l(\boldsymbol{X}_m), \Lambda_\gamma[g_l](\boldsymbol{X}_m)\right)\right\}_{l=1}^{L}$, where $\boldsymbol{X}_m = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m)$ denotes sensors on the boundary. In our experiments, we use $m = 128$ equally spaced sensors along the four boundaries, as illustrated on the right of Fig. 2(a). To approximate the DtN map, we set $L = 20$, with the l-th input voltage defined as $\cos(2\pi(x\cos(\theta_l)+y\sin(\theta_l)))$ with $\theta_l = 2\pi l/20$. Fig. 2(b) shows examples of three input voltages ($l = 1, 10, 20$), their corresponding boundary currents, and the PDE solutions, with the conductivity field displayed on the left of Fig. 2(a).

Since the EIT problem involves operator-based measurements, the PI-DIONs method is not applicable. Therefore, we only evaluate the proposed IGNO in this problem. For training, $N = 1000$ conductivity fields are generated from trigonometric functions of the form $\gamma(x,y) = \sum_{k=1}^{K}\exp(c_k\sin(k\pi x)\sin(k\pi y))$ with $K = mod(\bar{K})$, where $\bar{K} \sim \mathrm{U}[1,5]$ and $c_k \sim \mathrm{U}[-1,1]$. An example of a sampled conductivity is shown on the left of Fig. 2(a). For each conductivity, the $L = 20$ input voltages g_l are used as Dirichlet conditions in (13), leading to $NL = 20000$ total data samples $\{(\gamma^{(n)}, g^{(n)})\}_{n=1}^{NL}$. Importantly, the training of IGNO does not require computing the corresponding currents, avoiding expensive PDE simulations.

For the inverse problem, we consider two target scenarios:

- **In-distribution target:** γ is drawn from the same distribution used for training (with different random seed).
- **Out-of-distribution target:** γ is sampled from a shifted distribution with $\bar{K} \sim \mathrm{U}[1,5]$ and $c_k \sim \mathrm{U}[1,1.5]$.

In both cases, operator-based measurements $\mathcal{M}_{op}$ are generated by solving (13) with FEM and adding Gaussian noise. We consider three noise levels: SNR= 50 (low), SNR= 25 (medium), and SNR= 15 (high).

5.4.2 Model Setup

To extract latent representations from input coefficients, a uniform grid Ξ_a with size 32×32 is used, which leads to the coefficient encoder input being $\boldsymbol{a} = a(\Xi_a)$. Since this problem involves multiple boundary conditions, a boundary encoder $E_{\boldsymbol{\theta}_{\boldsymbol{\beta}_2}}$ is needed to encode boundary inputs. Given that only $L = 20$ boundary conditions are used, this encoder is implemented as a one-hot encoding rather than a trainable network. To enforce boundary conditions automatically, we define a mollifier for the solution decoder output. Specifically, for solution prediction with boundary condition g_l:

$$u_{pred} = \mathcal{G}_{\boldsymbol{\theta}_u}(\boldsymbol{\beta}) \cdot f + g_l, \tag{15}$$

where $f(x, y) = \sin(\pi x)\sin(\pi y)$. This ensures that the predicted solution satisfies the boundary conditions without introducing an explicit boundary loss term.

Below, we provide details of the network architectures:

- **Coefficient Encoder $E_{\boldsymbol{\theta}_{\boldsymbol{\beta}_1}}$.** The coefficient encoder $E_{\boldsymbol{\theta}_{\boldsymbol{\beta}_1}}$ consists of a CNN followed by an FFCN. The CNN has four hidden layers with 64 output channels per layer, kernel size $(3, 3)$, and a stride of 2. The subsequent FFCN has two hidden layers of 64 neurons each. SiLU activations are applied to all hidden layers, and Tanh is used at the output.
- **Boundary Encoder $E_{\boldsymbol{\theta}_{\boldsymbol{\beta}_2}}$.** To encode the boundary conditions $\{g_l\}_{l=1}^{L}$, we use a one-hot encoding: for boundary g_l, the latent vector $\boldsymbol{\beta}_2 = e_l \in \mathbb{R}^{20}$, where e_l is the l-th unit vector. This enables IGNO to handle operator-based measurements with multiple boundary conditions efficiently.
- **Solution Decoder $\mathcal{G}_{\boldsymbol{\theta}_u}$.** The solution decoder adopts the MultiONet architecture. Both branch and trunk networks are FFCNs with five hidden layers of 100 neurons per layer. The custom Tanh_Sin activation is applied to all hidden layers.
- **Coefficient Decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$.** The coefficient decoder $\mathcal{G}_{\boldsymbol{\theta}_a}$ shares the same architecture as $\mathcal{G}_{\boldsymbol{\theta}_u}$.
- **The NF model $F_{\boldsymbol{\theta}_{NF}}$.** The NF model consists of three flow steps, each parameterized by an FFCN with two hidden layers of 128 neurons and SiLU activations.

