INTERPOLATING NEURAL OPERATOR (INO): A DATA-FREE AND EFFICIENT APPROACH FOR LEARNING PDE SOLUTION OPERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural operators have become a popular approach to approximate the solution operators of parametric partial differential equations (PDEs). However, existing neural operators either require a large amount of simulation data or a long physics-informed training on GPUs, and they cannot tell how accurate an individual prediction is. In this paper, we propose the Interpolating Neural Operator (INO), a data-free interpolating neural network that is trained directly on the weak form of the PDE. In INO, the Karhunen–Loève coordinates of the input field are treated as additional inputs together with the spatial coordinates, and each input is approximated by a C-HiDeNN sub-network whose trainable parameters are nodal values. Since the network is multilinear in its parameters, training reduces to a sequence of one-dimensional linear solves by greedy alternating least squares. As a result, INO trains on one CPU core and predicts a new solution in microseconds. For coercive problems, the total error of every prediction is bounded by a computable residual bound that requires no reference solution, and the same bound applies to the predictions of other methods that satisfy the boundary conditions exactly. Before each prediction, INO checks whether the leading coordinates of the input lie within the range on which it is trained, and inputs outside this range can be passed to a conventional solver or to an INO trained on a wider range. INO is compared with physics-informed FNO and DeepONet on different benchmarks. INO is the most accurate model on most of these problems, by $15\times$ on two-dimensional Helmholtz at 65^2 and $53\times$ on the diffusion–reaction benchmark, and on the one- and two-dimensional problems its training on one CPU core takes 3–$80\times$ less time than the physics-informed baselines on one GPU.

1 INTRODUCTION

Many engineering tasks, such as design optimization, inverse identification and uncertainty quantification, require solving a PDE for many different input functions. Conventional numerical solvers such as the finite element method have to run a new simulation for every input, which becomes expensive when thousands of evaluations are needed. Data-driven neural operators such as DeepONet and the Fourier neural operator (FNO) (Lu et al., 2021; Li et al., 2021) learn the solution operator from pairs of input and solution fields. However, every training sample requires a full numerical simulation. Physics-informed variants (Raissi et al., 2019; Wang et al., 2021; Li et al., 2024) remove the need for data by minimizing the PDE residual, but they typically require long training time for fine resolution and 3D problems with stochastic optimization, and their errors vary across random initializations (Appendix D.4). None of these methods can tell how accurate a particular prediction is.

In this paper, we propose the Interpolating Neural Operator (INO), a data-free interpolating neural network for the solution operators of parametric PDEs. As shown in Fig. 1, the input field is first represented by the coordinates of a truncated Karhunen–Loève (KL) expansion. These coordinates are treated as additional inputs of the network together with the spatial coordinates, so that a single network over space and parameters represents the solution operator. Each input is fed into a convolution hierarchical deep-learning neural network (C-HiDeNN) sub-network (Lu et al., 2023; Park et al., 2025), whose connectivity and activation functions are fixed by a one-dimensional mesh

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

and whose trainable parameters are nodal values. The sub-networks of all inputs are multiplied to form a rank-one term, called a mode, and M modes are summed. The network is multilinear in its parameters, so each training step is a linear solve of one-dimensional size, and training reduces to greedy alternating least squares, which requires no data and no stochastic optimizer. The separated form is compact, and inference is a contraction of one-dimensional factors (Section 4.3). Since the C-HiDeNN interpolant has the Kronecker delta property, INO satisfies the boundary conditions exactly, and its predictions can be substituted back into the PDE to bound the total error.

The main contributions of this paper are summarized as follows:

- An efficient data-free operator, demonstrated on a moving heat source, diffusion and Helmholtz problems up to 513^2 and 129^3, a nonlinear diffusion problem and a space–time diffusion–reaction benchmark. On the one- and two-dimensional problems, its training by linear solves takes less time than physics-informed training, and its error depends little on the random initialization (Sections 4.1 and 4.3).
- A computable error bound for every prediction (Section 3.4). The bound uses the true input field rather than its truncated expansion, so it covers the total error, needs no reference solution, and holds outside the parametric domain. It applies to any prediction that satisfies the boundary conditions, including those of a physics-informed FNO (Section 4.2).
- A domain check that decides from the input alone whether INO makes a prediction, so that inputs outside the parametric domain can be passed to a conventional solver or to an INO trained on a wider parametric domain (Sections 3.3 and 4.1).
- A controlled comparison with physics-informed FNO and DeepONet on identical test inputs against reference solutions for the true input field, with each baseline trained with its original settings (Section 4.1).
- An analysis of the error of INO, which shows that its spatial modes are accurate and that the error comes from how they are weighted for a given input; computing these weights by a small linear solve reduces the error by one to two orders of magnitude on the linear problems (Section 4.3).

2 Related work

Operator-learning methods such as DeepONet (Lu et al., 2021), FNO (Li et al., 2021) and grid- or graph-based encoders (Gao et al., 2021; Pfaff et al., 2021; Huang et al., 2023) learn the solution operator from pairs of input and solution fields generated by a conventional numerical solver. Physics-informed variants replace the data by the PDE residual, which is evaluated at collocation points in the physics-informed DeepONet (PI-DeepONet) (Raissi et al., 2019; Wang et al., 2021) or on the output grid in the physics-informed FNO (PI-FNO) (Li et al., 2024). These methods are data-free like INO, but their training is a nonconvex optimization with known failure modes (Wang et al., 2022; Krishnapriyan et al., 2021; Grossmann et al., 2024; McGreivy & Hakim, 2024). Separable PINNs (Cho et al., 2023) also factorize the network by dimension, but they still rely on a collocation loss and a stochastic optimizer.

The canonical polyadic (CP) decomposition approximates a multivariate function by a sum of products of univariate functions. It is commonly computed by alternating least squares (Carroll & Chang, 1970; Kolda & Bader, 2009) or by greedy rank-one enrichment (Zhang & Golub, 2001), and low-rank tensor methods have been widely used to solve PDEs (Bachmayr, 2023). In computational mechanics, proper generalized decomposition (PGD) treats parameters as extra coordinates (Ammar et al., 2006; Chinesta et al., 2011; Nouy, 2010), including stochastic PDEs with KL inputs (Nouy, 2007), and interpolating neural networks combine this idea with C-HiDeNN interpolation (Lu et al., 2023; Li et al., 2023; Guo et al., 2024; 2025; 2026; Park et al., 2025). INO builds on the separated form and the greedy training of these methods and on the C-HiDeNN factors of interpolating neural networks. Relative to these works, INO makes predictions for arbitrary input fields, which are projected onto their KL coordinates and checked against the parametric domain before any prediction, and it bounds the error of each prediction with the true input field, as discussed below. We also compare INO with physics-informed neural operators on meshes up to 129^3.

Stochastic collocation (Bungartz & Griebel, 2004) and stochastic Galerkin methods with polynomial chaos (Ghanem & Spanos, 1991; Cohen et al., 2010) are the classical methods for parametric

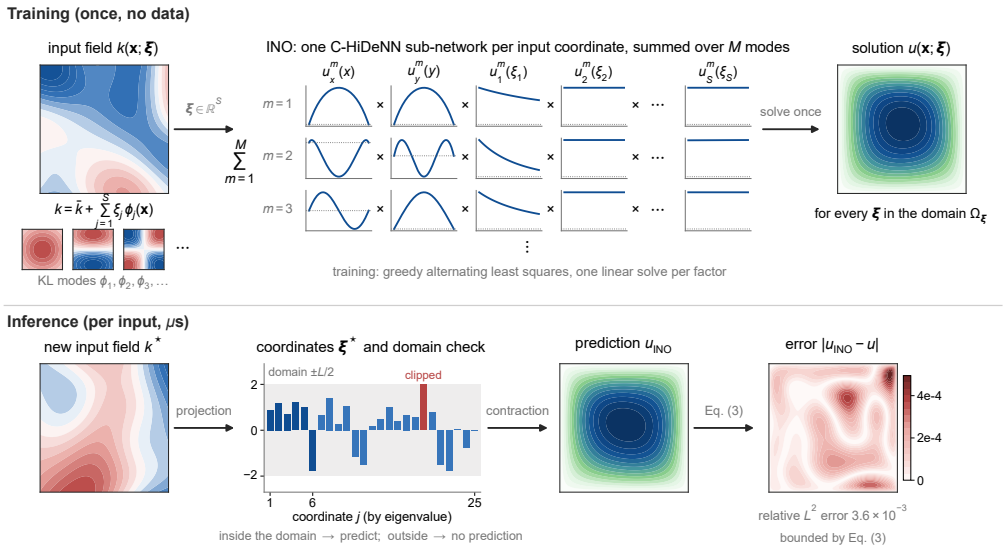


Figure 1: Overview of INO, shown on 2D diffusion problem. **Training**: the input field is represented by its S KL coordinates $\boldsymbol{\xi}$, each coordinate becomes an input of the network with its own C-HiDeNN sub-network (the curves are the one-dimensional factors of three modes of the trained operator of Table 1), and the weak form is solved once over the parametric domain without training data. **Inference**: a new input field $k^\star$ is projected onto its coordinates $\boldsymbol{\xi}^\star$; if its leading coordinates lie within the parametric domain (grey band), the prediction is obtained by contracting the factors, and its error is bounded by Eq. (3) evaluated for the true input field $k^\star$.

