GAIA: Geometry-Adaptive Operator Learning for Forward and Inverse Problems

Meenakshi Krishnan
University of Maryland, College Park
mkrishn9@umd.edu

Pranav Pulijala*
University of Maryland, College Park

Ke Chen
University of Delaware

Haizhao Yang
University of Maryland, College Park

Ramani Duraiswami
University of Maryland, College Park

Abstract

Operator learning for partial differential equations (PDEs) on arbitrary geometries builds fast neural surrogates for large-scale simulation. While recent geometry-adaptive neural operators have made substantial progress, they are mainly designed for forward problems in which inputs and outputs share the same spatial domain. This limits their applicability for boundary value problems (BVPs) and inverse problems, where inputs and outputs may live on different domains. We introduce the Geometry-Adaptive Integral Autoencoder (GAIA), an operator learning model that encodes the domain boundary and the interior field distribution into geometry tokens, and conditions integral transform layers on these tokens via cross-attention, allowing the kernel to adapt locally to geometric features. This yields an architecture capable of learning solution operators for forward and inverse problems on arbitrary domains, solving each in a single pass at inference without per-domain optimization, iterative refinement, or graph construction. We evaluate GAIA on seven 2D and 3D benchmarks, four of which are new or substantially extended benchmarks for inverse problems and BVP: electrical impedance tomography, optical tomography, 3D Darcy flow on varying geometries, and a modified setting of the Poisson BVP on mechanical components benchmark (MCB). GAIA sets new state-of-the-art results on every inverse and BVP task, reducing median relative L^2 error by 64% on airfoil flow reconstruction and 22% on EIT relative to the next best amortized method, and outperforming all baselines on every shape category of MCB. On other forward problems, GAIA is competitive with specialized solvers while maintaining stable accuracy across point resolutions on which transformer-based baselines degrade.

1 Introduction

Partial differential equations are used to model problems across science and engineering, from fluid dynamics [1, 2] to medical imaging [3]. Traditional solvers are accurate but expensive, particularly in multi-query settings such as inverse problems, optimal design, and uncertainty quantification, where the forward model is invoked thousands of times. Early neural methods developed to address these costs, such as Physics-Informed Neural Networks (PINNs) [4] and deep inverse priors [5], are instance-specific, requiring retraining for every new boundary condition or parameter change.

*Significant technical contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To overcome the need for repeated retraining, operator learning models instead learn maps between function spaces directly. However, classical neural operators such as FNO [6] and DeepONet [7] require regular grids or fixed discretizations. Many problems in structural analysis, medical tomography, and aerodynamic design are posed on domains whose geometry varies across problem instances, in both forward and inverse settings. A broadly applicable neural proxy must thus apply to arbitrary varying geometries without retraining.

Recent geometry-adaptive operators, such as GAOT [8], GINO [9], Transolver [10], handle unstructured point clouds via graph neural networks or transformer attention. However, these methods are mostly designed for forward modeling. Some, such as Transolver, require the input and output to share the same spatial discretization, precluding problems where the input and output live on different point sets - as in BVPs and inverse problems with boundary measurements. Applying these architectures to the latter requires computationally expensive iterative optimization loops over forward solvers. Others, such as GAOT, are designed and evaluated exclusively for forward problems. While GINO can in principle handle inverse problems, its reliance on graph message passing incurs substantial memory and computational overhead.

We introduce the Geometry-Adaptive Integral Autoencoder (GAIA), a unified architecture for solving both forward and inverse problems on arbitrary domains. Building on IAE-Net [11], we introduce a dual-pathway tokenization that encodes both the domain boundary and the interior field distribution, and condition the integral kernels on these tokens via multi-head cross-attention. Unlike methods that encode geometry implicitly through per-sample neural field fitting [12] or require expensive graph construction [9, 8], this provides explicit, spatially adaptive geometric context at each integration point. The query structure accommodates problems where the input and output live on different spatial domains, enabling efficient single-pass solution of inverse problems without iterative optimization.

In summary:

- **Unified forward/inverse geometry-adaptive framework.** We propose GAIA, a geometry-adaptive integral operator that conditions learned kernels on domain geometry via cross-attention, enabling both forward and inverse problems on arbitrary geometries to be solved in a single pass.
- **New benchmarks for varying-geometry problems.** We introduce four new or substantially extended benchmarks for varying-geometry problems, including inverse problems such as Electrical Impedance Tomography (EIT) and Optical Tomography (OT). All datasets are publicly available.
- **Empirical validation.** We validate GAIA on seven time-independent PDE benchmarks spanning 2D and 3D problems. GAIA achieves state-of-the-art on all inverse and BVP tasks, reducing median relative L^2 error by 22% on EIT and 64% on airfoil relative to the next best method, while remaining competitive on forward benchmarks with stable accuracy across resolutions where transformer baselines degrade.

2 Related Work

2.1 Operator Learning

Early scientific machine learning methods such as Physics-Informed Neural Networks [4, 13–17] and CNN-based surrogates [18–20] are either instance-specific or restricted to regular grids; we focus on operator learning. Neural operators learn mappings between function spaces, and can, for certain classes of PDEs, do so efficiently without suffering from the curse of dimensionality [21]. DeepONet [7] uses a branch-and-trunk decomposition but requires inputs sampled on a fixed grid. FNO [6] parameterizes integral kernels in the spectral domain via the Fast Fourier Transform (FFT), achieving good performance on regular grids. Numerous extensions improve scalability and discretization handling through low-rank or factorized kernels [22–25], graph message passing [26, 27], and wavelet bases [28]. IAE-Net [11] achieves discretization invariance through a stack of integral autoencoder blocks with data-driven kernels, but its kernels depend only on local coordinates and function values with no access to global domain geometry.

For inverse problems, neural operators have been used either as differentiable surrogates within iterative optimization, MCMC, or diffusion-based sampling loops [29–33], or to directly learn amortized inverse maps from measurement-to-parameter data pairs, enabling single-pass inference [34, 35]. The former does not avoid the cost of iterative sampling at inference. GAIA belongs in the latter group. Neural Inverse Operators [34] learn operator-to-function mappings via compositions of

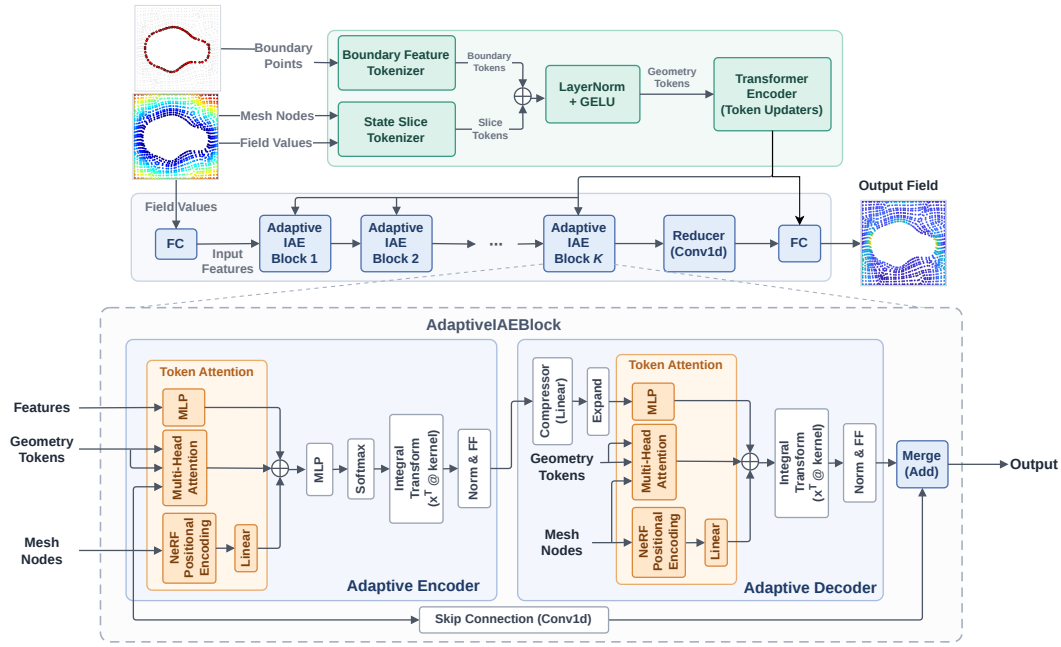


Figure 1: GAIA architecture. *Top:* Two complementary tokenizers produce geometry tokens from boundary points and interior mesh-field values, refined by transformer encoder layers. *Middle:* A stack of geometry-conditioned Adaptive IAE blocks with DenseNet-style skip connections maps input features to the output field. *Bottom:* Each block follows an encode–process–decode pattern with integral-transform kernels conditioned on geometry tokens via cross-attention. Separate encoder and decoder query sets enable inverse problems and BVPs.

DeepONet and FNO; invertible architectures such as iFNO [35] jointly learn forward and inverse maps through shared invertible blocks. However, both are tied to fixed regular grids due to their reliance on FFT. Unlike these deterministic amortized inverse methods, which produce point estimates, Bayesian approaches characterize posterior distributions over unknown parameters [36, 37]. Recent work has explored probabilistic operator learning for inverse problems and conditional generative models for PDE simulations [38–40].

2.2 Geometry-Adaptive Operators

Geometry-adaptive operators take three broad approaches. **Coordinate-transform methods** like Geo-FNO [41] learn a deformation from the physical domain to a regular latent grid; this works for topologically simple domains but fails on complex shapes. **Graph- and message-passing methods** including GINO [9], GAOT [8], and RIGNO [42] handle unstructured point clouds natively but require per-sample graph construction that dominates inference cost. Other recent graph-based approaches include HAMLET [43], which employs graph transformers with a Galerkin cross-attention decoder, and AMG [44], which constructs multiscale and physics-informed graphs. MINO [45] develops a mesh-informed operator backbone for functional generative modeling through flow matching. **Implicit-representation methods** [46, 47] include CORAL [12], which encodes fields as implicit neural representations and learns maps between latent codes. Transolver [10] and LNO [48] are **attention-based methods** which project mesh points into a small set of learnable physical states via attention; broader transformer operators include GNOT [49] and OFormer [50, 51]. There are also specialized architectures such as Neural Green's Function [52] for symmetric linear PDEs.

Among these, only LNO has been demonstrated on inverse tasks with differing input-output domains; CORAL and GeoFNO demonstrate iterative optimization over their forward surrogates for inversion. GAIA differs from all three: unlike CORAL it does not require per-sample neural-field fitting; unlike GINO it constructs no graph; and unlike LNO, which uses attention to compress spatial information into a fixed set of latent states, GAIA conditions the integral kernel itself on geometry at each query point via cross-attention. Our slice tokenizer adapts the soft-clustering aggregation of Transolver [10]; the boundary tokenizer and the use of both token pathways to condition integral kernels via cross-attention are novel, and the results demonstrate the architecture's efficacy.

3 Methods

We formalize operator-learning, recap the IAE-Net model (§3.1), and describe GAIA (§3.2).

Problem Formulation. We consider a generic time-independent PDE,

$$\mathcal{D}(c, u) = q, \quad \forall x \in D \subset \mathbb{R}^d, \quad \mathcal{B}(u) = u_b, \quad x \in \partial D, \tag{1}$$

where $u : D \to \mathbb{R}^m$ is the solution, $c \in \mathcal{C}$ represents the physical parameters, $q \in \mathcal{F}$ is the forcing term, u_b prescribes the boundary values; and $\mathcal{D}$ and $\mathcal{B}$ are the differential and boundary operators, respectively. Also, let χ_D denote the domain indicator. Here $\mathcal{U}$, $\mathcal{C}$, and $\mathcal{F}$ denote suitable function spaces on D. Encapsulating the problem setup into $a = (c, q, u_b, \chi_D) \in \mathcal{A}$, the *forward operator* $\mathcal{S} : \mathcal{A} \to \mathcal{U}$ maps $a \mapsto u$.

In the inverse setting, we do not have access to the full solution field u. We only have access to observables, often measured at the domain boundaries. This is formalized by a measurement operator $\mathcal{M} : \mathcal{U} \to \mathcal{V}$ mapping u to an observation $v = \mathcal{M}(u)$. The generalized forward operator $\mathcal{F} = \mathcal{M} \circ \mathcal{S} : \mathcal{A} \to \mathcal{V}$ maps parameters directly to observables, and the *inverse operator* $\mathcal{F}^\dagger : \mathcal{V} \to \mathcal{C}$ recovers $c = \mathcal{F}^\dagger(v, \chi_D)$ from these measurements.

Throughout, let $\Psi : \mathcal{X} \to \mathcal{Y}$ denote the target mapping ($\Psi \equiv \mathcal{S}$ in the forward case, $\Psi \equiv \mathcal{F}^\dagger$ in the inverse case). Our goal is to approximate Ψ given access to data pairs $(f^{(i)}, g^{(i)})$ with $g^{(i)} = \Psi(f^{(i)})$.

3.1 IAE-Net

IAE-Net [11] learns an operator $\Psi : \mathcal{X} \to \mathcal{Y}$ between function spaces on compact domains $\Omega_x \subset \mathbb{R}^{d_x}$ and $\Omega_y \subset \mathbb{R}^{d_y}$, from finite discretizations $S_x \subset \Omega_x$, $S_y \subset \Omega_y$ whose cardinalities may vary across samples. The architecture is a recursive stack of L densely-connected autoencoder blocks, performing a series of transformations $f \to a_0 \to a_1 \to \cdots \to a_L \to g$, with each intermediate function a_i defined on $\Omega_a = [0, 1]^{d_a}$. Each block performs an encode–process–decode transformation $a \to v \to u \to b$, where v and u live on a fixed latent domain $\Omega_z = [0, 1]^d$, discretized on a grid $S_z = \{z_j\}_{j=1}^m$ independent of the input:

$$v(z) = \int_{\Omega_x} \phi_1(a(x), x, z; \theta_1)\, a(x)\, dx, \; u(z) = \phi_0(v(z); \theta_0), \; b(y) = \int_{\Omega_z} \phi_2(u(z), y, z; \theta_2)\, u(z)\, dz. \tag{2}$$

Here, ϕ_1 is the encoding kernel, ϕ_0 a latent-space MLP, ϕ_2 the decoding kernel, and $y \in \Omega_a$. The integrals are approximated as discrete sums, mapping variable-cardinality inputs to the fixed grid S_z. This allows IAE-Net to natively handle different input resolutions. Blocks are connected with DenseNet-style skip connections.