5.4.3 Latent-space Optimization Details

For In-distribution targets, gradients of the objective $\mathcal{F}$ with respect to $\boldsymbol{\beta}_1$ are computed via PyTorch automatic differentiation. Weights are set as $\rho_{data} = 100$ and $\rho_{pde} = 1$. The ADAM optimizer is used with an initial learning rate of 0.01, reduced by half every 25 steps, for a total of 200 updates. For Out-of-distribution targets, weights are set to $\rho_{data} = 100$ and $\rho_{pde} = 0.001$. The ADAM optimizer is used with an initial learning rate of 0.01, decreased by a factor of 2/3 every 100 steps, with a total of 200 updates.

6 Supplementary Tables

6.1 Continuous Coefficient Recovery with Solution-based Measurements

6.1.1 Table 1: In-distribution case

IGNO achieves 3 to 7 times lower reconstruction errors than the PI-DIONs method across all noise levels for in-distribution continuous coefficient recovery. Even under severe noise (SNR=15 dB), IGNO maintains RMSE below 1.2%, while the PI-DIONs method exceeds 4%.

Table 1: RMSEs obtained by different methods under different noise levels in recovery of **continuous coefficients** with solution-based measurements. (**In-distribution case**)

	SNR = 50	SNR = 25	SNR = 15
IGNO	0.0056	0.0061	0.0115
PI-DIONs	0.0366	0.0360	0.0469

6.1.2 Table 2: Out-of-distribution case

For out-of-distribution targets with frequency parameters entirely outside training range, IGNO maintains strong generalization with only modest performance degradation compared to in-distribution cases. The PI-DIONs method shows substantial error increases (4.8-5.8%), while IGNO remains 3 to 5 times more accurate.

Table 2: RMSEs obtained by different methods under different noise levels in recovery of **continuous coefficients** with solution-based measurements. (**Out-of-distribution case**)

	SNR = 50	SNR = 25	SNR = 15
Inv-GenNO	0.0064	0.0113	0.0194
PI-DIONs	0.0476	0.0496	0.0583

6.2 Piecewise-Constant Coefficient Recovery with Solution-based Measuremen

6.2.1 Table 3: In-distribution case

The cross-correlation indicator I_{corr} measures morphological similarity, with values near 1 indicating strong agreement between reconstructed and true phase topologies. IGNO achieves $I_{corr} > 0.95$ across all noise levels, indicating accurate phase topology recovery. The PI-DIONs method fails with $I_{corr} \sim 0.72$, indicating essentially random reconstructions due to the inability to handle undefined gradients at discontinuities.

Table 3: Cross-correlations I_{corr} obtained by different methods under different noise levels in recovery of **piecewise-constant coefficients** with solution-based measurements. (**In-distribution case**)

	SNR = 50	SNR = 25	SNR = 15
IGNO	0.968	0.960	0.952
PI-DIONs	0.718	0.714	0.730

6.2.2 Table 4: Out-of-distribution case

For out-of-distribution piecewise-constant targets with different Gaussian process correlation structures producing more complex phase geometries, IGNO maintains $I_{corr} > 0.91$ even under high noise. The PI-DIONs method continues to fail ($I_{corr} \sim 0.78$), demonstrating an inability to generalize for discontinuous inverse problems.

Table 4: Cross-correlations I_{corr} obtained by different methods under different noise levels in recovery of **piecewise-constant coefficients** with solution-based measurements. (**Out-of-distribution case**)

	SNR = 50	SNR = 25	SNR = 15
IGNO	0.961	0.954	0.912
PI-DIONs	0.779	0.779	0.777

6.3 The EIT Problem with Operator-based Measurements

6.4 Table 5: the EIT problem

For the operator-based EIT problem, IGNO demonstrates robust performance across both distribution types. In-distribution reconstruction achieves sub-0.5% error even under low noise, degrading gracefully to 2.8% at SNR=15 dB. Out-of-distribution targets show only modest error increases (2.2-2.9%), indicating strong extrapolation capabilities. The PI-DIONs method cannot handle operator-based measurements and does not apply to this problem.

Table 5: RMSEs obtained by the proposed IGNO in solving the **EIT** problem in both In-distribution and Out-of-distribution cases under different noise levels.

	SNR = 50	SNR = 25	SNR = 15
In-distribution	0.0044	0.0128	0.0277
Out-of-distribution	0.0219	0.0246	0.0293

7 Supplementary Figures

7.1 Figure 1: The MultiONet architecture

The MultiONet architecture in Fig. 1 shows branch and trunk networks with multi-layer feature aggregation.