PDEs. Collocation solves the PDE at a set of parameter points and interpolates between them, one deterministic solve per point, and stochastic Galerkin methods solve one coupled system for all polynomial coefficients. Residual-based a posteriori error bounds have been developed for reduced-basis methods (Rozza et al., 2008; Binev et al., 2011), for PGD (Ladevèze & Chamoin, 2011) and for stochastic Galerkin methods (Eigel et al., 2014). The bound of Section 3.4 is of this type, but its residual is computed with the true input field instead of its truncated expansion. As a result, it bounds the total error of a prediction, including the error from truncating the input field, and remains valid outside the parametric domain. Since it needs only the prediction and the input field, it also applies to other models whose predictions satisfy the boundary conditions, which we use to bound the error of PI-FNO (Section 4.2).

3 INO FORMULATION

3.1 NETWORK ARCHITECTURE

The input field is first represented by the truncated KL expansion of a given covariance, which can be written as:

$$k(\boldsymbol{x};\boldsymbol{\xi}) \;=\; \bar{k}+\sum_{j=1}^{S}\phi_j(\boldsymbol{x})\,\xi_j, \qquad \xi_j\in[-L/2,\,L/2], \tag{1}$$

where $\bar{k}$ is the mean of the field; S is the number of retained KL modes; ϕ_j is the j-th KL mode scaled by the square root of its eigenvalue; ξ_j is the j-th KL coordinate (Ghanem & Spanos, 1991); and L is the width of the parametric domain in each direction, measured in standard deviations of the coordinates. As a result, a truncated input field is represented by a point $\boldsymbol{\xi}$ in the parametric domain $\Omega_{\boldsymbol{\xi}} = [-L/2, L/2]^S$. In INO, the S coordinates are treated as inputs together with the d spatial coordinates, so that the solution $u(\boldsymbol{x},\boldsymbol{\xi})$ becomes a function of $d+S$ variables. Since a grid in $d+S$ dimensions is not feasible, INO approximates the solution in separated form (Appendix A.2), which

can be written as:

$$u(\boldsymbol{x}, \boldsymbol{\xi}) \approx \sum_{m=1}^{M} \prod_{i=1}^{d} u_{x_i}^m(x_i) \prod_{j=1}^{S} u_j^m(\xi_j), \tag{2}$$

where M is the total number of modes; $u_{x_i}^m$ and u_j^m are the univariate functions of the m-th mode in the spatial direction x_i and in the parametric direction ξ_j; and the amplitude c_m of each mode (Appendix B) is absorbed into one of its factors. Each univariate function $u_j^m(\xi_j)$ is a C-HiDeNN sub-network (Guo et al., 2025) of the form $\widetilde{\boldsymbol{N}}(x; s, a, p)\,\boldsymbol{u}$, where $\boldsymbol{u}$ contains the nodal values of a one-dimensional mesh and $\widetilde{\boldsymbol{N}}$ is the row vector of C-HiDeNN basis functions, which is controlled by the patch size s, the dilation parameter a and the reproducing order p (Lu et al., 2023). The sub-network is a partially connected neural network whose hidden layers are the linear finite element basis functions of the mesh and a convolution patch of radial and polynomial neurons, and whose only trainable parameters are the nodal values (Appendix A). With n nodes per spatial direction and n_ξ nodes per parameter direction, INO has $M(d\,n + S\,n_\xi)$ parameters, a number that grows linearly with S, whereas a grid would require $n^d n_\xi^S$ unknowns. Since the basis functions satisfy the Kronecker delta property, Dirichlet boundary conditions are imposed exactly on the nodal values. Therefore, INO needs no boundary penalty term, whose weight has to be tuned in physics-informed methods (Wang et al., 2022). All integrals are computed using Gaussian quadrature.

This paper distinguishes three errors. The *rank error* is measured against a reference solution of the same discretization for the truncated field of Eq. (1). The *representation error* is the difference between this truncated field and the true input field. The *total error* is measured against a reference solution for the true input field itself, and it is the error reported against the baselines.

3.2 Data-free training

INO is trained on the weak form of the PDE integrated over the parametric domain: the nodal values of Eq. (2) are chosen such that $\int_{\Omega_{\boldsymbol{\xi}}} \mathcal{A}_k(v, u)\,\mathrm{d}\boldsymbol{\xi} = \int_{\Omega_{\boldsymbol{\xi}}} F(v)\,\mathrm{d}\boldsymbol{\xi}$ for every test function v of the same form, where $\mathcal{A}_k(\cdot,\cdot)$ is the bilinear form of the PDE for the input field $k = k(\cdot\,; \boldsymbol{\xi})$ and $F(v) = \int_\Omega f v$ is its linear form. Two properties make the training efficient. First, since k is affine in $\boldsymbol{\xi}$, the weak form of $-\nabla\cdot(k\nabla u) = f$ splits into exactly $d(S{+}1)$ separated terms, each of which is a product of one-dimensional matrices, so that no integral over the $(d + S)$-dimensional domain is needed (Appendix B). Second, Eq. (2) is multilinear in its parameters. If all factors except one are fixed, the weak form is linear in the nodal values of the remaining factor, which are therefore obtained from one linear system whose size equals the number of nodes of that factor's mesh. Therefore, training alternates over the $d+S$ input directions with one linear solve per step, which is the alternating least squares algorithm for CP decompositions (Carroll & Chang, 1970; Kolda & Bader, 2009). Modes are added one at a time (greedy enrichment) (Zhang & Golub, 2001): the M-th mode is the rank-one sub-network that solves the weak form for the residual left by the previous modes (Appendix B), as in the boosting training of interpolating neural networks (Park et al., 2025; Guo et al., 2025) and in PGD (Ammar et al., 2006; Nouy, 2010), and the amplitudes of all modes are updated afterward.

3.3 Inference

Given a new input field $k^\star$, INO first projects it onto the KL modes to obtain its coordinates $\boldsymbol{\xi}^\star$, and then evaluates the univariate functions at $\boldsymbol{\xi}^\star$ and contracts them (Section 4.3). If the leading coordinates of the projected input fall outside the parametric domain, INO does not return a prediction by default; the user can then run a conventional numerical solver, evaluate INO anyway and accept its prediction if the error bound of Section 3.4 is small enough, or train a new INO on a wider parametric domain. INO assumes the input fields are smooth, so they can be accurately represented by a truncated KL expansion. It also assumes that, when the input is a coefficient, it remains positive throughout the entire parametric domain to ensure the problem stays elliptic; no comparable requirement is needed when the input appears in the source term.

3.4 Error bound

INO is constructed using the locally supported basis functions based on C-HiDeNN interpolation, so its prediction is an explicit mesh-based function. This makes it interpretable and allows it to

be inserted back into the discrete PDE. Let $A(k)$ be the matrix of the bilinear form $\mathcal{A}_k$ on the finite element space and F the load vector, so that the discrete solution $u(k)$ for the true input field k satisfies $A(k)\,u(k) = F$, and let K_0 be any symmetric positive definite matrix chosen to measure the error, with the norm $\|v\|_{K_0} = (v^\top K_0 v)^{1/2}$ and its dual norm $\|r\|_{K_0^{-1}} = (r^\top K_0^{-1} r)^{1/2}$. Here $u(k)$ and the prediction u_M are identified with their vectors of nodal values. If $v^\top A(k)\, v \geq \alpha(k)\, v^\top K_0\, v$ holds for every nodal vector v with a coercivity constant $\alpha(k) > 0$, then taking $v = u(k) - u_M$ and using $A(k)(u(k) - u_M) = F - A(k)u_M$ gives the error bound:

$$\|u(k) - u_M\|_{K_0} \;\leq\; \frac{\|F - A(k)\, u_M\|_{K_0^{-1}}}{\alpha(k)}. \tag{3}$$