The kernels ϕ_1, ϕ_2 in equation 2 of IAE-Net depend only on local coordinates and the function values $a(x)$, with no access to global domain geometry. For elliptic PDEs, the Green's function governing the solution operator depends on the domain shape, so a geometry-agnostic kernel cannot capture this dependence. As a result, the same kernel is applied regardless of the domain geometry. Additionally, while IAE-Net's integral transform structure can in principle support differing input and output domains, this has not been explored for boundary-to-interior mappings.

3.2 Geometry-Adaptive Integral Autoencoder (GAIA)

GAIA conditions IAE-Net's integral kernels on the domain geometry. We describe tokenizing the boundary points and interior fields, the conditioning mechanism, and the resulting blocks for forward and inverse problems. To address the geometry-agnostic limitation of IAE-Net described in §3.1, the kernels are conditioned on a tokenized representation of the actual domain boundary and interior field distribution at inference time, with each spatial query point attending selectively via cross-attention, providing geometric context that is spatially adaptive. Decoupling query points from integration points lets the encoder integrate over X_{bnd} while the decoder projects onto X_{dom}, enabling single-pass solve of inverse problems and BVPs.

Geometry Tokenization The geometry conditioning is constructed from two complementary token pathways: *boundary feature tokens*, which encode the domain shape from boundary coordinates alone, and *state slice tokens*, which encode the joint distribution of spatial coordinates and field values in the domain. Let $X_{\text{bnd}} \in \mathbb{R}^{M \times d}$ be the boundary/sensor coordinates and $X_{\text{dom}} \in \mathbb{R}^{N \times d}$ the interior

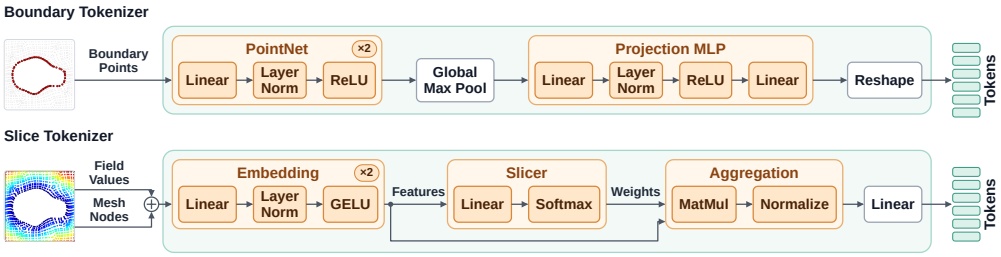


Figure 2: **The two tokenizer pathways:** GAIA encodes geometry through two complementary token sets. *Top – Boundary tokenizer:* Boundary point coordinates are processed by a PointNet-style encoder with shared MLPs and global max-pooling to give permutation-invariant tokens that summarize the global domain shape. *Bottom – Slice tokenizer:* The interior mesh coordinates and their associated field values are jointly embedded and assigned to soft clusters; mass-normalized aggregation produces tokens that summarize the spatially varying domain physics.

mesh; let U be the associated input field or measurements. The two pathways produce tokens that are concatenated into a unified $T \in \mathbb{R}^{K \times d_t}$; K is the total token count, d_t the token dimension.

Boundary feature tokenization. The boundary coordinates X_{bnd} are mapped to K_{bnd} tokens $T_{\text{bnd}} \in \mathbb{R}^{K_{\text{bnd}} \times d_t}$ using a PointNet-style encoder [53], which applies shared MLPs followed by permutation-invariant global max-pooling, then projects and reshapes the resulting vector into K_{bnd} tokens:

$$T_{\text{bnd}} = \text{Proj}\left(\max_{p \in X_{\text{bnd}}} \text{MLP}(p) \right). \tag{3}$$

State slice tokenization. Adapting the soft-clustering aggregation of Transolver [10] to inputs that include both spatial coordinates and physical values, we let $X \in \mathbb{R}^{P \times d}$ be a point cloud (instantiating as X_{dom} for forward problems or X_{bnd} for inverse problems) and U the associated values. The concatenated state $[X, U]$ is mapped to K_{slice} soft-cluster tokens via:

$$W = \text{Softmax}(\text{MLP}_{\text{cluster}}([X, U])), \quad H = \text{MLP}_{\text{feat}}([X, U]), \tag{4}$$

$$T_{\text{slice}} = \text{diag}(W^\top \mathbf{1})^{-1} W^\top H, \tag{5}$$

with mass-normalized aggregation; $\mathbf{1} \in \mathbb{R}^P$ is a vector of ones. A linear projection brings T_{slice} to the standard token dimension d_t. The initial token set $T^{(0)} = [T_{\text{bnd}}, T_{\text{slice}}]$ is normalized and passed through GELU; between Adaptive IAE blocks, tokens are refined by standard transformer encoder layers, so that each block sees an updated token set $T^{(l)}$.

Geometry-conditioned integral transforms. GAIA modifies the encoding and decoding integral transforms (Eq. 2) so that the learnable kernel is conditioned on the geometry tokens $T^{(l)}$. Spatial coordinates are first lifted via Fourier feature encodings $\gamma(x) \in \mathbb{R}^{2d_F + d}$ with F frequency bands [54]. To inject geometric context into the kernel, we define a spatially-queried context vector $C(x)$ via cross-attention between the encoded coordinates and the geometry tokens:

$$C(x) = \text{Softmax}\left(\frac{(\gamma(x) W_Q)(T^{(l)} W_K)^\top}{\sqrt{d_k}} \right) (T^{(l)} W_V), \tag{6}$$

where W_Q, W_K, W_V are learnable projections and d_k the scaling factor. The kernel is parameterized by an MLP taking the concatenated spatial features and geometric context:

$$K(x, z; \theta) = \text{MLP}_{\text{kernel}}([\gamma(x), \gamma(z), U(x), C(x)]), \tag{7}$$

where $U(x)$ is the input feature at point x. The integrals in (2) with this kernel are approximated as:

$$I(z) = \frac{1}{|X|} \sum_{x_j \in X} K(x_j, z; \theta)\, U(x_j), \tag{8}$$

followed by a learned residual connection and GELU non-linearity.

Adaptive IAE Block. GAIA uses these geometry-conditioned integral transforms as building blocks. Each Adaptive IAE block follows the encode–process–decode structure (equation 2) of IAE-Net, but with kernels conditioned on geometry tokens via cross-attention (Equation 7). The encoder maps from the N-point input to m fixed latent modes, a pointwise MLP processes the latent representation, and the decoder maps back to N points using the same conditioned kernel queried at the output coordinates. This bottleneck gives per-block complexity $O(Nm)$ with $m \ll N$.

Main architecture. For forward problems where input and output share the same spatial domain, the Adaptive IAE blocks perform volume integrals over the domain mesh. For inverse problems and BVPs, an additional module first transfers information from boundary measurements to the interior.

Forward problems. The input field $U \in \mathbb{R}^{N\times c}$ is defined on X_{dom}. The network applies L Adaptive IAE blocks with dense skip connections: the input $H_{\text{in}}^{(l)}$ to the l-th block is the concatenation of all preceding outputs, compressed back to width w via 1D convolution. The final prediction comes from passing the densely concatenated outputs through a reduction layer and a GELU MLP.

Inverse problems and BVPs. Here the input is sparse measurements $U \in \mathbb{R}^{M\times c_{\text{obs}}}$ at boundary locations X_{bnd}, and the target lives on the dense interior mesh X_{dom}. To bridge this gap, an Observation-to-Domain Decoder first transfers information from the boundary to a fixed set of K_{modes} learnable latent coordinates $Z \in \mathbb{R}^{K_{\text{modes}}\times d}$, then projects from the latent set onto the interior mesh. Concretely, the boundary data is embedded as $H_{\text{bnd}} = \text{MLP}_{\text{in}}(U) \in \mathbb{R}^{M\times w}$. A geometry-conditioned boundary integral lifts it to the latent set:

$$H_{\text{mode}}(z_k) = \frac{1}{|X|}\sum_{x_j\in X_{\text{bnd}}} K_{\text{enc}}(x_j, z_k; T^{(0)})\, H_{\text{bnd}}(x_j) \qquad \forall z_k \in Z. \tag{9}$$

A fixed-size MLP processes the latent representation, $\tilde{H}_{\text{mode}} = \text{MLP}(H_{\text{mode}})$, after which a second geometry-conditioned integral projects onto X_{dom}:

$$H^{(0)}(y_i) = \frac{1}{|Z|}\sum_{z_k\in Z} K_{\text{dec}}(y_i, z_k; T^{(1)})\, \tilde{H}_{\text{mode}}(z_k)\, \Delta z_k, \qquad \forall y_i \in X_{\text{dom}}. \tag{10}$$

The resulting $H^{(0)} \in \mathbb{R}^{N\times w}$ enters the stack of Adaptive IAE blocks identically to the forward case.

4 Results

We evaluate GAIA against five geometry-adaptive baselines—GINO [9], CORAL [12], Transolver [10], GAOT [8], and LNO [48]—on seven benchmarks spanning forward, inverse, and BVP settings. Four of these benchmarks are new or extended contributions (Section 4.1); the remaining three are standard benchmarks. Transolver requires shared input/output discretization and is omitted from inverse and BVP comparisons. NIO [34], the closest amortized inverse-operator method, is FNO-based and operates on fixed regular grids; it cannot be applied to our varying-geometry benchmarks by construction. For the 3D Poisson BVP we additionally compare against NGF [52], which is restricted to forward problems for linear symmetric PDEs.

For each task, we train GAIA using the prescribed architecture for forward (including BVPs) or inverse problems. All models are trained on a single NVIDIA A6000 GPU. We minimize the relative L^2 error,

$$\mathcal{L}(\theta) = \mathbb{E}_{f\sim\mu}\left[\frac{\|(\Psi_n(f;\theta) - \Psi(f))\odot M\|_2}{\|\Psi(f)\odot M\|_2 + \epsilon}\right], \tag{11}$$

where $\odot$ denotes the Hadamard product. We use masking for varying mesh sizes; M is a binary mask indicating valid interior nodes. Optimizers, learning rates, schedules, and full architectural hyperparameters for both GAIA and baselines are reported in Appendix B.

4.1 Benchmarks

Inverse problems. *Electrical Impedance Tomography (EIT).* EIT is an imaging technique that reconstructs the internal conductivity $a(x)$ from boundary electrical measurements, solving an ill-posed [55] inverse problem governed by $\nabla\cdot(a(x)\nabla u(x)) = 0$ on $\Omega \subseteq \mathbb{R}^2$. We apply L=20 Dirichlet excitation patterns at M=272 sensors on star-shaped domains and measure the Neumann response; the inverse operator maps the resulting Dirichlet-to-Neumann data Λ_a to $a(x)$. We extend the fixed-geometry setting of [34] to varying star-shaped domains.

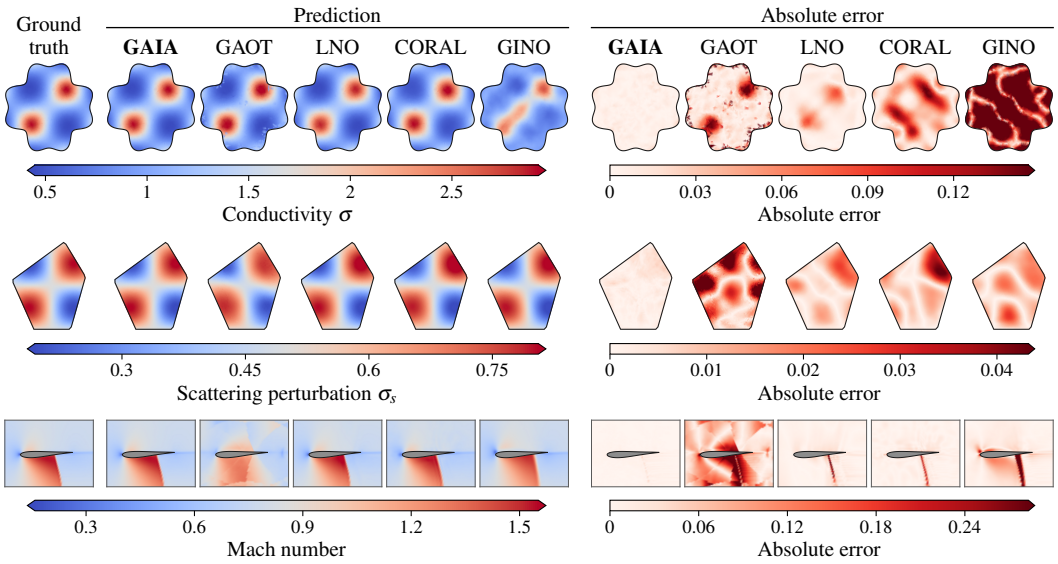


Figure 3: Model predictions on inverse benchmarks. **Top:** EIT conductivity reconstruction. **Middle:** OT scattering coefficient reconstruction. **Bottom:** Airfoil Mach field reconstruction from sparse noisy observations. GAIA captures fine-scale features like the conductivity patterns and transonic shock that baselines either mislocate (GINO) or diffuse (CORAL, LNO).

Optical Tomography (OT). OT is an imaging technique that recovers the scattering coefficient $\sigma_s(x)$ of a tissue-like medium from boundary light measurements. The transport of photons through the medium $\Omega \subset \mathbb{R}^2$ is governed by the stationary radiative transport equation (RTE):

$$v \cdot \nabla_x u(x,v) + \sigma_t(x)\, u(x,v) = \sigma_s(x) \int_{S^{d-1}} \Phi(v \cdot v')\, u(x,v')\, dv', \quad x \in \Omega, \tag{12}$$

where $u(x,v)$ is the photon density at position x traveling in direction v, $\sigma_t = \sigma_s + \sigma_a$ is the total attenuation, and Φ is the scattering phase function. The boundary observable is the Albedo operator $\Lambda : L^1(\partial\Omega) \to L^1(\partial\Omega)$, which maps incoming illumination $\phi(x)$ to the outgoing flux $J_+(x) = \int_{v \cdot n_x > 0} (v \cdot n_x)\, u(x,v)\, dv$. The forward-peaked anisotropic scattering is modeled via the Henyey-Greenstein phase function. The Albedo operator is discretized as a dense source-to-receiver matrix. We use difference imaging [56]: the Albedo operator Λ_0 of a homogeneous baseline ($\sigma_{s,0}$=1) is used as a reference, and the inverse operator maps the difference $\Delta\Lambda = \Lambda - \Lambda_0$ to the interior scattering perturbation $\delta\sigma_s(x)$. Domains are random convex pentagons. This is a new dataset.