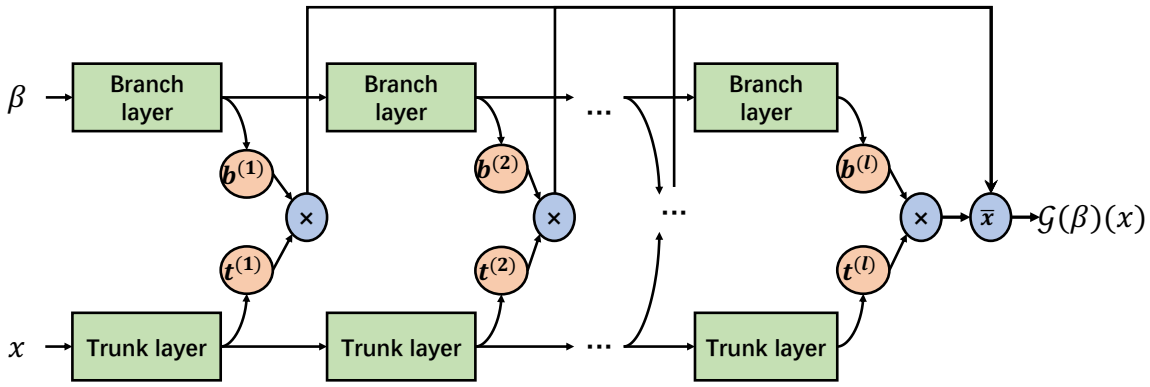

Fig. 1: The MultiONet architecture.

7.2 Figure 2: Illustration of operator-based measurements in the EIT problem

In the EIT problem, the unknown conductivity field γ is inferred from the DtN operator Λ_γ, which maps imposed boundary voltages g to the corresponding boundary currents $\gamma \frac{\partial u}{\partial \vec{n}}|_{\partial\Omega}$. This operator is approximated using $L = 20$ distinct boundary voltage patterns and measurements collected from $m = 128$ equally spaced boundary sensors. Figure 2(a) illustrates an example conductivity field γ (left) and the corresponding boundary sensor configuration (right). Figure 2(b) shows three representative boundary conditions g_l, the resulting current responses $\Lambda_\gamma[g_l]$, and the corresponding PDE solutions u for $l = 1, 10, 20$.

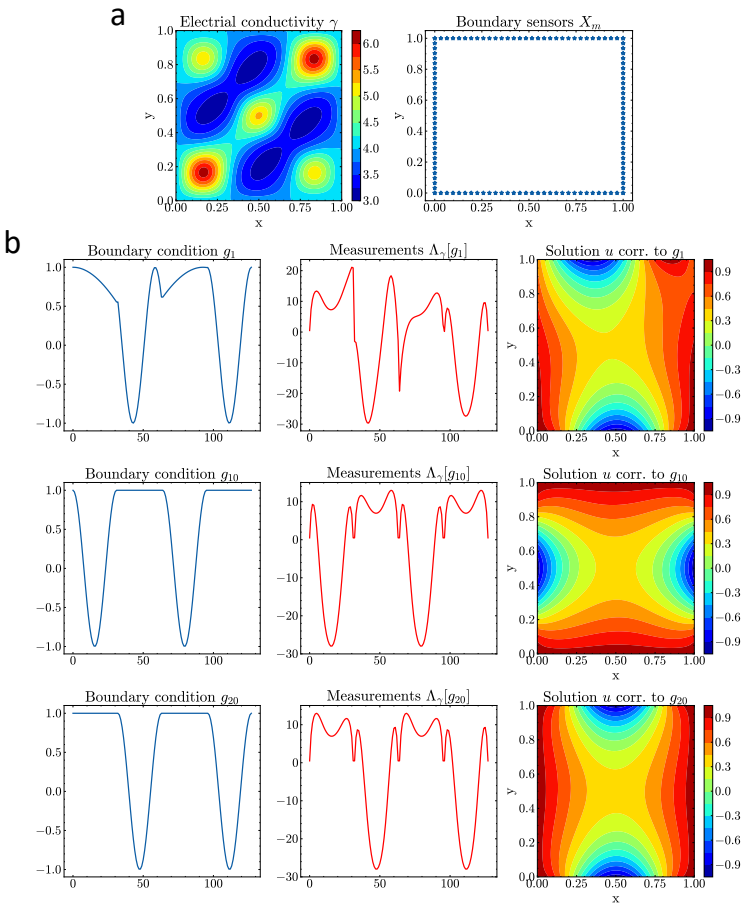

Fig. 2: Illustration of operator-based measurements in the EIT problem: (a) An example of conductivity γ (left) and the boundary sensors $\boldsymbol{X}_m$ (right); (b) Examples of boundary conditions g_l (left), corresponding current measurements $\Lambda_\gamma[g_l]$ (middle), and PDE solutions u (right), when $l = 1, 10, 20$.

References

[1] Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. arXiv preprint arXiv:1605.08803 (2016)

[2] Cho, S.W., Son, H.: Physics-informed deep inverse operator networks for solving pde inverse problems. arXiv preprint arXiv:2412.03161 (2024)

[3] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895 (2020)

[4] Zang, Y., Koutsourelakis, P.-S.: Dgenno: a novel physics-aware neural operator for solving forward and inverse pde problems based on deep, generative probabilistic modeling. Journal of Computational Physics **538**, 114137 (2025)