Eq. (3) is the standard residual bound of reduced-basis methods (Rozza et al., 2008), where the residual is computed for the parametrized problem. In INO, it can be evaluated for either field: with the truncated field of Eq. (1) it bounds the rank error, and with the true input field it bounds the total error. The bound requires only the residual $F - A(k)u_M$ and the constant $\alpha(k)$, both of which are computed from the values of k at the quadrature points. Therefore, it requires no reference solution, holds outside the parametric domain whenever $\alpha(k) > 0$, and applies to any u_M that satisfies the Dirichlet boundary conditions, including the prediction of a trained neural operator (Section 4.2). In this paper, K_0 is the stiffness matrix of the Laplacian, $\int_\Omega \nabla v \cdot \nabla u$, which does not depend on the input field, so that $\|\cdot\|_{K_0}$ is the energy norm and $\alpha(k)$ has a closed form. For diffusion, $\alpha(k) = \min_q k(\boldsymbol{x}_q)$ is the minimum of the coefficient over the quadrature points $\boldsymbol{x}_q$. For the Helmholtz problem $-\Delta u - \kappa^2 u = f$ with the squared wavenumber κ^2 as input, $\alpha(\kappa^2) = 1 - \kappa^2_{\max}/\mu_1$, where $\kappa^2_{\max}$ is the largest value of κ^2 at the quadrature points and μ_1 is the first Dirichlet eigenvalue of the Laplacian, so that $\alpha > 0$ as long as $\kappa^2_{\max} < \mu_1$. If $\alpha \leq 0$, the bound is not defined. The bound measures the error with respect to the discrete solution $u(k)$, so it does not include the discretization error; accordingly, all reference solutions in this paper except that of the moving source use the same discretization as INO.

4 Results

In this section, we compare INO with physics-informed neural operators. All comparisons with the baselines follow the same protocol. Each test input is a sample of the random field with all modes retained, each method receives this true input field, and the reference solution is computed for the true input field, so that the errors in Tables 1 and 4 are total errors. The baselines are trained for the number of iterations given in their original papers (Li et al., 2024; Wang et al., 2021), and all errors are those of the fully trained models. Unless stated otherwise, errors are relative L^2 errors evaluated by Gaussian quadrature at the same points for every method. We also report the error of the mean-input solution, i.e., the solution for the mean input field, as the error of a model that ignores its input (Table 4). The PDEs, input distributions, reference solutions and detailed settings are given in Appendix C. All INO timings are measured on one CPU core.

4.1 Comparison with physics-informed operators

In this example, we compare INO with PI-FNO and PI-DeepONet on the moving-source, diffusion–reaction, diffusion, Helmholtz and nonlinear diffusion problems of Appendix C.1. Each baseline is trained with the settings of its original paper (Appendix C.3). The two-dimensional operators of INO use $S = 25$ KL coordinates, $M = 96$ modes and a parametric domain of ± 2 standard deviations, and the three-dimensional operators use up to $S = 164$ coordinates (Table 2). The moving-source, diffusion–reaction and nonlinear diffusion operators use Hermite parametric factors and have no bounded parametric domain.

INO makes a prediction only if the six leading coordinates of the input, i.e., the coordinates of the six largest KL eigenvalues, lie within the parametric domain; we call this test the domain check. The remaining coordinates are clipped to the parametric domain. Only the leading coordinates are checked, since the trailing KL modes have small amplitudes, so clipping them hardly changes the input field (Appendix D.2). For a Gaussian input field, the probability of passing the check is $0.9545^6 \approx 76\%$, and 71% of our two-dimensional samples pass it. In three dimensions, the check uses the ten leading coordinates and keeps 70% of the samples. Since the moving-source

Table 1: Relative L^2 errors (for the moving source, relative ℓ^2 errors on 500×51 points in (x, t)) of INO and the physics-informed baselines on the test samples within the parametric domain of INO, selected as described in Section 4.1 (all samples: Table 4). Each cell gives the geometric mean over three random initializations and, in parentheses, the ratio of the largest to the smallest error; a cell without parentheses is a single initialization, and **bold** marks the most accurate model. Every model is trained on the mesh of the problem. Training time is given in seconds as INO / PI-FNO / PI-DeepONet, with INO on one CPU core and the baselines on one GPU. The settings of the baselines are given in Appendix C.3.

problem (mesh)	INO	PI-FNO	PI-DeepONet	training time (s)
1D heat, moving source	**2.92**e-3 (1.13)	3.98e-1 (1.25)	3.89e-1 (1.91)	115 / 5,784 / 361
1D diffusion–reaction	**4.84**e-5 (1.41)	2.56e-3 (1.35)	5.21e-3 (1.04)	8 / 53 / 202
2D diffusion 65^2	**4.34**e-3 (1.01)	5.53e-3 (1.45)	5.42e-2 (1.07)	42 / 1,718 / 2,016
2D diffusion 129^2	**4.23**e-3 (1.09)	4.81e-2 (1.18)	5.85e-2 (1.13)	79 / 6,189 / 2,047
2D diffusion 513^2	**4.58**e-3 (1.08)	1.63e-1 (1.02)	5.25e-2 (1.04)	530 / 77,231 / 1,844
2D Helmholtz 65^2	**4.14**e-4 (1.07)	6.17e-3 (1.77)	1.04e-1 (1.45)	22 / 1,768 / 1,973
2D Helmholtz 129^2	**4.17**e-4 (1.08)	6.70e-2 (1.72)	1.17e-1 (1.24)	35 / 6,229 / 1,960
2D Helmholtz 513^2	**3.74**e-4 (1.21)	7.99e-2 (1.00)	9.93e-2 (1.09)	373 / 77,627 / 1,700
3D diffusion 65^3	7.67e-3 (1.02)	**2.08**e-3 (1.53)	2.08e-1 (1.09)	452 / 15,049 / 2,750
3D diffusion 129^3	8.73e-3 (1.12)	**6.57**e-3 (1.15)	2.00e-1 (1.06)	20,499 / 102,879 / 3,269
3D Helmholtz 65^3	**5.88**e-4 (1.08)	1.20e-3 (1.82)	4.07e-1 (1.36)	3,335 / 15,095 / 2,405
3D Helmholtz 129^3	**6.62**e-4 (1.08)	9.35e-4 (1.73)	4.16e-1 (1.26)	3,053 / 103,236 / 3,051
2D nonlinear diffusion 65^2	**2.79**e-3 (1.06)	9.11e-3 (2.25)	2.21e-1 (1.05)	278 / 813 / 2,306
2D nonlinear diffusion 129^2	**2.58**e-3 (1.10)	5.28e-2 (4.46)	2.21e-1 (1.17)	348 / 2,194 / 2,345

and diffusion–reaction operators have no bounded parametric domain, they are evaluated on every sample. The nonlinear diffusion operator has none either, but it is evaluated on the samples that pass the same check (77%), so that all two-dimensional problems are compared on the same kind of samples; on every sample, its error is about 10% larger. All models in Table 1 are evaluated on the same samples, and the results on every sample are given in Appendix D.2. Fig. 2 shows three predictions of INO, each for the test sample with the median error among the samples of Table 1.

INO is the most accurate model in 12 of the 14 cases of Table 1, and each factor below is relative to the more accurate baseline. On two-dimensional Helmholtz, it is 15–214$\times$ more accurate than PI-FNO, and on two-dimensional diffusion, it is 1.3$\times$ more accurate at 65^2 and 11$\times$ at 129^2 and 513^2. It is also more accurate on three-dimensional Helmholtz, by 2.0$\times$ at 65^3 and 1.4$\times$ at 129^3, and on the nonlinear diffusion problem, by 3.3$\times$ at 65^2 and 20$\times$ at 129^2. On the moving-source problem, both baselines have larger errors than the mean-input solution (0.13). PI-FNO is more accurate on three-dimensional diffusion, by 3.7$\times$ at 65^3 and 1.3$\times$ at 129^3.

The 1D diffusion–reaction problem is the space–time benchmark of Wang et al. (2021), $\partial_t u = 0.01\,\partial_{xx} u + 0.01\,u^2 + f(x)$ for $(x, t) \in (0, 1) \times (0, 1]$ with zero initial and boundary conditions, where the source f is the input field. Since the reaction term is a polynomial in u, INO linearizes it about the previous iterate, and each linearized problem keeps the separated form with a number of terms that is independent of S. On this problem, INO is 53$\times$ more accurate than PI-FNO, whereas the mean-input solution, which is zero since the source has zero mean, has an error of 1.00.

The error of INO is nearly independent of the mesh, whereas the error of PI-FNO trained on the 129^2 and 513^2 grids is 6–30$\times$ larger than at 65^2, since its training loss stops decreasing within a few thousand steps on the finer grids. If the network trained at 65^2 is applied on the finer grids instead, PI-FNO is more accurate than INO on two-dimensional diffusion, but not on Helmholtz or nonlinear diffusion (Appendix D.3). PI-DeepONet trains quickly, but on the moving-source, Helmholtz and three-dimensional problems its error is larger than that of the mean-input solution (Appendix C.3). The error of PI-FNO changes by 1.0–4.5$\times$ across random initializations, whereas that of INO changes by at most 1.4$\times$ (Appendix D.4).