Airfoil reconstruction. We use the transonic airfoil dataset of [41], which contains steady-state solutions of the compressible Euler equations at $M_\infty = 0.8$ over randomly deformed NACA-0012 profiles. We reformulate a sparse-to-full inverse reconstruction task: only 10% of mesh nodes are observed, and at inference, the observations are corrupted by 1% relative Gaussian noise. The model uses the sparse noisy observations to recover the full Mach field over the domain.

Forward problems. *3D Poisson BVP on mechanical components.* We solve $-\Delta u = f$ on 3D mechanical parts from the MCB dataset [57], spanning four shape categories: gears, nuts, fittings, and screws & bolts. We adapt the setting and meshes of [52], but formulate a pure BVP with fixed source and varying randomized Dirichlet boundary conditions, solved with FEniCS [58]. The target operator thus maps the Dirichlet boundary function to the solution on the interior.

3D Darcy flow. We learn the map from a log-normal permeability field $\kappa(x)$ to the pressure $u(x)$ satisfying $-\nabla \cdot (\kappa \nabla u) = 1$ with homogeneous Dirichlet conditions on star-shaped 3D domains defined by random spherical harmonic expansions.

Poisson-Gauss and *Elasticity*. These are standard benchmarks with shared input–output discretizations from [42] and [41] respectively. Details are in Appendix A.

4.2 Discussion

Inverse Problems Table 1 reports median relative L^2 error on inverse and BVP benchmarks; GAIA achieves the lowest error on every task. Against the next best amortized method, GAIA reduces

error by 22% on EIT (0.71% vs. 0.91% for LNO), 18% on OT (1.45% vs. 1.76% for LNO), and 64% on the Airfoil task (0.58% vs. 1.59% for CORAL). The Airfoil margin is the largest, likely because baselines struggle with the combination of noise, sparsity, and geometric variation. Visual reconstructions in the Appendix (Figures 9-15) show GAIA recovering fine-scale features (such as oscillatory conductivity patches in EIT, transonic shock structure in Airfoil) that baselines either smear or miss entirely.

Boundary Value Problems Figure 4 visualizes representative reconstructions on some MCB shapes. On the 3D Poisson BVP (Table 1), GAIA outperforms all baselines on every shape category, with the largest gains on gears (3.63% vs. 5.33% for NGF) and screws & bolts (3.45% vs. 5.22% for LNO). NGF is purpose-built for forward problems on linear symmetric PDEs; that GAIA exceeds it on every shape category, while remaining a general-purpose architecture, suggests the geometry-conditioned integral kernels carry meaningful advantage on real geometries.

(a) Inverse Problems

Dataset	GINO	CORAL	LNO	GAOT	GAIA
EIT	26.42	1.65	0.91	8.02	**0.71**
OT	1.90	3.06	1.76	5.79	**1.45**
Airfoil	4.79	1.59	2.49	9.53	**0.58**

(b) Mechanical Components BVP

Shape	CORAL	NGF	LNO	GAIA
Fitting	18.81	11.25	14.88	**10.30**
Gear	13.81	5.33	5.91	**3.63**
Nut	20.69	5.34	6.95	**4.95**
Screws/Bolts	20.47	6.31	5.22	**3.45**

Table 1: Comparison on (a) inverse problems and (b) the Mechanical Components BVP benchmark. Median relative L^2 error (%). GAIA achieves the lowest error across all benchmarks.

Other Forward Problems Table 2 reports forward problem results on benchmarks where input and output share a discretization. GAIA achieves the lowest error on Poisson-Gauss (0.77%) and is competitive with the leading method on Elasticity (1.39% vs. 0.99% for GAOT). On 3D Darcy flow, Transolver achieves lower error than GAIA at the training resolution (0.79% vs. 1.24%). This advantage, however, is contingent on evaluation matching the training discretization. As shown in §4.3, both Transolver and GAOT degrade by an order of magnitude when evaluated at coarser resolutions on the Elasticity benchmark where they look strongest in-distribution; GAIA's accuracy is stable. We view this as an architectural trade-off: GAIA's integral-transform structure may prioritize discretization invariance over peak in-distribution accuracy on shared-mesh forward tasks. For forward problems where deployment-time mesh resolution is fixed and matches training, Transolver remains a good choice; for applications where resolution varies—which includes many multi-query settings—GAIA's robustness profile is preferable.

Table 2: Comparison on forward benchmarks. Median relative L^2 error (%). GAIA is competitive with baselines.

Dataset	GINO	CORAL	Transolver	GAOT	GAIA
Poisson-Gauss	1.21	4.36	1.22	1.37	**0.77**
Elasticity	1.93	1.98	1.00	**0.99**	1.39
3D Darcy	20.51	21.58	**0.79**	36.27	1.24

4.3 Discretization Invariance

A central claim of integral-transform operators like IAE-Net is *discretization invariance* (DI): accuracy must be maintained when the model is evaluated at point-cloud resolutions different from training. We test this on Elasticity—the forward benchmark where Transolver and GAOT achieve better results than GAIA—by training each model at the full resolution and evaluating on coarser subsets obtained via Farthest Point Sampling. Figure 5 shows that GAIA maintains accuracy across all tested resolutions, while Transolver and GAOT (Appendix D.2, Figure 16) degrade significantly away from the training resolution. The in-distribution Transolver-vs-GAIA gap on Elasticity reported in Table 2 is an artifact of evaluating at the training resolution;

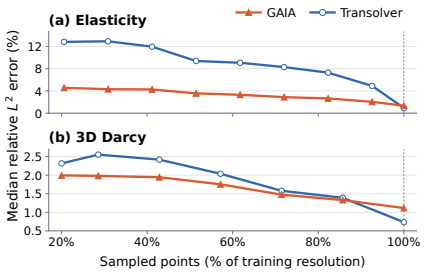


Figure 5: Discretization invariance on Elasticity and 3D Darcy. GAIA maintains accuracy across resolutions while Transolver degrades.

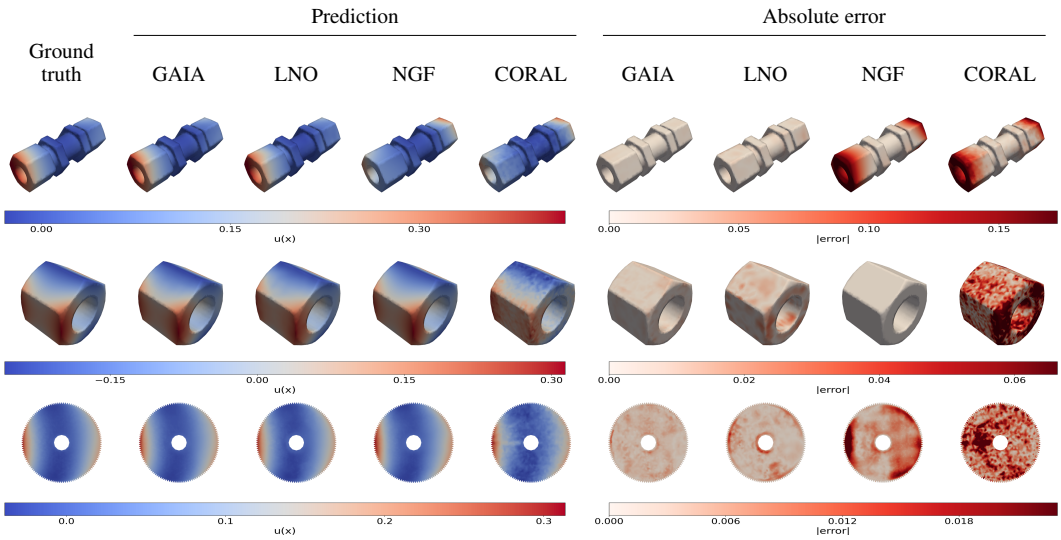


Figure 4: Results from the Mechanical Components Benchmark on the fitting, nut and gear categories. As shown in the error plots, GAIA reconstructs the solution with great fidelity across all shapes.

under resolution mismatch, the comparison reverses. GAIA similarly outperforms Transolver at non-training resolutions on 3D Darcy.

4.4 Efficiency analysis

Table 3 reports inference cost on EIT ($N = 2000$ mesh nodes) on a single A6000 in FP32. Latency is the mean wall-clock time over 100 runs (batch size 1); throughput is measured at batch size 16. GAIA achieves the highest throughput among geometry-adaptive methods (365 samples/s) at competitive latency (16 ms) and modest peak memory (48 MB). The closest method on throughput is CORAL at 253 samples/s. Among graph-based methods, GINO incurs roughly $2\times$ GAIA's latency at $40\times$ the memory due to its dense FNO latent grid. We additionally compare GAIA inference with classical FEM on the Mechanical Components Benchmark, highlighting the computational savings from avoiding per-instance mesh generation and numerical solves (Tab. 10, Appendix D.5).

Model	**Params (M)**	**Latency (ms)**	**Memory (MB)**	**Throughput (samples/s)**
GAIA	3.02	15.78	47.60	364.70
CORAL	0.84	4.99	32.70	252.9
GINO	2.31	36.54	1886.50	27.3
GAOT	3.40	10.45	45.4	135.9

Table 3: Computational cost comparison on EIT. GAIA has competitive latency and memory while maintaining the highest throughput among geometry-adaptive methods

We evaluate scalability on the 3D Darcy benchmark (Table 4), observing approximately linear growth in runtime and memory usage. GAIA supports million-point inference on a single GPU, while training-step memory limits the resolution to approximately 400K points. A separately trained 100K-point model achieves 2.6% median relative L^2 error under a reduced training budget. The single-GPU is the shared bottleneck documented in [59], which addresses it via multi-GPU parallelism; the same standard techniques (gradient checkpointing, chunked query evaluation, parallelism) can extend GAIA's range. Full results and comparisons with Transolver are provided in Appendix D.6.

Table 4: Scalability on 3D Darcy (single A6000 GPU). Training measurements are per-step; inference uses batch size 1.

	Training		Inference	
Points	ms	MB	ms	MB
7K	57.8	930	14.18	102
100K	262.2	12,220	91.57	1,165
400K	1082.0	48,726	397.04	4,595
1M	OOM	OOM	1002.37	11,449

4.5 Noise Robustness

We evaluate GAIA's robustness to test-time measurement noise by training once on clean data and evaluating at noise levels $\eta \in \{0\%, 1\%, 2\%, 5\%, 10\%\}$ using an additive Gaussian noise model scaled by the per-sample L^∞ norm. Figure 6 shows graceful degradation across all three inverse benchmarks: at $\eta = 10\%$, error grows by roughly $13\times$ on Airfoil, $4\times$ on EIT, and $7\times$ on RTE relative to the clean baseline, with no sharp transitions or instability. The graceful-degradation behavior is consistent with what we would expect from an integral-transform architecture: noise in the input is averaged across the integration kernel rather than being amplified by point-wise operations.

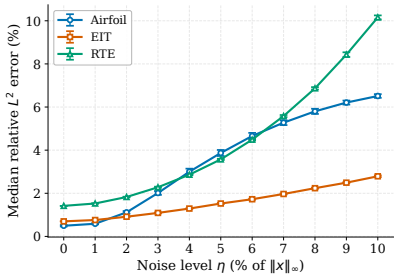


Figure 6: Graceful degradation in median relative L^2 error (%) when increasing noise on inverse benchmarks.

4.6 Ablations

To evaluate the improvement over the IAE-Net backbone, we compare GAIA against IAE-Net at matched parameter count on Elasticity. GAIA achieves 1.39% median relative L2 error, compared to 3.79% for IAE-Net. Table 5 reports two ablations on Elasticity (forward) and Airfoil (inverse). *Conditioning mechanism:* cross-attention strictly outperforms both concatenation and FiLM, with the largest gap on Airfoil (0.58% vs. 5.91% for concatenation, an order of magnitude). This confirms that spatially adaptive conditioning—each query point attending independently to the geometry tokens—is essential for capturing local geometric features; a single global summary vector is insufficient on tasks with complex spatially varying physics. *Tokenizer pathways:* removing either the boundary tokens or the slice tokens degrades performance on both benchmarks. The two pathways encode complementary information: boundary tokens summarize global domain shape, while slice tokens capture spatially varying input physics. We further investigate architectural choices, including token refinement, boundary encoding, skip connections, positional encoding, and the observation-to-domain decoder, as well as hyperparameter sensitivity to token count and dimension (Tabs. 17-19, Appendix E). We find that no ablated variant consistently outperforms the full model and sensitivity to token hyperparameters is mild across the tested ranges.

(a) Conditioning

Method	Elasticity	Airfoil
Concatenation	1.77	5.91
FiLM	1.88	0.88
Cross-attention	**1.39**	**0.58**

(b) Tokenizer pathways

Variant	Elasticity	Airfoil
No boundary	2.28	1.06
No slice	1.90	1.03
Full model	**1.39**	**0.58**

Table 5: Ablations. (a) Geometry conditioning: Cross-attention shows the best performance among conditioning methods. (b) Tokenizer pathways: Both tokenizers contribute to GAIA's performance. Median relative L^2 error (%).

5 Conclusion and Limitations

GAIA conditions integral kernels on tokenized representations of domain geometry to solve forward and inverse problems on arbitrary domains in a single pass. It achieves state-of-the-art on all inverse and BVP benchmarks while maintaining stable accuracy across resolutions. Three limitations bound the present work. First, GAIA is designed for time-independent operators; extending the framework to time-dependent problems requires additional architectural treatment for temporal evolution and is left to future work. Second, on forward problems where input and output share a fixed dense discretization, GAIA trades a small amount of in-distribution accuracy for substantially improved discretization invariance. Thirdly, the geometry-conditioning cross-attention scales as $\mathcal{O}(NK)$ in point-cloud size N and token count K; for very large 3D point clouds ($N > 10^6$), latent compression or sparse attention [60] may be needed to manage memory.

Code and data availability.

The code and newly introduced datasets are available at github.com/mkrishnan9/GAIA and Google Drive, respectively.