As shown in Fig. 3, tightening the domain check from every sample to the setting of Table 1 reduces the error of INO at 65^2 by 2.5$\times$ on diffusion and 7.2$\times$ on Helmholtz, whereas the errors of PI-

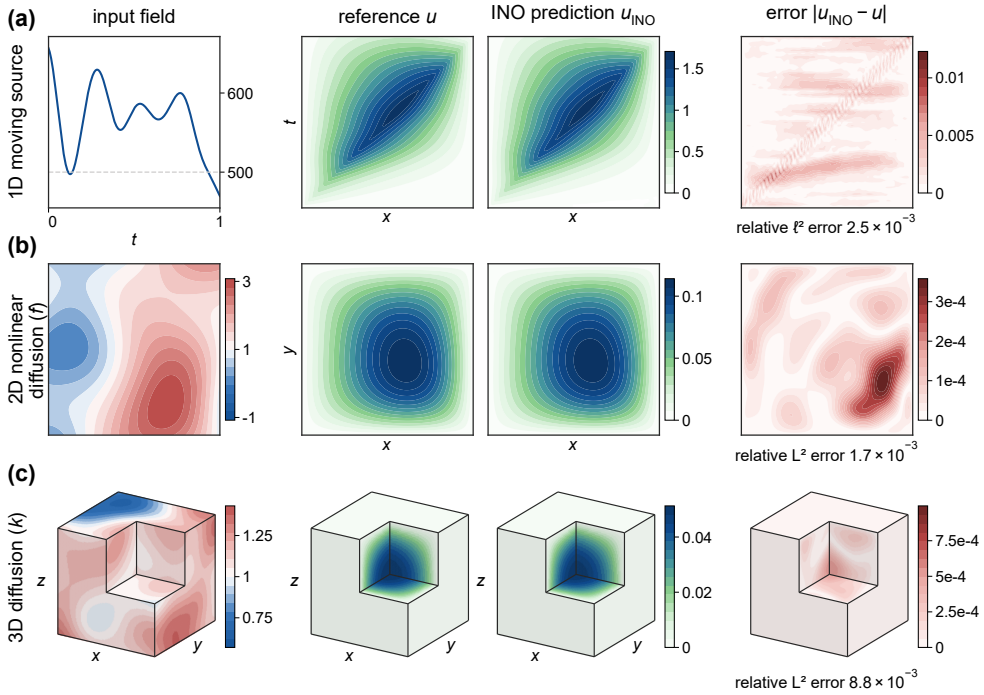


Figure 2: Predictions of INO. (a) The moving heat source on the 2001×51 space–time mesh for a Gaussian sample of the power history $P(t)$, with the operator of Table 1; the reference is a Crank–Nicolson solution, and the relative ℓ^2 error is 2.5×10^{-3}. (b) Two-dimensional nonlinear diffusion at 129^2 with the operator of Table 1, for a Gaussian sample of the source f; the reference is a Newton solve for the true source, and the relative error is 1.7×10^{-3}. (c) Three-dimensional diffusion at 129^3 with the operator of Table 1, for a Gaussian sample of the coefficient k; the reference is solved for the true input field, and the relative error is 8.8×10^{-3}.

FNO and of the mean-input solution change by less than 30%, since INO is trained only on the parametric domain. Most of the error of INO thus comes from the inputs outside the parametric domain, which the domain check identifies from the input before any prediction. We use the check as a coverage policy, in which INO predicts the inputs within its parametric domain and the other inputs are passed to a conventional solver or to an INO trained on a wider parametric domain, as long as the coefficient remains positive over it (Section 3.3). A rejected input is not necessarily predicted poorly, since the check only tests whether the input lies within the domain on which INO is trained. When every sample is included, INO is still 2.5–32$\times$ more accurate on two-dimensional Helmholtz and remains the most accurate model on two-dimensional diffusion at 129^2 and 513^2 and on the nonlinear diffusion problem, whereas PI-FNO is the most accurate model on two-dimensional diffusion at 65^2, by 1.8$\times$, and on every three-dimensional problem (Appendix D.2).

4.2 Rank error and error bound

In this example, we study how the error of INO decreases with the rank M and how closely the bound of Eq. (3) follows it. We use the two-dimensional diffusion operator of Table 1 at 65^2 ($S = 25$) and the test samples within its parametric domain. In the relative energy norm, the total error decreases from 6.0×10^{-2} at $M = 8$ to 1.3×10^{-2} at $M = 96$, and the rank error from 5.8×10^{-2} to 9.4×10^{-3} (Fig. 4). The gap between the two errors is caused by the truncation of the input field, so it does not decrease with M.

The bound is never violated in 2,000 evaluations, and it is 1.43–1.46 times the true error at every rank, for both the total error and the rank error. The bound thus follows the true error closely, and the rank beyond which adding modes barely reduces the total error can be read from the bound alone,

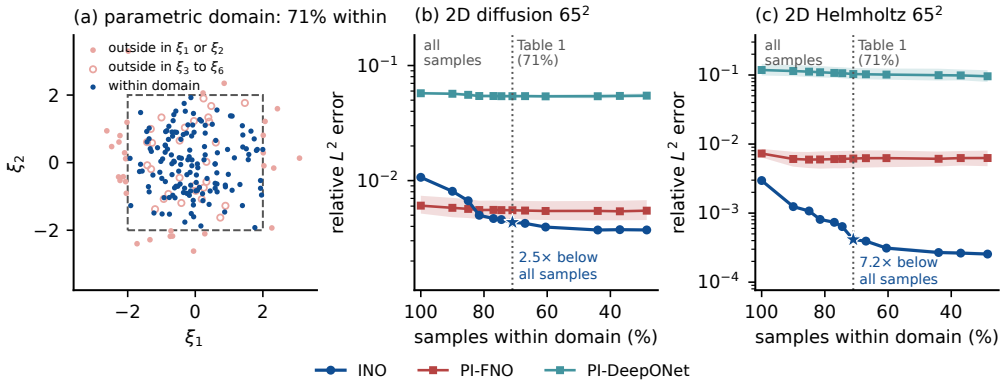


Figure 3: Accuracy versus the fraction of samples within the parametric domain at 65^2. (a) The 200 diffusion samples in the plane of their two leading coordinates ξ_1 and ξ_2 and the parametric domain (dashed); since the check uses the six leading coordinates, the hollow samples lie inside this square but outside the parametric domain in ξ_3 to ξ_6. (b, c) Error versus the fraction of samples within the parametric domain as the check is tightened, with every model evaluated on the same samples (geometric mean and range over three initializations). The star marks the setting of Table 1, and the left end of each curve includes every sample.

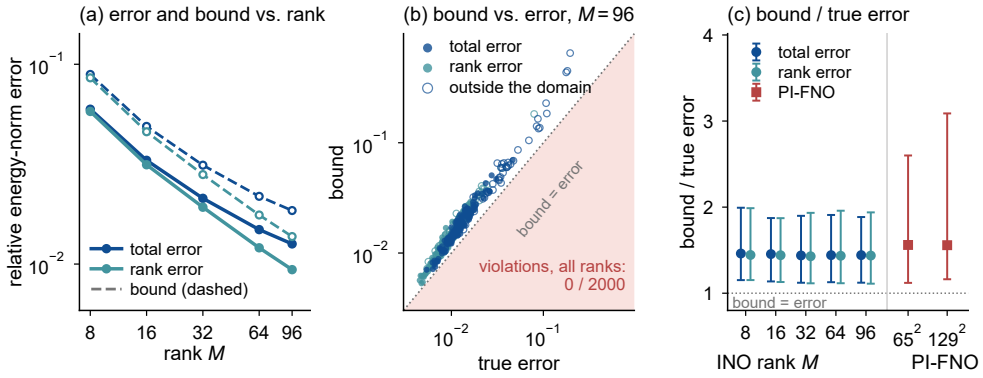


Figure 4: Error bound of Eq. (3) for the two-dimensional diffusion operator of Table 1 at 65^2 ($S = 25$, parametric domain of ± 2 standard deviations), on the 200 test samples of the full random field, 71% of which lie within the parametric domain. Errors are relative energy-norm errors; the relative L^2 error of the same predictions at $M = 96$ is 4.3×10^{-3} (Table 1); blue: total error with respect to the true input field, teal: rank error with respect to the truncated field. (a) Mean true error (solid) and mean bound (dashed) versus rank, samples within the domain. (b) Bound versus true error for every sample at $M = 96$; the shaded region would contain violations. (c) Ratio of the bound to the true error (mean, minimum and maximum over the samples within the domain) for INO at every rank and for the predictions of PI-FNO trained at 65^2 and applied at 65^2 and 129^2.

without any reference solution. The rank error also hardly depends on the mesh, changing by less than a factor of two from 65^2 to 513^2 for $S = 9$ and $M = 96$ (Appendix D.1).

Since Eq. (3) only requires a prediction that satisfies the Dirichlet boundary conditions, it can also be applied to the baselines. PI-FNO satisfies them exactly, since its output is multiplied by a mollifier that vanishes on the boundary (Appendix C.3). For PI-FNO on two-dimensional diffusion, trained at 65^2 and applied at 65^2 and 129^2, the bound is never violated in 240 predictions, and its mean ratio to the true energy-norm error is 1.5–1.6 (Fig. 4(c)). On the other hand, the predictions of PI-DeepONet,

whose boundary conditions are imposed only by a penalty term, violate the boundary condition by about 12% of the solution maximum, so the bound does not apply to them and is violated in 117 of 120 cases. After the boundary values are imposed, the bound holds again.

4.3 Efficiency and error analysis

As shown in Table 1, training INO takes 22–530 s on the two-dimensional problems, compared with 813–77,627 s for PI-FNO and 1,700–2,345 s for PI-DeepONet on one GPU. On the three-dimensional problems at 129^3, it takes 3,053–20,499 s, compared with 102,879–103,236 s for PI-FNO and 3,051–3,269 s for PI-DeepONet. The training time of INO grows much more slowly than the number of degrees of freedom. From 65^2 to 513^2, the degrees of freedom increase by a factor of 62 and the training time by a factor of 13 to 17, since mesh refinement only affects d of the $d + S$ directions (Appendix D.7).

INO stores $M(d\,n + S\,n_\xi)$ numbers, from 91,520 for the $S = 8$, $M = 128$ operator of Table 3 to about 2.2 million for the $S = 164$ operators of Table 1 at 129^3, which is about the number of nodal values of a single solution on that mesh ($129^3 \approx 2.1$ million). The inference cost is compared with a conventional numerical solve in Table 6 (Appendix D.5). The speed-up increases from 484× at 65^2 to 2,892× at 513^2, since an inference only contracts one-dimensional factors, whereas the cost of a numerical solve grows faster than the number of degrees of freedom. Against a matrix-free solve of the same C-HiDeNN discretization, the speed-up at 65^2 is 375× for $M = 64$. For an input given by its KL coordinates, an inference together with its total-error bound costs about half of a numerical solve at 65^2 and a quarter at 129^2, and for a full random-field input, the bound costs 0.2–0.5 s (Appendix D.5).

We further examine the source of the error of INO. The prediction is a sum of spatial modes $\prod_i u^m_{x_i}$ weighted by modal coefficients $\prod_j u^m_j(\xi^\star_j)$, which depend on the input. At 65^2 with $S = 9$ and $M = 96$, the best approximation of the reference solution in the space spanned by the spatial modes has a relative error of 1.4×10^{-5}, whereas the error of INO is 3.6×10^{-4}. Therefore, the error comes from the modal coefficients rather than from the spatial modes. Since the matrix $A(k)$ is affine in $\boldsymbol{\xi}$ for the coefficient inputs, the modal coefficients can instead be computed for a given input by solving a reduced system of size $r \leq M$. This reduces the error by one to two orders of magnitude on the linear problems, at an inference cost of about one millisecond or less (Appendix D.6).

5 Limitations

The error of INO is limited by the truncation of the input field rather than by the rank (Section 4.2). The parametric domain of ± 2 standard deviations excludes 29% of the samples, which require a conventional numerical solve or an INO parameterized on a wider parametric domain. A coefficient input must remain positive over the whole parametric domain, which limits its contrast, whereas source and reaction inputs have no such limit. The input fields must also be smooth, and the bound does not cover the discretization error.

6 Conclusion

In this paper, we introduced INO, a data-free interpolating neural network for the solution operators of parametric PDEs, which is trained by one-dimensional linear solves and whose total error is bounded by the residual for the true input field on coercive problems. INO is the most accurate model on most of the problems within its parametric domain, whereas the physics-informed FNO is more accurate on three-dimensional diffusion. In future work, we will use the computable bound to adapt the rank and the parametric domain during training, extend INO to transport-dominated and multiphysics problems, and use its certified predictions for design optimization, inverse problems and uncertainty quantification.

AI use statement

In this work, we used generative AI tools for implementing the baseline Crank-Nicolson finite difference solvers. We have not used generative AI tools for generating synthetic data sets, helping

develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, cleaning or reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results. Additionally, we used generative AI tools for tasks such as editing a research paper to improve readability, identifying relevant literature, and formatting references. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This work develops AI-enhanced numerical methods for parametric partial differential equations to approximate PDE operators. We are not aware of ethical concerns specific to this work beyond those that apply to scientific computing in general.

Reproducibility statement

Section 3 and Appendices A and B describe the network, the training step and the inference of INO, including the one-dimensional systems solved during training. Appendix C.1 defines every problem, input distribution and reference solver; Appendix C lists the discretization, rank, parametric domain and solver settings of every experiment (Table 2); and Appendix C.3 gives the architectures, training budgets and evaluation of the baselines, which follow their published implementations. All results use fixed random seeds and the same test samples for every model, and the variability over initializations is reported in Appendix D.4.

References

Amine Ammar, Béchir Mokdad, Francisco Chinesta, and Roland Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. *Journal of Non-Newtonian Fluid Mechanics*, 139(3):153–176, 2006. doi: 10.1016/j.jnnfm.2006.07.007.

Markus Bachmayr. Low-rank tensor methods for partial differential equations. *Acta Numerica*, 32: 1–121, 2023.

Peter Binev, Albert Cohen, Wolfgang Dahmen, Ronald DeVore, Guergana Petrova, and Przemyslaw Wojtaszczyk. Convergence rates for greedy algorithms in reduced basis methods. *SIAM Journal on Mathematical Analysis*, 43(3):1457–1472, January 2011. ISSN 1095-7154. doi: 10.1137/100795772. URL http://dx.doi.org/10.1137/100795772.

Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

J. Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart–Young" decomposition. *Psychometrika*, 35(3):283–319, 1970. doi: 10.1007/BF02310791.

Francisco Chinesta, Pierre Ladeveze, and Elías Cueto. A short review on model order reduction based on proper generalized decomposition. *Archives of Computational Methods in Engineering*, 18(4):395–404, October 2011. ISSN 1886-1784. doi: 10.1007/s11831-011-9064-7. URL http://dx.doi.org/10.1007/s11831-011-9064-7.

Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36:23761–23788, 2023.

Albert Cohen, Ronald DeVore, and Christoph Schwab. Convergence rates of best n-term galerkin approximations for a class of elliptic spdes. *Foundations of Computational Mathematics*, 10(6): 615–646, 2010.

Martin Eigel, Claude Jeffrey Gittelson, Christoph Schwab, and Elmar Zander. Adaptive stochastic Galerkin FEM. *Computer Methods in Applied Mechanics and Engineering*, 270:247–269, 2014. doi: 10.1016/j.cma.2013.11.015.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), October 2001. ISSN 0090-5364. doi: 10.1214/aos/1013203451. URL http://dx.doi.org/10.1214/aos/1013203451.

Han Gao, Luning Sun, and Jian-Xun Wang. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428:110079, 2021.

Roger G. Ghanem and Pol D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer New York, 1991. ISBN 9781461230946. doi: 10.1007/978-1-4612-3094-6. URL http://dx.doi.org/10.1007/978-1-4612-3094-6.

Tamara G Grossmann, Urszula Julia Komorowska, Jonas Latz, and Carola-Bibiane Schönlieb. Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics*, 89(1):143–174, 2024. doi: 10.1093/imamat/hxae011.

Jiachen Guo, Chanwook Park, Xiaoyu Xie, Zhongsheng Sang, Gregory J Wagner, and Wing Kam Liu. Convolutional hierarchical deep learning neural networks-tensor decomposition (c-hidenn-td): a scalable surrogate modeling approach for large-scale physical systems. *arXiv preprint arXiv:2409.00329*, 2024.

Jiachen Guo, Xiaoyu Xie, Chanwook Park, Hantao Zhang, Matthew Politis, Gino Domel, Thomas J. R. Hughes, and Wing Kam Liu. Interpolating neural network-tensor decomposition (INN-TD): a scalable and interpretable approach for large-scale physics-based problems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21138–21162. PMLR, 2025.

Jiachen Guo, Chanwook Park, Dong Qian, Thomas JR Hughes, and Wing Kam Liu. Large language model-empowered next-generation computer-aided engineering. *Computer Methods in Applied Mechanics and Engineering*, 450:118591, 2026.