Funding

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility, operated under Contract No. DE-AC02-05CH11231 using NERSC award ALCC-ERCAP0034775. MK and RD were partially supported by ONR Award N00014-23-1-2086. H.Y. was partially supported by the US National Science Foundation under awards IIS2520978, GEO/RISE-5239902, and the DARPA D24AP00325. Approved for public release; distribution is unlimited.

References

[1] Roger Temam. *Navier–Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Society, 2024.

[2] KR Arun, Meenakshi Krishnan, and Saurav Samantaray. A unified asymptotic preserving and well-balanced scheme for the euler system with multiscale relaxation. *Computers & Fluids*, 233:105248, 2022.

[3] Liliana Borcea. Electrical impedance tomography. *Inverse problems*, 18(6):R99–R136, 2002.

[4] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[5] Sören Dittmer, Tobias Kluth, Peter Maass, and Daniel Otero Baguer. Regularization by architecture: A deep prior approach for inverse problems. *Journal of Mathematical Imaging and Vision*, 62(3):456–470, 2020.

[6] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[7] Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[8] Shizheng Wen, Arsh Kumbhat, Levi Lingsch, Sepehr Mousavi, Yizhou Zhao, Praveen Chandrashekar, and Siddhartha Mishra. Geometry aware operator transformer as an efficient and accurate neural surrogate for PDEs on arbitrary domains. *arXiv preprint arXiv:2505.18781*, 2025.

[9] Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3D PDEs. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023.

[10] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

[11] Yong Zheng Ong, Zuowei Shen, and Haizhao Yang. Integral autoencoder network for discretization-invariant learning. *Journal of Machine Learning Research*, 23(286):1–45, 2022.

[12] Louis Serrano, Lise Le Boudec, Armand Kassaï Koupaï, Thomas X Wang, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Operator learning with neural fields: Tackling PDEs on general geometries. *Advances in Neural Information Processing Systems*, 36:70581–70611, 2023.

[13] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[14] Bing Yu et al. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[15] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[16] Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

[17] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43 (5):A3055–A3081, 2021.

[18] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366: 415–447, 2018.

[19] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

[20] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

[21] Ke Chen, Meenakshi Krishnan, and Haizhao Yang. Error analysis for learning the time-stepping operator of evolutionary PDEs. *arXiv preprint arXiv:2509.04256*, 2025.

[22] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[23] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized Fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.

[24] Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022.

[25] Jean Kossaifi, Nikola Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-grid tensorized Fourier neural operator for high-resolution PDEs. *arXiv preprint arXiv:2310.00120*, 2023.

[26] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[27] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

[28] Tapas Tripura and Souvik Chakraborty. Wavelet neural operator: a neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*, 2022.

[29] Tailin Wu, Takashi Maruyama, and Jure Leskovec. Learning to accelerate partial differential equations via latent global evolution. *Advances in Neural Information Processing Systems*, 35: 2240–2253, 2022.

[30] Tailin Wu, Willie Neiswanger, Hongtao Zheng, Stefano Ermon, and Jure Leskovec. Uncertainty quantification for forward and inverse problems of PDEs via latent global evolution. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 320–328, 2024.

[31] Lianghao Cao, Thomas O'Leary-Roseberry, and Omar Ghattas. Derivative-informed neural operator acceleration of geometric MCMC for infinite-dimensional Bayesian inverse problems. *Journal of Machine Learning Research*, 26(78):1–68, 2025.

[32] Sebastian Kaltenbach, Paris Perdikaris, and Phaedon-Stelios Koutsourelakis. Semi-supervised invertible neural operators for Bayesian inverse problems. *Computational Mechanics*, 72(3): 451–470, 2023.

[33] Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José M Hernández-Lobato, Richard E Turner, and Emile Mathieu. On conditional diffusion models for PDE simulations. *Advances in Neural Information Processing Systems*, 37:23246–23300, 2024.

[34] Roberto Molinaro, Yunan Yang, Björn Engquist, and Siddhartha Mishra. Neural inverse operators for solving PDE inverse problems. *arXiv preprint arXiv:2301.11167*, 2023.

[35] Da Long, Zhitong Xu, Qiwei Yuan, Yin Yang, and Shandian Zhe. Invertible Fourier neural operators for tackling both forward and inverse problems. *arXiv preprint arXiv:2402.11722*, 2024.

[36] Andrew M Stuart. Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559, 2010.

[37] Jari P Kaipio and Erkki Somersalo. *Statistical and computational inverse problems*. Springer, 2005.

[38] Nicholas H Nelsen and Yunan Yang. Operator learning meets inverse problems: A probabilistic perspective. *arXiv preprint arXiv:2508.20207*, 2025.

[39] Lorenzo Baldassari, Ali Siahkoohi, Josselin Garnier, Knut Solna, and Maarten V de Hoop. Conditional score-based diffusion models for bayesian inference in infinite dimensions. *Advances in Neural Information Processing Systems*, 36:24262–24290, 2023.

[40] Jiachen Yao, Abbas Mammadov, Julius Berner, Gavin Kerrigan, Jong Chul Ye, Kamyar Azizzadenesheli, and Animashree Anandkumar. Guided diffusion sampling on function spaces with applications to pdes. *Advances in Neural Information Processing Systems*, 38:127057–127094, 2026.

[41] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

[42] Sepehr Mousavi, Shizheng Wen, Levi Lingsch, Maximilian Herde, Bogdan Raonić, and Siddhartha Mishra. RIGNO: A graph-based framework for robust and accurate operator learning for PDEs on arbitrary domains. *arXiv preprint arXiv:2501.19205*, 2025.

[43] Andrey Bryutkin, Jiahao Huang, Zhongying Deng, Guang Yang, Carola-Bibiane Schönlieb, and Angelica Aviles-Rivero. Hamlet: Graph transformer neural operator for partial differential equations. *arXiv preprint arXiv:2402.03541*, 2024.

[44] Zhihao Li, Haoze Song, Di Xiao, Zhilu Lai, and Wei Wang. Harnessing scale and physics: A multi-graph neural operator framework for pdes on arbitrary geometries. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 729–740, 2025.

[45] Yaozhong Shi, Zachary E Ross, Domniki Asimaki, and Kamyar Azizzadenesheli. Mesh-informed neural operator: A transformer generative approach. *arXiv preprint arXiv:2506.16656*, 2025.

[46] Giovanni Catalani, Siddhant Agarwal, Xavier Bertrand, Frederic Tost, Michael Bauerheim, and Joseph Morlier. Neural fields for rapid aircraft aerodynamics simulations. *Scientific Reports*, 14 (1):25496, 2024.

[47] Pan Du, Meet Hemant Parikh, Xiantao Fan, Xin-Yang Liu, and Jian-Xun Wang. Conditional neural field latent diffusion model for generating spatiotemporal turbulence. *Nature Communications*, 15(1):10416, 2024.

[48] Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse PDE problems. *Advances in Neural Information Processing Systems*, 37:33085–33107, 2024.

[49] Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: A general neural operator transformer for operator learning. In *International conference on machine learning*, pages 12556–12569. PMLR, 2023.

[50] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

[51] Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. *arXiv preprint arXiv:2310.12487*, 2023.

[52] Seungwoo Yoo, Kyeongmin Yeo, Jisung Hwang, and Minhyuk Sung. Neural Green's functions. *arXiv preprint arXiv:2511.01924*, 2025.

[53] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[54] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[55] Alberto P Calderón. On an inverse boundary value problem. *Computational & Applied Mathematics*, 25:133–138, 2006.

[56] Yuwei Fan and Lexing Ying. Solving optical tomography with deep learning. *arXiv preprint arXiv:1910.04756*, 2019.

[57] Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In *European conference on computer vision*, pages 175–191. Springer, 2020.

[58] Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer Science & Business Media, 2012.

[59] Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

[60] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[61] William RB Lionheart. EIT reconstruction algorithms: pitfalls, challenges and recent developments. *Physiological measurement*, 25(1):125, 2004.

[62] Guillaume Bal, Alexandre Jollivet, and Vincent Jugnon. Inverse transport theory of photoacoustics. *Inverse Problems*, 26(2):025011, 2010.

[63] Gerald Farin. *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014.

[64] Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. Fast tetrahedral meshing in the wild. 2020.

[65] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Datasets and Benchmarks

Below is a brief summary of the datasets that we use in the paper.

Table 6: Summary of benchmark problems. All datasets use median relative L^2 error (%) as the evaluation metric.

Benchmark	Governing Equations & Operator	Remarks
Poisson-Gauss [42]	$-\Delta u = f$, $x \in (0,1)^2$, $u\|_{\partial\Omega} = 0$. Operator: $\mathcal{G}_{pg} : L^2(\Omega) \to H_0^1(\Omega) \cap H^2(\Omega)$, $f \mapsto u$. Source f is a superposition of M random Gaussians.	**Forward.** Fixed unit-square domain. 128×128 uniform grid. 2048 train / 256 test samples.
Elasticity [9]	$\rho\,\partial_{tt}u - \nabla\cdot\boldsymbol{\sigma} = \mathbf{0}$. Operator: $\mathcal{G}_{el} : C(\Omega) \ni d \mapsto \boldsymbol{\sigma} \in L^2(\Omega;\mathbb{R}^{2\times2})$, where $d(x) = \mathrm{dist}(x,\Gamma_{\mathrm{inner}})$. Incompressible Rivlin–Saunders constitutive model.	**Forward.** Unit square with randomly shaped interior hole ($r \in [0.2, 0.4]$). 972 unstructured nodes. 1,024 train / 256 test samples.
3D Darcy Flow	$-\nabla\cdot(\kappa(x)\nabla u(x)) = 1$, $u\|_{\partial\Omega} = 0$. Operator: $\mathcal{G}_{df} : L^\infty(\Omega) \to H_0^1(\Omega)$, $\kappa \mapsto u$. Log-normal permeability field; $\kappa = \exp(\gamma\,\mathcal{S}/\sigma_{\mathcal{S}})$.	**Forward.** Star-shaped 3D domains (spherical harmonic boundary, $l_{\max} = 4$). Point cloud resolution 7000. 4096 train / 2048 test samples.
3D Poisson BVP [52]	$-\Delta u = f_{\mathrm{analytical}}$, $u\|_{\partial\Omega} = g(x,y,z)$. Oscillatory analytical source; randomized polynomial Dirichlet BCs.	**BVP / Forward.** 3D mechanical parts (MCB): gears, nuts, fittings, screws & bolts. Setting adapted from [52]. 200 shapes/category train, 20 test with 50 boundary conditions per shape.
EIT [34]	$\nabla\cdot(a(x)\nabla u) = 0$, mixed BCs. Operator: $\Lambda_a : H^{1/2}(\partial\Omega) \to H^{-1/2}(\partial\Omega)$, $u\|_{\partial\Omega} \mapsto a\nabla u\cdot n$. Inverse map: $\Lambda_a \mapsto a(x)$.	**Inverse.** Extends [34] to 2D star-shaped domains ($r(\theta) = r_0[1 + c_1\cos4\theta + c_2\cos8\theta]$). $L = 20$ Dirichlet patterns; $M = 272$ boundary sensors. Log-trigonometric conductivity.
Optical Tomography [56]	$v\cdot\nabla_x u + \sigma_t u = \sigma_s\int\Phi(v\cdot v')u\,dv'$. Albedo operator: $\Lambda : L^1(\partial\Omega) \to L^1(\partial\Omega)$, $\varphi \mapsto J_+$. Inverse map: Albedo difference data $\Delta\Lambda \mapsto \delta\sigma_s(x)$ scattering perturbation.	**Inverse.** Extends [56] to convex pentagon domains. S_{32} Discrete Ordinates solver; HG phase function ($g = 0.9$). 32 sources $\times$ 32 receivers.
Airfoil [41]	Compressible Euler equations around a 2D airfoil ($M_\infty = 0.8$, AoA $= 0^\circ$). Inverse map: sparse noisy Mach observations $\{\tilde{u}(x_j)\}_{j\in\mathcal{O}} \mapsto u(x)$ over the full domain.	**Inverse.** Varying NACA airfoil geometries. 221×51 C-grid. 1% relative Gaussian noise, 10% sparse observations. 1,000 train / 100 test.

A.1 Inverse Problem: Electrical Impedance Tomography

Electrical Impedance Tomography (EIT) is an imaging technique that involves solving an inverse problem to reconstruct the internal material distribution of an object using boundary measurements. It is governed by the electrostatic equations with variable conductivity given by the Laplace equation with mixed Dirichlet and Neumann boundary conditions:

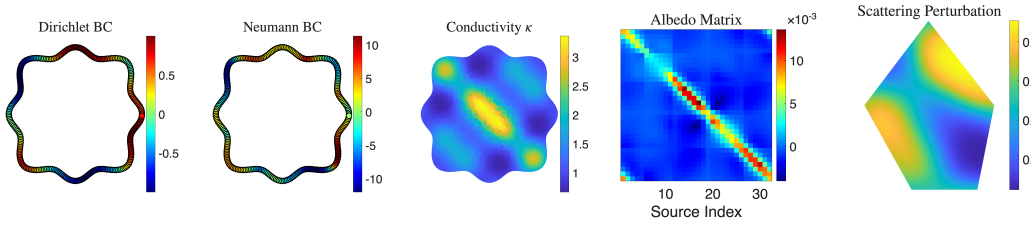
$$\nabla\cdot(a(x)\nabla u(x)) = 0, \text{ for } x \in \Omega \subseteq \mathbb{R}^d,\ d = 2, 3, . \tag{13}$$

Here, $a(x)$ is the conductivity distribution of the medium. The goal is to recover this distribution using only the boundary data on $\partial\Omega$ using the Dirichlet-to-Neumann map (DtN map, also known as the Calderon operator [55]) Λ_a given by:

$$\Lambda_a : H^{1/2}(\partial\Omega) \ni u_{|\partial\Omega} \mapsto a\nabla u_{|\partial\Omega}\cdot n \in H^{-1/2}(\partial\Omega).$$

where $H^{-1/2}(\partial\Omega)$ is the space of bounded linear functionals on the Sobolev space $H^{1/2}(\partial\Omega)$. In the case of full boundary measurement, it is a known result [55] that a is uniquely determined under suitable assumptions. However, in practice, we only have a finite amount of noisy data, making the EIT problem highly ill-posed and sensitive to noise [61].

We aim to directly learn the inverse map using boundary measurements across different geometries. The training and test datasets were generated by solving equation 13 on strictly star-shaped domains Ω defined by the polar radius $r(\theta) = r_0[1 + c_1\cos(4\theta) + c_2\cos(8\theta)]$. Extending the data generation process in [34], the isotropic conductivity coefficient $a(x, y)$ was drawn from a log-trigonometric



(a) EIT: Dirichlet-Neumann boundary measurements. (b) OT: Albedo matrix and scattering.

Figure 7: Inverse problems examples. (a) The EIT problem aims to reconstruct the internal conductivity distribution from boundary Dirichlet and Neumann values. (b) In OT, we reconstruct the scattering coefficient from the intensity at receivers given different source positions.

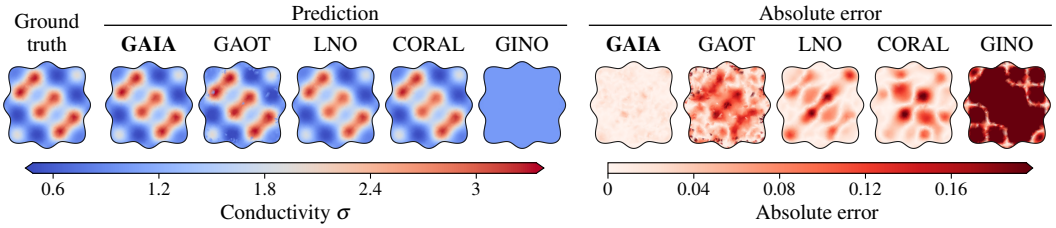


Figure 8: Model predictions on EIT conductivity reconstruction.

probability distribution to ensure strict positivity and smoothness. Specifically, $a(x, y)$ is constructed as:

$$a(x, y) = \exp\left(\sum_{k=1}^{m} c_k \sin(k\pi x)\sin(k\pi y)\right),$$

where m is drawn uniformly from $\{1, \ldots, 5\}$, and the coefficients $\{c_k\}_{k=1}^{m}$ are sampled independently from a uniform distribution $\mathcal{U}([-1, 1])$. For each sampled conductivity, the governing equation was solved numerically using quadratic Finite Element Method (FEM) under $L = 20$ distinct Dirichlet boundary conditions. The boundary data $\{g_\ell\}_{\ell=1}^{L}$ corresponds to plane waves incident from equispaced angles $\theta_\ell = \frac{2\pi(\ell-1)}{L}$, formulated as:

$$u(x, y)\big|_{\partial\Omega} = g_\ell(x, y) = \cos\left(2\pi(x\cos\theta_\ell + y\sin\theta_\ell)\right).$$

The corresponding Neumann measurement, $\Psi_\ell = a\frac{\partial u}{\partial \nu}$, representing the current flux, was evaluated at $M = 272$ boundary sensors for each excitation ℓ. Figure 7 shows the Dirichlet and Neumann measurements on the boundary for a given star-shaped domain and its corresponding recovered internal conductivity distribution.

A.2 Inverse Problem: Optical Tomography

In optical tomography, the transport of photons through a scattering and absorbing medium occupying a bounded domain $\Omega \subset \mathbb{R}^d$ is modeled by the stationary radiative transport equation (RTE). Let $u(x, v)$ denote the particle density at spatial location $x \in \Omega$ and velocity direction $v \in S^{d-1}$. The medium is characterized by its scattering coefficient $\sigma_s(x) \geq 0$ and absorption coefficient $\sigma_a(x) \geq 0$. The RTE is given by:

$$v \cdot \nabla_x u(x, v) + \sigma_t(x)u(x, v) = \sigma_s(x)\int_{S^{d-1}} \Phi(v \cdot v')u(x, v')dv', \quad x \in \Omega, \tag{14}$$

$$u(x, v) = \phi(x, v), \quad (x, v) \in \Gamma_-. \tag{15}$$

Here, $\sigma_t(x) = \sigma_a(x) + \sigma_s(x)$ is the total attenuation coefficient, and $\Phi(v \cdot v')$ is the scattering phase function, normalized such that $\int_{S^{d-1}} \Phi(v \cdot v')dv' = 1$. The boundary phase space is partitioned into inflow (Γ_-) and outflow (Γ_+) boundaries, defined as

$$\Gamma_\pm = \{(x, v) \in \partial\Omega \times S^{d-1} : \pm n_x \cdot v > 0\}, \tag{16}$$

where n_x is the unit outer normal vector at $x \in \partial\Omega$. In our experiment, following [56], both the boundary illumination and the recorded data are assumed to be independent of the velocity variable.

Specifically, the prescribed incoming particle density is modeled as an isotropic source depending only on the spatial variable, $\phi(x) \in L^1(\partial\Omega)$. Similarly, the measurements on the outflow boundary record the outgoing angular flux (or current), integrating the particle density over all outward-pointing velocities: $J_+(x) = \int_{v \cdot n_x > 0} (v \cdot n_x) u(x, v) dv$. The associated boundary observation operator, known as the Albedo operator Λ, thus maps the spatial input on the boundary to the spatial output:

$$\Lambda : L^1(\partial\Omega) \to L^1(\partial\Omega), \quad \Lambda : \phi(x) \mapsto J_+(x). \tag{17}$$

The continuous inverse problem seeks to recover the unknown medium properties, characterized by the scattering and absorption coefficients, from the knowledge of Λ. Formally, this defines the continuous inverse map $\mathcal{F}^{-1}$ from the space of bounded linear operators to the space of continuous medium parameters:

$$\mathcal{F}^{-1} : \mathcal{L}\left(L^1(\partial\Omega), L^1(\partial\Omega)\right) \to C(\Omega) \times C(\Omega), \quad \mathcal{F}^{-1}(\Lambda) = (\sigma_s, \sigma_a).$$

The well-posedness and Lipschitz stability of this inverse map have been established under suitable conditions [62].

For the numerical experiments and data generation in this work, we restrict our domain $\Omega \subset \mathbb{R}^2$ to randomly scaled convex pentagons. We assume the absorption coefficient is a known, spatially invariant constant, $\sigma_a(x) \equiv 0.01$, reducing the inverse problem to the recovery of the spatially varying scattering coefficient $\sigma_s(x)$. The highly forward-peaked anisotropic scattering in biological tissues is modeled using the 2D Henyey-Greenstein (HG) phase function,

$$\Phi(v \cdot v') = \frac{1 - g^2}{2\pi(1 + g^2 - 2g(v \cdot v'))},$$

where the anisotropy factor is set to $g = 0.9$. The spatial distribution of the scattering coefficient is constructed using a randomized superposition of low-frequency trigonometric modes passed through a scaled logistic sigmoid function. Let $x' = (x'_1, x'_2)$ denote the spatial coordinates normalized by the domain scale factor. The scattering field is defined as:

$$\sigma_s(x) = 1.0 + \left(1 + \exp\left(-3 \sum_{k=1}^{m} c_k \sin(\omega_{x,k} \pi x'_1) \sin(\omega_{y,k} \pi x'_2)\right)\right)^{-1},$$

where the number of modes m is drawn uniformly from $\{2, \ldots, 5\}$, the expansion coefficients c_k are sampled independently from a uniform distribution $\mathcal{U}([-1, 1])$, and the spatial frequencies $\omega_{x,k}$ and $\omega_{y,k}$ are drawn uniformly from the integer set $\{1, 2, 3\}$. This construction ensures that the scattering coefficients are smooth and strictly bounded within the physical range $\sigma_s(x) \in (1.0, 2.0)$.

The continuous boundary illumination is discretized using $N_{src} = 32$ localized isotropic point sources distributed uniformly along $\partial\Omega$. The forward transport problem is solved using the Discrete Ordinates (S_{32}) method with angular discretization of 32 angles, and a Finite Element spatial discretization, using GMRES for the source iteration. The corresponding outgoing current is measured at $N_{rec} = 32$ interleaved receiver locations. This discrete setup approximates the Albedo operator as a dense matrix $\Lambda \in \mathbb{R}^{32 \times 32}$. To isolate the scattering anomalies and suppress structural geometric artifacts, we employ a difference imaging approach. A baseline Albedo matrix Λ_0 is simulated using a known, constant background scattering coefficient $\sigma_{s,0} = 1.0$. The objective of the discrete inverse problem is then to infer the scattering perturbation $\delta\sigma_s(x) = \sigma_s(x) - \sigma_{s,0}$ strictly from the measured difference data $\Delta\Lambda = \Lambda - \Lambda_0$.

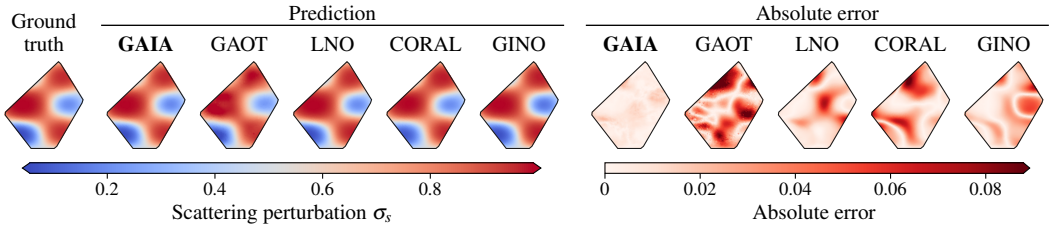


Figure 9: Model predictions on RTE. Comparison of ground truth solution against predictions from GAIA, GAOT, LNO, CORAL, and GINO.

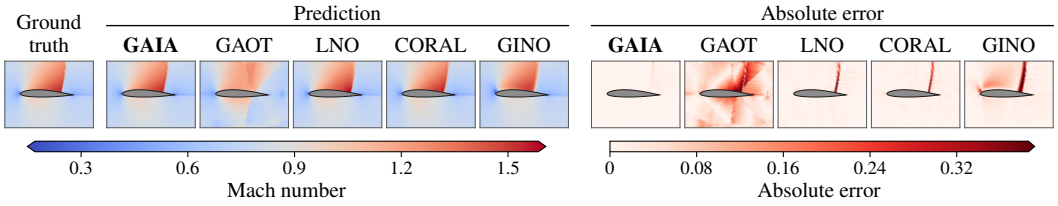


Figure 10: Model predictions on Airfoil Mach field reconstruction from sparse noisy observations.

A.3 Inverse Problem: Airfoil Flow Reconstruction

We consider the problem of reconstructing a full aerodynamic flow field from sparse, noisy measurements over a family of transonic solutions over airfoil geometries. The underlying physical system is governed by the compressible Euler equations,

$$\frac{\partial \rho_f}{\partial t} + \nabla \cdot (\rho_f v) = 0, \qquad \frac{\partial \rho_f v}{\partial t} + \nabla \cdot (\rho_f v \otimes v + pI) = 0, \qquad \frac{\partial E}{\partial t} + \nabla \cdot \big((E+p)v\big) = 0, \tag{18}$$

where ρ_f is the fluid density, v is the velocity vector, p is the pressure, and E is the total energy. Viscous effects are neglected. Far-field conditions are prescribed as $\rho_\infty = 1$, $p_\infty = 1.0$, $M_\infty = 0.8$, $AoA = 0^\circ$, with a no-penetration condition imposed at the airfoil surface. Here M_∞ is the Mach number and AoA is the angle of attack.

The dataset is drawn from the NACA airfoil corpus of [41], comprising $N = 1{,}200$ flow solutions, one per airfoil shape, generated with a second-order implicit finite volume solver on a body-fitted C-grid of $N_x \times N_y = 221 \times 51 = 11{,}271$ quadrilateral elements, refined near the airfoil surface. Airfoil geometries are parameterized via a design element approach [63] in which the baseline NACA-0012 profile is deformed by displacing the control nodes of an enclosing cubic design element in the vertical direction, with displacements drawn from $d \sim \mathcal{U}[-0.05, 0.05]$. The dataset is partitioned into $N_{\text{tr}} = 1{,}000$ training, $N_{\text{val}} = 100$ validation, and $N_{\text{te}} = 100$ test configurations.

We formulate it as a sparse-to-full field reconstruction problem. The quantity of interest is the steady-state Mach number field $u : \Omega \to \mathbb{R}$, discretized over all $11{,}271$ mesh nodes. At inference time, only a sparse subset of observations $\{u(x_j)\}_{j \in \mathcal{O}}$, with $\mathcal{O} \subset \{1, \ldots, N_x N_y\}$, is available as input, and these observations are further corrupted by zero-mean Gaussian noise scaled to the per-sample peak Mach number:

$$\tilde{u}_i(x_j) \;=\; u_i(x_j) \;+\; \eta_{i,j}, \qquad \eta_{i,j} \sim \mathcal{N}\big(0,\, \sigma^2 \|u_i\|_\infty^2\big), \quad j \in \mathcal{O}, \tag{19}$$

with $\sigma = 0.01$, corresponding to 1% of the per-sample L^∞ norm. The task is to recover the complete Mach field u at all grid nodes, including those with no sensor coverage. The model must simultaneously denoise, interpolate, and extrapolate from incomplete observations over geometrically varying domains.

A.4 3D Boundary Value Problem: Poisson on MCB

To test the generalization of our framework to unseen problem domains, we construct a PDE dataset using the Mechanical Components Benchmark (MCB) dataset [57], which contains a variety of 3D mechanical part shapes. We adapt the tetrahedral meshes for shapes in four categories (SCREWS & BOLTS, NUTS, FITTINGS, and GEARS) in [52] built by meshing the interiors of unit-cube-normalized shapes using fTetWild [64]. We used a similar setting as in [52] but formulated and generated data for a boundary value problem. The shape collection is divided into 200 shapes for training and 20 shapes for testing per category. Shapes within the same category exhibit diverse geometries, presenting a standard challenge for learned solution operators to generalize effectively.