Owen Huang, Sourav Saha, Jiachen Guo, and Wing Kam Liu. An introduction to kernel and operator learning methods for homogenization by self-consistent clustering analysis. *Computational Mechanics*, 72(1):195–219, 2023.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3): 455–500, 2009.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

Pierre Ladevèze and Ludovic Chamoin. On the verification of model reduction methods based on the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 200(23–24):2032–2047, 2011. doi: 10.1016/j.cma.2011.02.019.

Hengyang Li, Stefan Knapik, Yangfan Li, Chanwook Park, Jiachen Guo, Satyajit Mojumder, Ye Lu, Wei Chen, Daniel W Apley, and Wing Kam Liu. Convolution hierarchical deep-learning neural network tensor decomposition (c-hidenn-td) for high-resolution topology optimization. *Computational Mechanics*, 72(2):363–382, 2023.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024. doi: 10.1145/3648506.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Ye Lu, Hengyang Li, Lei Zhang, Chanwook Park, Satyajit Mojumder, Stefan Knapik, Zhongsheng Sang, Shaoqiang Tang, Daniel W Apley, Gregory J Wagner, et al. Convolution hierarchical deep-learning neural networks (c-hidenn): finite elements, isogeometric analysis, tensor decomposition, and beyond. *Computational Mechanics*, 72(2):333–362, 2023.

Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6 (10):1256–1269, 2024. doi: 10.1038/s42256-024-00897-5.

Anthony Nouy. A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 196 (45–48):4521–4537, 2007. doi: 10.1016/j.cma.2007.05.016.

Anthony Nouy. A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1603–1626, April 2010. ISSN 0045-7825. doi: 10.1016/j.cma.2010.01.009. URL http://dx.doi.org/10.1016/j.cma.2010.01.009.

Chanwook Park, Ye Lu, Sourav Saha, Tianju Xue, Jiachen Guo, Satyajit Mojumder, Daniel W Apley, Gregory J Wagner, and Wing Kam Liu. Convolution hierarchical deep-learning neural network (c-hidenn) with graphics processing unit (gpu) acceleration. *Computational Mechanics*, 72(2): 383–409, 2023.

Chanwook Park, Sourav Saha, Jiachen Guo, Hantao Zhang, Xiaoyu Xie, Miguel A. Bessa, Dong Qian, Wei Chen, Gregory J. Wagner, Jian Cao, Thomas J. R. Hughes, and Wing Kam Liu. Unifying machine learning and interpolation theory via interpolating neural networks. *Nature Communications*, 16(1):8753, 2025. doi: 10.1038/s41467-025-63790-8.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

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.

G. Rozza, D. B. P. Huynh, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: Application to transport and continuum mechanics. *Archives of Computational Methods in Engineering*, 15 (3):229–275, May 2008. ISSN 1886-1784. doi: 10.1007/s11831-008-9019-9. URL http://dx.doi.org/10.1007/s11831-008-9019-9.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40), October 2021. ISSN 2375-2548. doi: 10.1126/sciadv.abi8605. URL http://dx.doi.org/10.1126/sciadv.abi8605.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Tong Zhang and Gene H. Golub. Rank-one approximation to high order tensors. *SIAM Journal on Matrix Analysis and Applications*, 23(2):534–550, 2001. doi: 10.1137/S0895479899352045.

A C-HiDeNN INTERPOLATION AND STRUCTURE OF INO

In this section, we describe the C-HiDeNN sub-network used for the factors of INO and how INO combines the sub-networks.

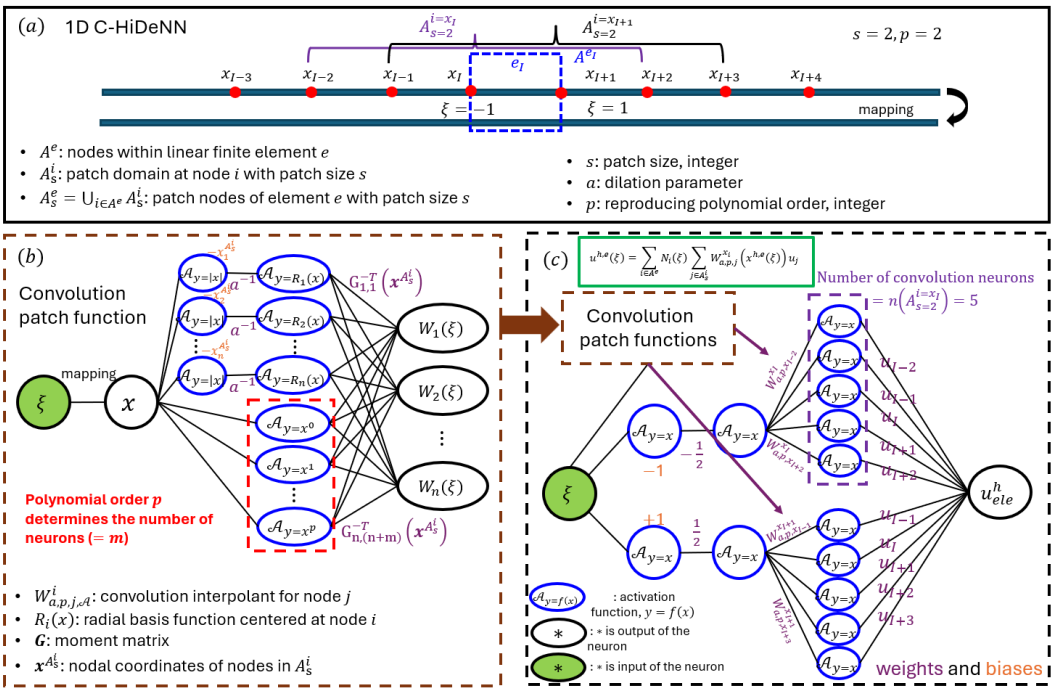


Figure 5: C-HiDeNN interpolation of a one-dimensional factor interpreted as a neural network. (a) Element, patch and hyperparameters s, a, p. (b) Convolution patch function formed from radial basis and polynomial neurons. (c) The factor as a partially connected network whose output is contracted with the nodal values. Adapted from Park et al. (2025).

A.1 C-HiDeNN SUB-NETWORK

Convolution hierarchical deep-learning neural network (C-HiDeNN) interpolation combines finite element interpolation, meshfree interpolation and machine learning (Lu et al., 2023; Park et al., 2023). A one-dimensional C-HiDeNN interpolation can be written as:

$$u(x) = \sum_{I \in A^e} N_I(x) \sum_{J \in A^I_s} W^{(J)}_I(x; s, a, p)\, u_J = \sum_{J \in A^e_s} \widetilde{N}_J(x; s, a, p)\, u_J = \widetilde{\boldsymbol{N}}(x)\, \boldsymbol{u}, \tag{4}$$

where $N_I(x)$ is the linear finite element basis function of node I of element e; A^e is the set of nodes of the element; $W^{(J)}_I(x; s, a, p)$ is the convolution patch function of node J on the patch A^I_s of the $2s{+}1$ nodes around node I; $A^e_s = \bigcup_{I \in A^e} A^I_s$ is the set of patch nodes of the element; and u_J is the nodal value at node J. As shown in Fig. 5, Eq. (4) can be interpreted as a partially connected neural network whose hidden layers produce the functions N_I and $W^{(J)}_I$ and whose only trainable parameters are the nodal values u_J. The patch functions are controlled by the patch size s, the dilation parameter a and the reproducing order p, so the function space can be adapted without changing the number of nodes.