We define a highly non-linear, oscillatory analytical source term $f_{\text{analytical}}(x, y, z)$. The base function is constructed using trigonometric variations along all three spatial axes:

$$f_{\text{analytical}} = T_1(x, y) + T_2(x, y) + T_3(z)$$

where the individual components are defined by the constants $A = 1.25$, $B = 1.5$, $C = 1.5$, and $D = 1.5$, such that:

$$T_1(x, y) = -\left((A\pi)^2 + (AC\pi)^2\right) \sin(A\pi x) \cos(AC\pi y)$$

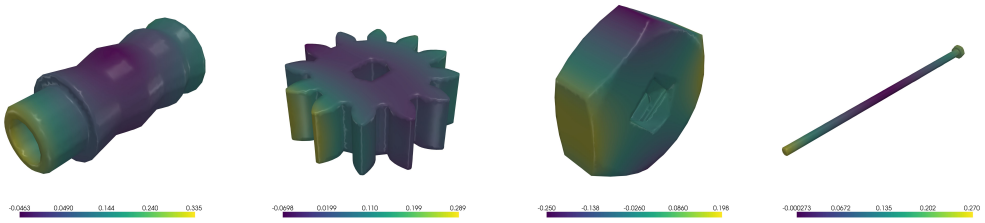


Figure 11: Boundary Values on MCB shapes

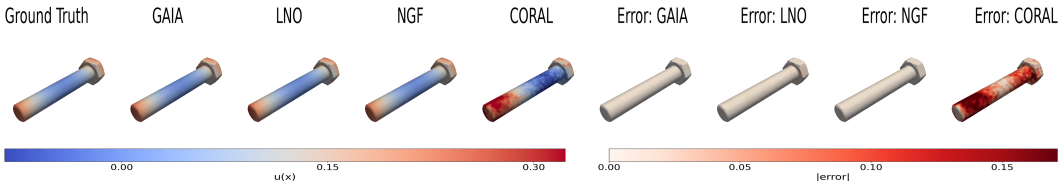


Figure 12: Results on the screws&bolts category from the Mechanical Components Benchmark.

$$T_2(x,y) = (A\pi)^2 \cos(A\pi x)\,(1-\sin(AB\pi y)) + (AB\pi)^2 \sin(AB\pi y)\,(1-\cos(A\pi x))$$
$$T_3(z) = 2(AD\pi)^2 \cos(2AD\pi z).$$

To match the magnitude of the source fields released by [52], we use the affinely rescaled source $f = \alpha_c f_{\text{analytical}} + \beta_c$, where the constants (α_c, β_c) for each category c are obtained by a least-squares fit of $f_{\text{analytical}}$ to the released source field on a reference shape of that category. These constants are held fixed for all shapes and samples within a category (FITTINGS: $(-0.8641, 15.8068)$; GEARS: $(-0.8463, 12.2468)$; NUTS: $(-0.7511, -1.0794)$; SCREWS & BOLTS: $(-1.7921, 121.6870)$).

To generate multiple unique samples per geometry, we randomize the Dirichlet boundary conditions $g(x, y, z)$ at the mesh surface. The boundary values are prescribed using a parameterized polynomial function:

$$g(x,y,z) = E(x^3 - 3xy^2) + F(y^3 - 3x^2y) + (x^2 - z^2)$$

For each generated sample, the coefficients E and F are sampled from uniform distributions such that $E \sim \mathcal{U}(-1.0, 1.0)$ and $F \sim \mathcal{U}(0.0, 1.0)$.

To generate the dataset, we employ the FEniCS finite element framework [58] to solve Poisson's equation on the generated meshes.

A.5 Forward Problem: 3D Darcy Flow

We consider the stationary Darcy flow through a heterogeneous porous medium. Let the spatial domain $\Omega \subset \mathbb{R}^3$ represent three-dimensional domain. The fluid pressure $u(x)$ is modeled by the elliptic partial differential equation:

$$-\nabla \cdot (\kappa(x)\nabla u(x)) = 1, \quad x \in \Omega,$$

subject to homogeneous Dirichlet boundary conditions, $u|_{\partial\Omega} = 0$. Here, $\kappa(x)$ represents the strictly positive permeability field of the medium. The operator learning objective is to approximate the continuous solution operator $\mathcal{G}_d$ that maps the permeability field, defined on a variable geometries, to the resulting scalar pressure field. Formally, for a given domain Ω, the mapping is defined as:

$$\mathcal{G}_{df} : L^\infty(\Omega) \to H_0^1(\Omega), \quad \mathcal{G}_{df}(\kappa) = u.$$

To learn this operator, we construct a dataset by solving the Darcy flow equation on $\Omega \subset \mathbb{R}^3$. The boundary $\partial\Omega$ of each sample is generated as a "star shape" defined by a random spherical harmonic expansion up to degree $l_{max} = 4$:

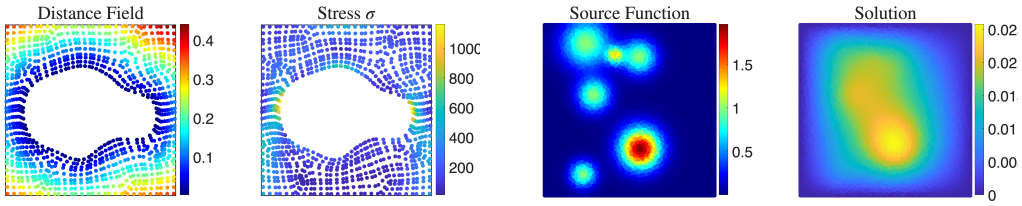
$$r(\theta,\phi) = \sum_{l=0}^{4} \sum_{m=-l}^{l} c_{lm} Y_l^m(\theta,\phi), \quad c_{lm} \sim \mathcal{N}(0, \sigma_l). \tag{20}$$

The resulting geometries are discretized into watertight unstructured tetrahedral meshes. The permeability field $\kappa(x)$ is computed as a continuous log-normal random field, $\kappa(x) = \exp(\gamma \cdot \mathcal{S}(x)/\sigma_{\mathcal{S}})$, where $\mathcal{S}(x)$ is a superposition of $M = 30$ random Fourier modes:

$$\mathcal{S}(x) = \sum_{j=1}^{M} \alpha_j \cos(k_j \cdot x + \phi_j). \tag{21}$$

Here, the wave vectors $k_j \sim \mathcal{N}(0, \sigma_k^2 I)$ control the spatial correlation length, phases $\phi_j \sim \mathcal{U}[0, 2\pi]$ ensure translation invariance, and amplitudes $\alpha_j \sim \mathcal{N}(0, 1)$ provide variance. The field is normalized by its sample standard deviation $\sigma_{\mathcal{S}}$ and scaled by $\gamma = 0.5$ to ensure a smooth, strictly positive, and spatially heterogeneous medium.

The forward problem is solved using the standard Galerkin Finite Element Method (FEM) implemented in FEniCS. We utilize linear Lagrange (P_1) basis functions for both the solution space and the coefficient field. Following the FEM solution, we extract the training data by randomly sampling $N_p = 7000$ points within the domain volume for each simulation.



(a) Elasticity (b) Poisson-Gauss

Figure 13: Input output fields

A.6 Forward Problem: Elasticity Problem

We evaluate the model on a standard Elasticity benchmark, where the objective is to learn the mapping from a domain's geometry to its resulting internal stress field under external loading. The general force balance of a solid body is governed by the hyperelastic equation:

$$\rho \partial_{tt} u - \nabla \cdot \boldsymbol{\sigma} = 0,$$

where ρ represents the mass density, u the displacement field, and σ the Cauchy stress tensor. The strain field is related to the displacement field via standard kinematic relations, and the system is closed by a hyperelastic constitutive model describing the non-linear stress-strain relationship. For this benchmark from [41], the stationary solution is considered for a unit square hyper-elastic incompressible Rivlin-Saunders specimen with a hole at its center. The geometry of the hole is randomly sampled such that the radius always takes a value between 0.2 and 0.4 (see [41] for details). The specimen is subjected to mixed boundary conditions: it is clamped at the bottom boundary ($u = 0$) and is under a constant vertical tension traction on its top boundary. The stress field is computed for different geometries with a finite elements solver with about 100 quadratic quadrilateral elements. The target function in this dataset is the stress field σ, which is available at 972 unstructured coordinates. The operator learning task thus seeks to approximate the continuous solution operator $\mathcal{G}_{el}$ that maps the geometry of the domain directly to the stress field. The geometry is encoded as the shortest-path distance function from the inner boundary of the hole, $d(x) = \text{dist}(x, \Gamma_{\text{inner}})$. The formal operator mapping is thus defined as:

$$\mathcal{G}_{el} : C(\Omega) \ni d \to \boldsymbol{\sigma} \in L^2(\Omega; \mathbb{R}^{2\times 2}).$$

Figure 13a demonstrates an example of a sample with a deformation in the center. The input field is distance of the unordered point cloud from the inner boundary.

A.7 Forward Problem: Poisson-Gauss

To demonstrate GAIA's performance on a standard fixed-grid benchmark, we consider the classical Poisson's equation [42]. Let the domain be the unit square $\Omega = (0, 1)^2$. The governing PDE is given by:

$$-\Delta u(x) = f(x), \; x \in (0, 1)^2, \tag{22}$$

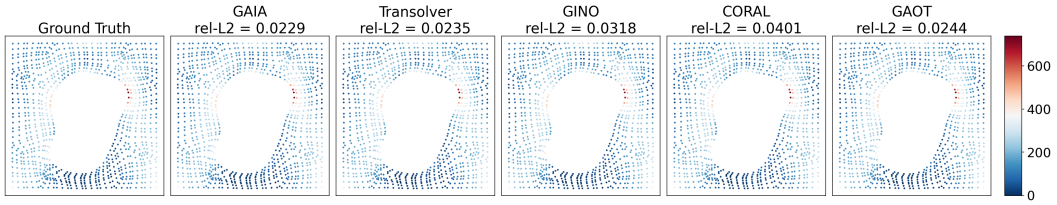


Figure 14: Model predictions on Elasticity. Comparison of ground truth solution against predictions from GAIA, Transolver, CORAL, GINO and GAOT.

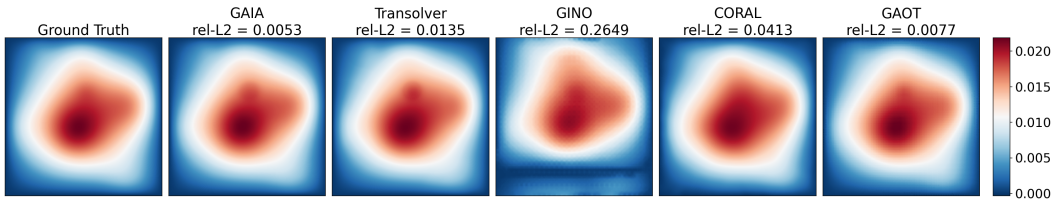


Figure 15: Model predictions on Poisson-Gauss. Comparison of ground truth solution against predictions from GAIA, Transolver, CORAL, GINO and GAOT.

subject to homogeneous Dirichlet boundary conditions, $u|_{\partial\Omega} = 0$. The objective of the neural operator is to learn the inverse Laplacian operator $\mathcal{G}_{pg} = -\Delta^{-1}$, which maps the right-hand side source function to the scalar solution field. Formally, this defines the continuous solution operator:

$$\mathcal{G}_{pg} : L^2(\Omega) \to H_0^1(\Omega) \cap H^2(\Omega), \quad \mathcal{G}_{pg}(f) = u. \tag{23}$$

The source term f is procedurally generated as a superposition of M random Gaussian pulses:

$$f(x, y) = \sum_{i=1}^{M} \exp\left(-\frac{(x - \mu_{x,i})^2 + (y - \mu_{y,i})^2}{2\sigma_i^2}\right), \tag{24}$$

where M is an integer drawn from a geometric distribution. The mean coordinates are sampled uniformly such that $\mu_{x,i}, \mu_{y,i} \sim \mathcal{U}(0, 1)$, and the standard deviations are sampled as $\sigma_i \sim \mathcal{U}(0.025, 0.1)$. The ground truth data pairs (f, u) are evaluated on a high-resolution 128×128 uniform grid. We refer to [42] for further dataset details. Figure 13b visualizes a representative input source function and its corresponding computed solution.

B GAIA Implementation Details

All models were trained on a single NVIDIA A6000 GPU using the Adam optimizer [65] with cosine decay learning rate scheduling and 5% warmup starting at 10% of the peak learning rate. We use 8 attention heads, Fourier positional encodings with 10 frequency bands and base frequency $\omega_0 = 30$. All hyperparameters for experiments are given in Table 7.

C Baselines

Unless specifically noted, all baseline implementations follow their default hyperparameters in their published repositories. While CORAL and LNO can technically handle varying input/output spatial domains, the codebase had to be modified slightly to accomplish this.

C.1 CORAL

CORAL [12] casts operator learning as a regression in the latent spaces of two implicit neural representations (INRs). Each input sample a_i and output sample u_i is encoded by fitting a shift-modulated SIREN via auto-decoding: a shared base network is trained jointly with per-sample latent codes z_{a_i}, z_{u_i}, where the codes are passed through small hypernetworks that produce additive shifts on the SIREN activations. The codes are obtained through a short inner loop of gradient descent on the reconstruction loss. Once the two INRs are trained, a separate network learns the mapping $z_{a_i} \mapsto z_{u_i}$ in latent space. Because the INRs are queried point-wise, CORAL natively supports varying input and output meshes across samples.

Table 7: Training and architectural hyperparameters across benchmarks. BS refers to batch size, ep.: epochs, $N_{\text{in}}/N_{\text{out}}$: number of input/output points per sample, GHD: Hidden dimension, GAD: Token dimension x Number of tokens, AD: attention dimension, NGT: Number of geometric tokens.

	Training							Architecture			Geometry Conditioning			
Dataset	N_{train}	N_{test}	N_{in}	N_{out}	BS	LR	Ep.	Modes	Width	Blocks	GHD	GAD	AD	NGT.
OT	8000	2000	32×32	$\sim$5000	4	2e-4	500	96	128	4	128	64	128	8
EIT	4096	2048	272	2000	4	1e-4		32		5		32		8
Airfoil	1000	100	1127	11271	8	5e-4		96		4		256		4
Elasticity	1024	256	972	972								128	64	16
P-Gauss	2048	256	128^2	128^2				128	64	5				8
3D Darcy	4096	2048	7000	7000	4	2e-4			128		256	256	128	8
MCB: Gear	10k	1k	$\sim$6700	$\sim$15800	1	2e-4	100	128	128	4	128	128	256	4
MCB: Screw						1e-4								
MCB: Fitting						1e-4								
MCB: Nut						2e-4		96			256	256		8

We use the official implementation[2]. Both INRs are 4-layer SIRENs of hidden width 256 with shift-only modulation generated by a single-layer hypernetwork of width 128. The latent regression network $z_a \mapsto z_u$ is a 3-block residual MLP with Swish activations, trained with Adam (no weight decay) and a step decay of $\gamma = 0.9$. The two-stage protocol trains the INRs first, then freezes them and fits the regression network. We meta-learn the inner learning rate at a meta-rate of 10^{-4}. As the official codebase assumes a shared mesh between input and output, we modified the data pipeline to query the two INRs on distinct point clouds for boundary-value problems and inverse problems where sensor and reconstruction grids differ.