On each element, the C-HiDeNN basis functions can be written as:

$$\begin{aligned} \widetilde{N}_J(x) &= \sum_{I \in A^e} N_I(x)\, W^{(J)}_I(x), \\ \boldsymbol{W}^I(x)^\top &= \left[\boldsymbol{R}^I(x)^\top \;\; \boldsymbol{p}(x)^\top\right] \mathbf{G}_I^{-1}, \qquad \mathbf{G}_I = \begin{bmatrix} \mathbf{R}_0 & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{0} \end{bmatrix}, \end{aligned} \tag{5}$$

where $\boldsymbol{W}^I(x)$ collects the patch functions $W^{(J)}_I(x)$ of the nodes $J \in A^I_s$; $\mathbf{R}_0 = [R(x_J - x_{J'})]$ and $\mathbf{P} = [\boldsymbol{p}(x_J)^\top]$ run over the nodes J, J' of that patch; $\boldsymbol{p}(x) = [1, x, \ldots, x^p]$; $\boldsymbol{R}^I(x) = [R(x - x_J)]$; and the entries of the product associated with the polynomial constraints are not used. The radial

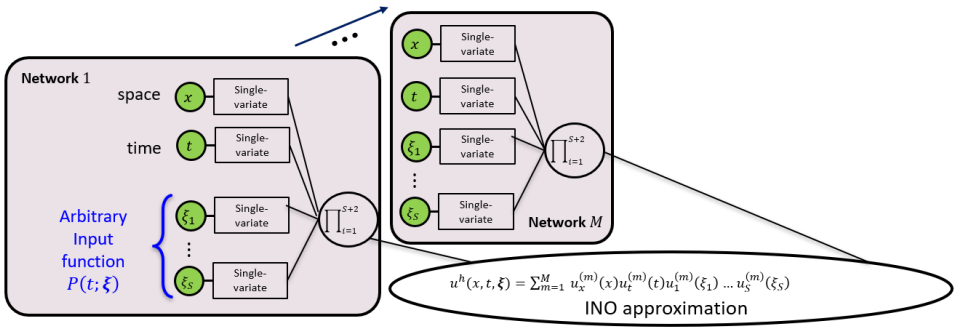


Figure 6: Structure of INO for the moving heat source. Each of the M networks has one single-variate sub-network per input, namely x, t and the KL coordinates of the power history P; the outputs of the sub-networks are multiplied, and the M networks are summed, as in Eq. (6).

basis function is the cubic spline $R(z) = \frac{2}{3} - 4z^2 + 4z^3$ for $0 \le z \le \frac{1}{2}$, $R(z) = \frac{4}{3} - 4z + 4z^2 - \frac{4}{3}z^3$ for $\frac{1}{2} < z \le 1$, and zero otherwise, with $z = |x - x_J|/a$, where a is a multiple of the element size h (Appendix C).

The C-HiDeNN basis functions are interpolatory and subject to the Kronecker delta property, i.e., $\widetilde{N}_I(x_J) = \delta_{IJ}$. As a result, the nodal values are the values of the function at the nodes, and a homogeneous Dirichlet condition is imposed exactly by removing the boundary unknowns. The basis functions also reproduce polynomials up to degree p. For a two-dimensional diffusion problem with a smooth random coefficient, the L^2 error of C-HiDeNN with $s = 3$ and $p = 2$ converges with order 3.44, compared with 2.00 for linear elements, and it is $72\times$ smaller at 65^2. Therefore, the discretization error stays far below the rank error in all experiments.

A.2 Structure of INO

We use the moving heat source of Appendix C.1 to show how INO combines the sub-networks. The input of this problem is the power history $P(t)$, which is represented by its first $S = 18$ KL coordinates $\xi_1, \dots, \xi_S$, and the solution operator maps P to the space–time field $u(x,t)$. INO approximates this operator as:

$$u(x,t;\boldsymbol{\xi}) \approx \sum_{m=1}^{M} u_x^m(x)\, u_t^m(t) \prod_{j=1}^{S} u_j^m(\xi_j), \tag{6}$$

where u_x^m, u_t^m and u_j^m are the single-variate sub-networks of the m-th mode in x, t and ξ_j. As shown in Fig. 6, each mode is a network with one sub-network per input; the outputs of the sub-networks are multiplied, and the M networks are summed. The sub-networks in x and t are C-HiDeNN interpolants of Eq. (4), and the parametric sub-networks of this problem are the Hermite expansions of Appendix C.1. For the other problems, the inputs x and t are replaced by the d spatial coordinates, and every sub-network is a C-HiDeNN interpolant, which gives Eq. (2).

B Separated form, training step and inference

In this section, we derive the separated form of the weak form for two-dimensional diffusion, the one-dimensional training step of Section 3.2 and the inference of Section 3.3.

B.1 Separated form

Sans-serif letters denote one-dimensional matrices, and K and M are the stiffness and mass matrices $\int \widetilde{\boldsymbol{N}}'^{\top} \widetilde{\boldsymbol{N}}'$ and $\int \widetilde{\boldsymbol{N}}^{\top} \widetilde{\boldsymbol{N}}$ of a one-dimensional C-HiDeNN mesh. Since k is affine in $\boldsymbol{\xi}$, substituting Eq. (1) into the bilinear form $\mathcal{A}_k$ of $-\nabla\cdot(k\nabla u) = f$ splits it, in one spatial dimension, into exactly

$S{+}1$ separated terms, which can be written as:

$$\int k(\boldsymbol{x};\boldsymbol{\xi})\,\nabla\delta u\cdot\nabla u\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{\xi} \;=\; \underbrace{\bar{k}\,\mathsf{K}\otimes\bigotimes_{j'}\mathsf{M}_{j'}}_{\text{mean}} \;+\; \sum_{j=1}^{S}\underbrace{\mathsf{K}^{(j)}\otimes\mathsf{M}_j^{\xi}\otimes\bigotimes_{j'\neq j}\mathsf{M}_{j'}}_{\text{one weighted direction}}, \tag{7}$$

where $\mathsf{K}^{(j)} = \int \phi_j\,\widetilde{\boldsymbol{N}}'^{\top}\widetilde{\boldsymbol{N}}'$ is the stiffness matrix weighted by the KL mode ϕ_j; M_j is the mass matrix of parameter direction j; and $\mathsf{M}_j^{\xi} = \int \widetilde{\boldsymbol{N}}^{\top}\xi\,\widetilde{\boldsymbol{N}}\,\mathrm{d}\xi$ is its ξ-weighted mass matrix.

In two dimensions, the covariance is a product over the spatial directions, so every KL mode is a product of one-dimensional functions, $\phi_j(x,y) = \phi_j^x(x)\,\phi_j^y(y)$. Each term of Eq. (7) then splits into one term per gradient component, whose matrix in x is the stiffness or mass matrix weighted by ϕ_j^x and whose matrix in y is the other of the two, weighted by ϕ_j^y (both unweighted for the mean term). This gives $T = 2(S{+}1)$ separated terms in two dimensions and $T = d(S{+}1)$ in d dimensions, so the matrix of the bilinear form can be written as:

$$A = \sum_{t=1}^{T}\gamma_t\,\mathsf{A}_x^t\otimes\mathsf{A}_y^t\otimes\bigotimes_{j}\mathsf{P}_j^t, \tag{8}$$

where γ_t is the scalar coefficient of term t ($\bar{k}$ for the mean terms and 1 otherwise); A_x^t and A_y^t are its one-dimensional spatial matrices; and $\mathsf{P}_j^t \in \{\mathsf{M}_j, \mathsf{M}_j^{\xi}\}$ is its matrix in parameter direction j.

B.2 Training step

For every problem, INO is trained on the Galerkin weak form of Section 3.2. Modes are added one at a time, and the M-th mode w_M is the rank-one sub-network that satisfies the weak form for the residual left by the previous modes, which can be written as:

$$\int_{\Omega_{\boldsymbol{\xi}}}\mathcal{A}_k\big(\delta w,\,u_{M-1}+w_M\big)\,\mathrm{d}\boldsymbol{\xi} \;=\; \int_{\Omega_{\boldsymbol{\xi}}}F(\delta w)\,\mathrm{d}\boldsymbol{\xi}, \qquad u_M \;=\; \sum_{m=1}^{M}c_m\,w_m, \tag{9}$$

where δw is any variation of one factor of w_M; w_m is the m-th rank-one mode; c_m is its amplitude; and u_M is the prediction with M modes. For a symmetric operator, Eq. (9) is the stationarity condition of the energy functional $\mathcal{J}(u) = \int_{\Omega_{\boldsymbol{\xi}}}\big[\frac{1}{2}\mathcal{A}_k(u,u) - F(u)\big]\,\mathrm{d}\boldsymbol{\xi}$, so every new mode decreases $\mathcal{J}$, as in the gradient boosting of Friedman (2001). If δw varies only one factor of the new mode, every other direction contributes a scalar to each term, so the T terms reduce to a weighted sum of one-dimensional matrices. For the x direction of the new mode $v_x\otimes v_y\otimes\bigotimes_j v_j$, the resulting linear system can be written as:

$$\Big[\textstyle\sum_t \gamma_t\,\theta_t\,\mathsf{A}_x^t\Big]\,v_x \;=\; \boldsymbol{q}, \tag{10}$$

where $\theta_t = (v_y^{\top}\mathsf{A}_y^t v_y)\prod_j(v_j^{\top}\mathsf{P}_j^t v_j)$ collects the inner products of the fixed factors for term t; and $\boldsymbol{q}$ contains the load and the previous modes through the same products. Therefore, each step is one dense solve of size n, the cost grows linearly with S, and no matrix of the size of the spatial grid is assembled. After each enrichment, the amplitudes c_m of all modes are updated by a Galerkin projection.