C.2 GINO

GINO [9] is a hybrid neural operator that combines a graph neural operator (GNO) [26] with a Fourier neural operator (FNO) to handle inputs and outputs on irregular point clouds. An input GNO encoder integrates the input function over local neighborhoods of each latent grid point, lifting the irregular sample onto a regular Cartesian latent grid. An FNO processor operates on this latent grid, exploiting the FFT for global integration. An output GNO decoder then projects the processed latent field back onto an arbitrary query point cloud. The encoder and decoder rely on radius graphs: each output node attends to all input nodes within a fixed radius r.

We use the official implementation[3]. The FNO processor uses 6 layers with hidden width 32, group normalization, and a channel MLP expansion of 0.5; AdaIN conditioning is enabled with 8 feature channels. Both GNO encoder and decoder use linear kernel transforms with a half-cosine kernel weighting. The input GNO MLP has hidden layers $[128, 256, 128]$ and the output GNO MLP has hidden layers $[256, 512, 256]$. The GNO radius r is tuned to the characteristic length scale of each domain and is shared between the input and output GNO; values are listed in Table 8.

Table 8: GINO GNO radius r across benchmarks. The same radius is used for the input and output GNO.

Dataset	EIT	Airfoil	Poisson-Gauss	Elasticity	RTE	3D Darcy
r	0.25	0.125	0.125	0.125	5.0	0.25

C.3 Transolver

Transolver [10] is a transformer-based neural operator that adapts standard self-attention to PDE solving on irregular meshes. Rather than computing attention directly between mesh nodes – which scales quadratically with the point-cloud size – it introduces a Physics-Attention mechanism that (softly) assigns each mesh point to a small set of learnable physical states (or *slices*), runs self-attention over the slice representations, and then projects back to the original mesh. This reduces attention cost from $\mathcal{O}(N^2)$ to $\mathcal{O}(NK)$ for K slices and lets the model scale to large unstructured meshes. Because both the slice assignment and the back-projection share the same point set, Transolver requires the

[2] `https://github.com/LouisSerrano/coral`

[3] `https://github.com/neuraloperator/neuraloperator`

input and output to live on a common discretization, which is why we omit it from inverse and BVP comparisons.

We use the official implementation[4]. The model uses 8 Transolver blocks of hidden width 256 with 8 attention heads, an MLP expansion ratio of 2, and 32 slice tokens per block; the slice projection uses 8 reference points. Training uses AdamW with a learning rate of 10^{-3}, weight decay 10^{-5}, and batch size 20.

C.4 Geometry Aware Operator Transformer (GAOT)

GAOT [8] is a graph-based transformer architecture for operator learning on unstructured grids. It uses a multiscale graph encoder (MAGNO) to aggregate point-cloud features onto a fixed Cartesian latent token grid, applies transformer attention over the latent tokens, and decodes back to the mesh through a corresponding graph decoder.

For the forward problems (Elasticity, Poisson-Gauss and 3D Darcy), we adopt the default hyperparameters provided in the GAOT repository without modification [5]. For 2D problems, the latent token grid is 64×64; for the 3D Poisson problem, we use a 64×32×32 latent grid as specified in the original paper [8]. Training uses the default AdamW optimizer with a mixed learning rate schedule (linear warmup followed by cosine decay).

For the three inverse problems (EIT, airfoil flow and OT), we use a modified version of the GAOT codebase. Since inverse problems involve distinct input and output domains (e.g., boundary measurements to interior fields), we modify the MAGNO graph construction to build separate encoder and decoder neighborhood graphs, rather than a single shared graph as in the original implementation. We additionally lower the initial learning rate to $1e^{-4}$, as the default $8e^{-4}$ did not yield good performance on this problem. All other architectural hyperparameters remain at their defaults.

C.5 Latent Neural Operator (LNO)

LNO [48] learns operator mappings via attention through a low-dimensional latent space. The encoder applies cross-attention between a fixed set of learnable latent queries and the input point cloud, producing a compact latent representation. A stack of self-attention blocks then processes these latent tokens. Finally, the decoder applies cross-attention from the output query points to the processed latent tokens to produce the predicted field.

We apply LNO to the four forward Poisson problems on MCB dataset, using the Plasticity config from the original repository[6] as the reference. The architectural hyperparameters are kept identical. Batch size is reduced to 1 to accommodate the MCB meshes in memory, and the model is trained for 100 epochs like our models. For the inverse problems, we use the LNO propagator config from the original repository as the reference, with batch size set to 64 for speed.

C.6 Neural Greens Function (NGF)

NGF [52] approximates the solution operator of linear symmetric PDEs by learning a low-rank decomposition of the Green's function $G(x, y)$ on a point cloud. Two coordinate networks predict left- and right-vector bases at each point, and the solution at a query is recovered by integrating the learned Green's function against the source field. The architecture is designed for forward problems whose governing operator is linear, self-adjoint, and admits a Green's function representation. It is not applicable to inverse problems or nonlinear forward problems. So, we apply NGF to four Poisson problems on irregular geometries (fitting domain, gear, nut, screws/bolts) using the default network and training configuration (trained with 3 data-parallel processes, matching the configuration of the released checkpoints) from the original repository[7] without modification.

[4] https://github.com/thuml/Transolver

[5] https://github.com/camlab-ethz/GAOT

[6] https://github.com/L-I-M-I-T/LatentNeuralOperator

[7] https://github.com/KAIST-Visual-AI-Group/NGF.git

D Additional Results

D.1 Noise Robustness

We adopt a relative-Gaussian noise model scaled by the per-sample L^∞ norm of the measurement field. Let x denote a clean measurement vector; the corrupted measurement is:

$$\tilde{x} = x + \varepsilon, \qquad \varepsilon \sim \mathcal{N}\Big(0,\ \big(\eta \,\|x\|_\infty\big)^2 I\Big)\,. \tag{25}$$

for noise levels $\eta \in \{0\%, 1\%, 2\%, 5\%, 10\%\}$

The three benchmarks differ in what the measurement vector x is and at what granularity the L^∞ norm is computed.

- **Airfoil.** The measurement is the partial Mach field observed at a fraction of mesh points. A single $\|x\|_\infty$ is computed per sample over all observed points.
- **EIT.** The measurement consists of $L = 20$ paired Dirichlet–Neumann boundary patterns evaluated at $M = 272$ sensors.
- **RTE.** The measurement is the difference Albedo matrix $\Delta\Lambda \in \mathbb{R}^{N_{rec} \times N_{src}}$. Because receivers have very different outgoing flux magnitudes, we use a per-receiver-row L^∞ scale: $\varepsilon_{r,s} \sim \mathcal{N}(0, (\eta \,\|\Delta\Lambda_{r,:}\|_\infty)^2)$.

For each benchmark and noise level, we evaluate the trained model under 10 independent noise realizations and report the mean and standard deviation of the median relative L^2 error across these seeds; results are summarized in Table 9. Across all three benchmarks, GAIA degrades gracefully under measurement noise. Seed-to-seed variation remains low relative to the mean at every noise level, showing that this robustness is a stable property of the trained model.

Table 9: Median relative L^2 error (%) under increasing noise levels η on the Airfoil, EIT, and RTE inverse benchmarks. Values are reported as mean $\pm$ std over 10 independent noise realizations.

	Noise level η				
Benchmark	0%	1%	2%	5%	10%
Airfoil	0.49 ± 0.00	0.58 ± 0.03	1.12 ± 0.06	3.87 ± 0.14	6.51 ± 0.09
EIT	0.70 ± 0.00	0.76 ± 0.01	0.91 ± 0.01	1.53 ± 0.03	2.79 ± 0.04
RTE	1.42 ± 0.00	1.53 ± 0.01	1.83 ± 0.01	3.57 ± 0.03	10.16 ± 0.09

D.2 Discretization invariance

We conduct a discretization invariance study for Elasticity and 3D Darcy benchmarks (see Figure 16). While Transolver's degradation is less severe than for the Elasticity benchmark, GAIA still outperforms Transolver at non-training resolutions. GAOT's performance also suffers under varying resolutions. We used GAOT's official implementation unmodified with their given Elasticity configuration file. All models were evaluated at the same subsampled coordinates used for GAIA and Transolver, with no per-resolution retraining or tuning for any method.

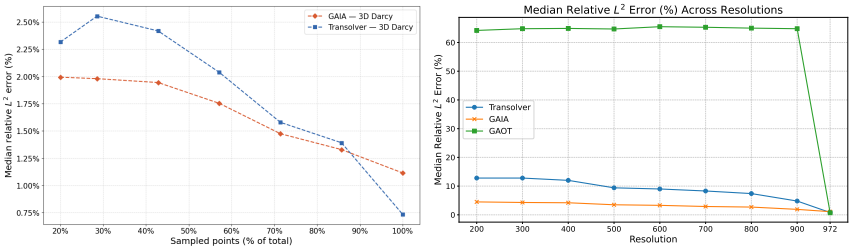


Figure 16: Discretization invariance on 3D Darcy (trained on 7000 points) and on Elasticity (trained at 972 points). Although Transolver and GAOT achieve lower error at the training resolution (100%), GAIA degrades more gradually under subsampling and achieves lower error at coarser resolutions.

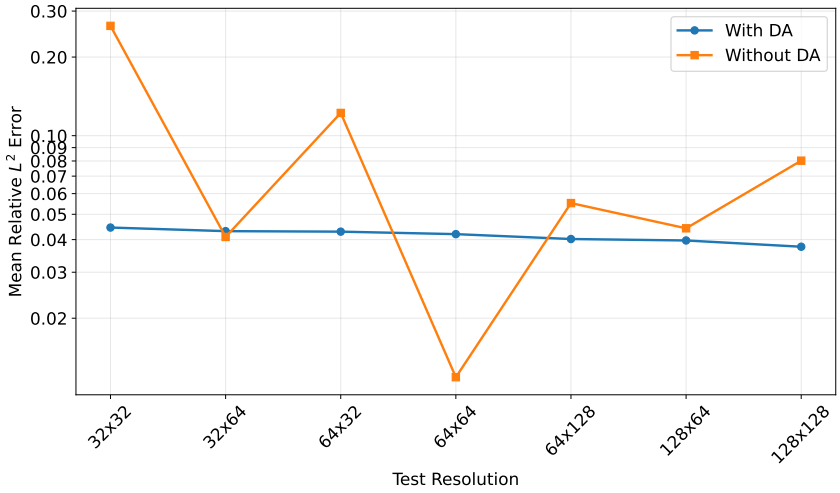


Figure 17: Discretization invariance on Poisson-Gauss. Multi-resolution augmentation (blue) provides one-shot generalization across resolutions compared to single-resolution training (orange).

D.3 Data augmentation for discretization invariance

Discretization invariance can be further improved by training on multiple resolutions simultaneously. We demonstrate this on the Poisson-Gauss benchmark, comparing models trained with and without multi-resolution data augmentation. The data augmentation used in IAE-Net is implemented here via a stride-based multi-resolution training scheme. The original data is generated on a 128×128 grid and subsampled to 64×64 for standard training. For augmented training, the 64×64 data is upsampled back to the canonical 128×128 resolution using bicubic interpolation. Multi-resolution views are then generated by striding this grid. Each training iteration simultaneously optimizes over multiple resolutions $\mathcal{R} = \{32 \times 32, 32 \times 64, 64 \times 32, 64 \times 64, 64 \times 128, 128 \times 64, 128 \times 128\}$, with the total loss:

$$\mathcal{L}_{\text{total}} = \sum_{T \in \mathcal{R}} \lambda_T \cdot \mathcal{L}(\hat{u}_T, u_T), \tag{26}$$

where λ_T are per-resolution weights (set to 1.0 in this experiment). As shown in Figure 17, multi-resolution training yields stable performance across all tested resolutions. We note that for unstructured point clouds, generating equivalent multi-resolution training views would require scattered data interpolation to obtain field values at new point locations, which is less straightforward than grid striding and may introduce interpolation artifacts.

D.4 Efficiency analysis

Table 3 reports computational costs on the EIT benchmark ($N = 2000$ mesh nodes) measured on a single NVIDIA A6000 GPU in FP32. Latency is the mean wall-clock time over 100 runs (batch size 1) after 20 warm-up iterations with CUDA synchronization; throughput is measured at batch size 16. GAIA achieves competitive latency and memory while maintaining the highest throughput among geometry-adaptive methods.

To assess the computational efficiency of GAIA relative to baseline methods, we conduct a comprehensive timing analysis on the EIT benchmark with $N = 2000$ mesh nodes. All experiments are performed on a single NVIDIA A6000 GPU using FP32 precision.

Metrics. We evaluate the following computational metrics:

- **Parameters:** Total number of learnable weights in the model, reported in millions (M).
- **Latency:** Wall-clock time (ms) for a single inference with batch size 1. We report the mean over 100 timed runs following 20 warm-up iterations to ensure stable GPU clock speeds and memory allocation. CUDA synchronization is enforced before each timing measurement to account for asynchronous kernel execution.
- **Peak Memory:** Maximum GPU memory allocated during a forward pass with batch size 1, measured after resetting CUDA memory statistics.

- **Throughput:** Maximum number of samples processed per second with batch size 16, measured by timing batched inference over 50 iterations.

D.5 Comparison with Finite Element Methods

We compare the computational cost of GAIA with conventional finite element methods (FEM) on the 3D Mechanical Components Benchmark (MCB). While FEM provides higher accuracy, GAIA targets amortized multi-query settings, such as engineering design, optimization, and inverse problems, where solutions are required across many geometries or boundary conditions. In these settings, mesh generation and numerical solves can contribute substantially to the computational cost. GAIA operates directly on point clouds without requiring volumetric mesh generation at inference.

Table 10: Computational cost comparison on MCB. Mesh generation times are reported for fTetWild [52], while FEM solve times are measured using FEniCS. GAIA inference uses point-cloud inputs and does not require mesh generation. Times are reported in seconds.

Category	Mesh gen.	FEM solve	FEM total	GAIA
Fitting	18.04	0.72	18.76	0.021
Gear	45.80	0.71	46.51	0.014
Nut	11.60	1.14	12.74	0.025
Screws/bolts	12.58	0.55	13.13	0.019

FEM solve times were measured using FEniCS 2019.1.0 (dolfin, PETSc 3.22.2) with the default solver on an AMD EPYC 9475F CPU. GAIA inference times represent batch-size-one latency on a single NVIDIA A6000 GPU.

As shown in Table 10, mesh generation accounts for most of the conventional FEM pipeline's computational cost on these geometries. By avoiding this step and performing single-pass inference, GAIA substantially reduces per-instance computational cost. This advantage is particularly relevant when evaluating many previously unseen geometries, where new meshes would otherwise be required.

D.6 Scalability on Large 3D Point Clouds

We evaluate the scalability of GAIA on the 3D Darcy benchmark as the number of input points increases. All experiments are performed on a single NVIDIA A6000 GPU (48 GB) in FP32.

Training scalability. Table 11 compares the training-step latency and peak GPU memory usage of GAIA and Transolver. Each training step includes the forward pass, backward pass, and optimizer update. Both methods exhibit approximately linear scaling in point-cloud size, with memory requirements approaching the GPU capacity at 400K points.

Table 11: Training scalability on 3D Darcy. Latency is reported per training step, including forward and backward passes and optimizer updates. Memory is reported in MB.

	GAIA		Transolver	
Points	Time (ms)	Mem. (MB)	Time (ms)	Mem. (MB)
7K	57.8	930	30.0	887
50K	137.3	6,140	116.0	5,832
100K	262.2	12,220	220.2	11,555
250K	681.9	30,444	533.6	28,590
400K	1082.0	48,726	848.6	45,681
450K	OOM	OOM	OOM	OOM

To assess predictive accuracy at higher resolutions, we additionally train GAIA on 100K-point clouds using 1024 training samples, 256 test samples, and 100 epochs, without additional hyperparameter tuning. The resulting model achieves approximately 2.6% median relative L^2 error.

Inference scalability. Table 12 reports the inference latency and GPU memory consumption of GAIA at increasing point-cloud resolutions.

Table 12: GAIA inference scalability on 3D Darcy. Batch-size-one inference on a single NVIDIA A6000 GPU in FP32.

Points	Latency (ms)	Memory (MB)
7K	14.18	102
50K	42.59	593
100K	91.57	1,165
250K	247.00	2,879
400K	397.04	4,595
500K	498.32	5,733
1M	1002.37	11,449

Inference requires substantially less memory than training, as gradients and optimizer states are not stored. GAIA supports inference on point clouds containing up to one million points on a single GPU, requiring approximately 1 second and 11.4 GB of memory.

D.7 Training Randomness

To assess sensitivity to training randomness (weight initialization, data shuffling), we train GAIA and all baselines with different random seeds. Table 13 reports the resulting statistics. The small standard deviation indicates stable performance across runs.

Dataset	GINO	CORAL	LNO	GAOT	GAIA
EIT	26.42 ± 2.63	1.65 ± 0.03	0.91 ± 0.04	8.02 ± 0.38	$\mathbf{0.71 \pm 0.04}$
OT	1.90 ± 0.15	3.06 ± 0.16	1.76 ± 0.06	5.79 ± 0.64	$\mathbf{1.45 \pm 0.04}$
Airfoil	4.79 ± 0.69	1.59 ± 0.06	2.49 ± 0.30	9.53 ± 0.34	$\mathbf{0.58 \pm 0.03}$

Table 13: Sensitivity to training randomness on inverse benchmarks. Mean $\pm$ standard deviation of the test-set median relative L^2 error (%) across 5 independent training runs.

Dataset	GINO	CORAL	Transolver	GAOT	GAIA
Poisson-Gauss	1.210 ± 0.045	4.360 ± 0.110	1.220 ± 0.310	1.372 ± 0.194	$\mathbf{0.770 \pm 0.040}$
Elasticity	1.930 ± 0.040	1.980 ± 0.048	1.000 ± 0.130	$\mathbf{0.994 \pm 0.004}$	1.393 ± 0.053
3D Darcy	20.510 ± 0.670	21.580 ± 0.220	$\mathbf{0.789 \pm 0.049}$	36.266 ± 2.020	1.240 ± 0.170

Table 14: Sensitivity to training randomness on forward benchmarks. Mean $\pm$ standard deviation of the test-set median relative L^2 error (%) across 5 independent training runs.

Shape	CORAL	NGF	LNO	GAIA
Fitting	18.81 ± 0.22	11.25 ± 0.59	14.88 ± 0.58	$\mathbf{10.30 \pm 0.49}$
Gear	13.81 ± 0.65	5.33 ± 0.51	5.91 ± 0.20	$\mathbf{3.63 \pm 0.30}$
Nut	20.69 ± 1.82	5.34 ± 0.13	6.95 ± 0.36	$\mathbf{4.95 \pm 0.25}$
Screws/Bolts	20.47 ± 0.38	6.31 ± 0.30	5.22 ± 0.28	$\mathbf{3.45 \pm 0.36}$

Table 15: Sensitivity to training randomness on MCB. Mean $\pm$ standard deviation of the test-set median relative L^2 error (%) across 3 independent training runs.

D.8 Additional Baselines on MCB

We additionally evaluate GINO and GAOT on the 3D Poisson boundary value problem using the Mechanical Components Benchmark (MCB). Table 16 reports their performance alongside GAIA. Both methods are non-competitive in all four shape categories.

E Ablations

We examine the sensitivity of the model to the tokenization hyperparameters, and provide details on the choice of conditioning mechanism, and the contribution of each token pathway.

Table 16: Additional baseline comparisons on MCB. Median relative L^2 error (%).

Model	Fitting	Gear	Nut	Screws/Bolts
GINO	45.08	34.89	26.19	52.83
GAOT	52.96	47.72	51.98	73.62
GAIA	**10.30**	**3.63**	**4.95**	**3.45**

First, we conduct ablations on every design module across benchmarks. We also conduct studies on the EIT benchmark to investigate the sensitivity of GAIA to the token dimension and the total number of tokens of the tokenization module. We additionally ablate the geometry conditioning mechanism itself.

E.1 Architectural Component Ablations

We evaluate the contribution of individual architectural components across four benchmarks: Elasticity, EIT, Airfoil, and Optical Tomography (RTE). Each ablation modifies one component of GAIA while keeping the remaining architecture and training protocol unchanged.

Table 17: Architectural component ablations across four benchmarks. Median relative L^2 error (%). Each variant modifies one component of the full model. Elasticity has no boundary→interior transfer, so O2D does not apply.

Variant	Elasticity	EIT	Airfoil	RTE
GAIA (full)	**1.39**	**0.71**	**0.58**	1.45
Refined tokens → static tokens	1.52	0.73	0.92	1.49
PointNet → linear pooling	1.49	0.85	0.82	**1.32**
Dense skips → ResNet skips	1.36	0.96	0.75	1.50
Dense skips → no skips	1.83	1.41	4.13	3.07
Fourier features → raw coordinates	1.91	1.03	1.72	1.43
O2D decoder → direct boundary map	–	1.15	0.73	**1.31**

Table 17 shows that the full model is the best on almost every benchmark; no reduced variant is uniformly competitive. Skip connections contribute significantly, removing them consistently degrades performance. Static tokens cost more on diverse geometry families (Elasticity, Airfoil, RTE), indicating token refinement matters. Direct boundary-to-interior (removing the O2D latent set) is often less accurate.

Intuitively, each module addresses a specific property of the operator being learned. Cross-attention kernel conditioning supports the integral kernel solution operator's spatially adaptive dependence on the domain. Dual-pathway tokenizer provides disentangled physics and geometry representations. Fourier positional features help capture high-frequency spatial dependence, which coordinate MLPs represent poorly under spectral bias. Dense skips preserve multi-scale features across the block stack, since the encode-process-decode bottleneck in each block may discard fine detail that later blocks need. Token refinement between blocks lets the conditioning signal evolve with the features. The O2D decoder transfers sparse boundary measurements to the interior through a fixed set of K latent modes, reducing the cost of a direct $O(MN)$ boundary-to-interior integral transform to $O(K(M+N))$, where M and N denote the numbers of boundary and interior points, respectively. The boundary encoder is permutation-invariant and cardinality-agnostic so that the shape representation does not depend on the ordering and resolution of boundary samples.

E.2 Token Dimension

Table 18 reports the median relative L^2 error as the token dimension varies from 4 to 64, with the number of tokens fixed at 8. Performance is best at 4 and degrades gradually as the dimension increases, suggesting that a compact token representation is sufficient to capture the relevant geometric information for this benchmark. Larger token dimensions may introduce unnecessary capacity that hinders optimization without providing additional representational benefit.

Token Dim	4	8	16	32	64
L2 Median	**0.00698**	0.00712	0.00737	0.00804	0.00832

Table 18: Effect of token dimension with number of tokens fixed to 8.

E.3 Number of Geometric Tokens

Table 19 examines the effect of varying the token count from 4 to 64 with the token dimension fixed at 8. Performance improves sharply from 4 to 8 tokens, after which it plateaus. This indicates that for simple shapes, modest number of tokens is sufficient to encode the geometric structure of the domain, and that additional tokens do not meaningfully improve accuracy.

Num Tokens	4	8	16	64
L2 Median	0.00851	**0.00712**	0.00727	0.00760

Table 19: Effect of number of tokens with token dimension fixed to 8.

E.4 Number of Slice Tokens

The slice tokenizer aggregates the input sensor data into a fixed set of physics-informed tokens via softmax-weighted clustering. Table 20 examines the effect of varying the slice-token count from 32 to 96, with all other hyperparameters held fixed. Performance is best at 64 slice tokens and degrades only mildly outside this setting, indicating that a moderate number of slices is sufficient to summarize the boundary measurements without introducing redundancy.

Num Slice Tokens	32	48	64	96
L2 Median	0.00797	0.00873	**0.00712**	0.00735

Table 20: Effect of the number of slice tokens on the EIT benchmark, with all other hyperparameters fixed.

E.5 Number of Blocks

Table 21 examines the effect of varying the number of blocks from 3 to 7 on the EIT benchmark, with all other hyperparameters held fixed. Performance is best at 5 blocks and remains close at neighbouring depths, indicating that the model is robust to this choice.

E.6 Geometry Conditioning Ablation Details

Table 5 compares three alternatives for using geometry information to condition the integral kernel on a forward problem (Elasticity) and an inverse problem (Airfoil). In *concatenation*, the geometry tokens are mean-pooled into a single vector, projected, and concatenated onto the per-point feature before entering the kernel MLP, thus providing global context but no spatial selectivity. In *FiLM* conditioning, the mean-pooled token vector is projected to produce per-channel scale and shift parameters that affinely modulate the combined feature representation. The proposed cross-attention mechanism instead lets each spatial query point attend independently to the full set of geometry tokens via multi-head attention. Cross-attention consistently achieves the lowest error on both benchmarks, implying that spatially adaptive conditioning is important for capturing local geometric features.

E.7 Tokenizer Ablation Details

The IAE refinement stage stacks several adaptive blocks, each applying integral transforms followed by token-updated cross-attention. Table 5 evaluates the contribution of each token pathway by removing either the boundary feature tokens (produced by the PointNet-style encoder from X_{bnd}) or the slice tokens (produced by the soft-clustering encoder from the interior mesh and field values). Both ablations increase error relative to the full model on the Elasticity and Airfoil benchmarks, implying that both tokenization pathways contribute information: boundary tokens capture the global domain shape, while slice tokens encode the spatially varying physics of the input field.

Num Blocks	3	4	5	6	7
L2 Median	0.00762	0.00789	**0.00712**	0.00739	0.008406

Table 21: Effect of number of blocks with all other hyperparameters fixed.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state our contributions—a geometry-conditioned integral autoencoder for unified forward, inverse, and BVP operator learning, and four new benchmarks for varying-geometry inverse and BVP problems—and the experiments in Section 4 demonstrate GAIA's performance in terms of accuracy, discretization invariance, and efficiency across seven PDE benchmarks.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Section 5, covering the steady-state restriction, in-distribution accuracy trade-off on dense forward problems, and cross-attention scaling at very large point cloud sizes.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper contains no formal theorems.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The architecture is fully specified in Section 3; per-benchmark hyperparameters and training protocol are in Appendix B; data generation procedures for all seven benchmarks are in Appendix A.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have released the GAIA codebase, and the four new benchmark datasets (Airfoil sparse-to-full reconstruction, 3D Poisson BVP on Mechanical Components, 3D Darcy on star-shaped domains, and Optical Tomography on convex pentagons). Datasets will be released under CC-BY 4.0 and code under the MIT license; full documentation of data generation is provided in Appendix A.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Optimizer, learning rate schedules, batch sizes, training epochs, and data splits are reported in Appendix B for GAIA and in Appendix C for all baselines.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report multi-seed runs on all benchmarks in Appendix D.7, and additionally evaluate noise robustness across multiple noise realizations on Airfoil, EIT, and RTE in Appendix D.1.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments were run on a single NVIDIA A6000 GPU (48GB) using FP32 precision. Training details are provided in Section B.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?

   Answer: [Yes]

   Justification: The work conforms to the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This is foundational research on operator learning for partial differential equations. Positive impacts include accelerating scientific simulation in domains such as medical imaging (EIT, optical tomography), aerodynamics, and engineering design. We are not aware of direct negative societal impacts; the methods accelerate existing scientific computation rather than enabling qualitatively new misuse pathways.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No high-risk models or datasets are released. The contributed benchmark datasets are synthetic PDE solutions and pose no misuse risk.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Baseline implementations (Transolver, GAOT, GINO, CORAL, LNO, NGF) are obtained from their official open-source repositories, with links provided in their respective subsections of the appendix. Existing benchmark datasets are obtained and used according to their respective licenses.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The four new benchmark datasets and the GAIA codebase will be released upon acceptance, accompanied by data-generation scripts, solver configurations, and dataset documentation as described in Appendix A. Datasets are released under the Creative Commons Attribution 4.0 International license (CC-BY 4.0); code is released under the MIT license.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard component.