B.3 Inference

For a new input field $k^{\star}$, the field is projected onto the KL modes, and the network is evaluated at the resulting coordinates, which can be written as:

$$\boldsymbol{\xi}^{\star} = \boldsymbol{\Phi}^{+}\big(k^{\star}-\bar{k}\big), \qquad u(\cdot\,;\boldsymbol{\xi}^{\star}) = \sum_{m=1}^{M}u_{\boldsymbol{x}}^m(\cdot)\prod_{j=1}^{S}u_j^m(\xi_j^{\star}), \tag{11}$$

where $u_{\boldsymbol{x}}^m = \prod_{i=1}^{d}u_{x_i}^m$ is the spatial part of the m-th mode; $\boldsymbol{\Phi}$ contains the scaled modes ϕ_j at the quadrature points; and $\boldsymbol{\Phi}^{+}$ is its pseudo-inverse, which is a projection since the modes are orthogonal. An inference requires S one-dimensional interpolations per mode and one contraction, and the projected coordinates $\boldsymbol{\xi}^{\star}$ are also used for the domain check of Section 3.3.

C EXPERIMENTAL SETUP

In this section, we define each problem, its input field and its reference solution, and then give the settings of INO and of the baselines.

C.1 PROBLEM DEFINITIONS

All input fields are Gaussian random fields with the squared-exponential covariance $C(\boldsymbol{x}, \boldsymbol{x}') = \exp\big(-\|\boldsymbol{x}-\boldsymbol{x}'\|^2/2\ell^2\big)$ on the unit domain, where ℓ is the correlation length. A sample can be written as:

$$g(\boldsymbol{x};\boldsymbol{\zeta}) = \sum_j \zeta_j\, \phi_j(\boldsymbol{x}), \qquad \zeta_j \sim \mathcal{N}(0,1) \text{ i.i.d.}, \tag{12}$$

where ϕ_j is the j-th KL mode scaled by the square root of its eigenvalue, in decreasing order of the eigenvalues; and ζ_j is its coordinate. For an input $\bar{k} + \sigma g$, the first S coordinates ζ_j are the ξ_j of Eq. (1). A test sample retains all modes of this expansion (256 in two dimensions and 533 in three), whereas INO retains only the first S (Table 2). All methods are evaluated on the same test samples, 200 per problem in two dimensions (60 at 513^2), 40 in three dimensions and 100 for the one-dimensional problems. Every reference solution is computed for the true input field and, except for the moving source, with the same discretization as INO: C-HiDeNN for diffusion and Helmholtz, and linear elements for the nonlinear diffusion and diffusion–reaction problems (Table 2).

Moving heat source. We consider the one-dimensional transient problem:

$$\begin{gathered}\partial_t u(x,t) - \partial_{xx} u(x,t) = P(t)\, \exp\big(-2(x-t)^2/\epsilon^2\big), \quad (x,t) \in (0,1)\times(0,1], \\ u(0,t) = u(1,t) = 0, \qquad u(x,0) = 0,\end{gathered} \tag{13}$$

where $\epsilon = 0.01$ is the width of a heat spot that crosses the domain at unit speed. The input is the power history $P(t) = 500 + \sigma g(t)$ with $\sigma = 150/\sqrt{3} \approx 86.6$ and $\ell = 0.1$. INO retains the 18 leading KL coordinates, and its parametric factors are the Hermite polynomials $\mathrm{He}_a(\xi_j)/\sqrt{a!}$, $a \le 2$, which are orthonormal under the standard normal density and contain the exact parametric factors 1 and ξ_j, since the source is affine in $\boldsymbol{\xi}$. Therefore, the operator has no bounded parametric domain. The reference solution is a Crank–Nicolson finite difference solution on a finer 2001×2001 grid, and errors are relative ℓ^2 errors on 500×51 points in (x,t).

Diffusion. We consider steady diffusion with a random coefficient:

$$-\nabla \cdot \big(k(\boldsymbol{x};\boldsymbol{\zeta})\, \nabla u(\boldsymbol{x})\big) = 1, \quad \boldsymbol{x} \in (0,1)^d, \qquad u(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \partial(0,1)^d, \tag{14}$$

with $d = 2$ and 3, where $k = \bar{k} + \sigma g$ with $\bar{k} = 1$, $\sigma = 0.2$ and $\ell = 0.3$. The operator maps k to u. The reference solution is computed by the preconditioned conjugate gradient method, with the operator for the mean input field as preconditioner, until the relative residual is below 10^{-11} (10^{-10} in three dimensions); at 129^2, it agrees with a sparse direct solve to 5.2×10^{-12}.

Helmholtz. We consider the Helmholtz equation with a random squared wavenumber:

$$-\Delta u(\boldsymbol{x}) - \kappa^2(\boldsymbol{x};\boldsymbol{\zeta})\, u(\boldsymbol{x}) = 1, \quad \boldsymbol{x} \in (0,1)^d, \qquad u(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \partial(0,1)^d, \tag{15}$$

with $d = 2$ and 3, where $\kappa^2 = \bar{\kappa}^2 + \sigma g$ with $\bar{\kappa}^2 = 10$, $\sigma = 1$ and $\ell = 0.3$. The parametric domain is chosen such that κ^2 stays below the first Dirichlet eigenvalue of the Laplacian, $\mu_1 = 2\pi^2 \approx 19.7$ in two dimensions and $3\pi^2 \approx 29.6$ in three, so that the problem is coercive for every input within the parametric domain. The operator maps κ^2 to u, and the reference solution is computed as for diffusion.

Nonlinear diffusion. We consider the 2D nonlinear equation:

$$-\nabla \cdot \big[(1 + \beta\,|\nabla u|^2)\, \nabla u(\boldsymbol{x})\big] = f(\boldsymbol{x};\boldsymbol{\zeta}), \quad \boldsymbol{x} \in (0,1)^2, \qquad u(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \partial(0,1)^2, \tag{16}$$

with $\beta = 1$, where $f = \bar{f} + \sigma g$ with $\bar{f} = 1$, $\sigma = 1$ and $\ell = 0.3$, so that the source changes sign within the domain. The operator maps f to u. The reference solution is computed by Newton's method, with each step solved by a sparse direct solver, until the norm of the residual is below 10^{-10}.

Diffusion–reaction. Following Wang et al. (2021), we consider the space–time problem:

$$\begin{aligned} \partial_t u(x,t) &= 0.01\,\partial_{xx} u(x,t) + 0.01\,u(x,t)^2 + f(x), \quad (x,t) \in (0,1) \times (0,1], \\ u(x,0) &= 0, \qquad u(0,t) = u(1,t) = 0, \end{aligned} \tag{17}$$

where the source f is a zero-mean Gaussian random field with $\ell = 0.2$, as in Wang et al. (2021). The operator maps f to $u(x,t)$. The reference solution is computed by Newton's method as for nonlinear diffusion, until the norm of the residual is below 10^{-12}.

C.2 Discretization and settings

For the coefficient problems, INO uses C-HiDeNN with $s = 3$, $a = 20$, $p = 2$ and three quadrature points per element in every direction (Appendix A); for the nonlinear diffusion and diffusion–reaction problems, it uses linear elements. INO is trained on one CPU core by greedy enrichment, and the parametric factors of each new mode are initialized with Gaussian bumps whose centers and widths differ between directions. For the moving source, whose input enters only the load, all $M = 500$ modes are instead updated together by block alternating least squares with five sweeps. For the nonlinear diffusion problem, the coefficient $1 + \beta|\nabla u|^2$ is evaluated for the previous iterate and frozen. Since this coefficient is itself in separated form, every iteration is a linear problem of the form of Eq. (8) with a number of terms independent of S, and the coefficient is compressed to rank 48 before each iteration. The settings of each experiment are listed in Table 2.

Table 2: Settings of each experiment. L is the width of the parametric domain in standard deviations, followed by the number of elements per parametric direction; "Hermite" marks Hermite parametric factors, which have no bounded parametric domain; "greedy" marks a rank set by the greedy enrichment; "candidates" is the number of random initial guesses per mode; and α_Ω is the smallest coercivity constant over the parametric domain.

experiment	mesh	S	domain L / elements	rank M	other
Table 1, moving source	2001×51	18	Hermite, no bounded parametric domain	500	block ALS, 5 sweeps
Table 1, Fig. 4, 2D diffusion, Helmholtz	65^2–513^2	25	4 / 80	96	$\sigma = 0.2$ / 1, $\bar{\kappa}^2 = 10$
Table 1, 3D diffusion	65^3, 129^3	56, 164	4 / 80	300, 160	3 candidates per mode
Table 1, 3D Helmholtz	65^3, 129^3	164	4 / 80	160	3 candidates per mode; $\sigma = 1$
Table 1, nonlinear diffusion	65^2, 129^2 (linear elements)	49	Hermite, no bounded parametric domain	295, 297	coefficient rank 48
Table 1, diffusion–reaction	65×65	16	Hermite	greedy	linear elements
Table 3, rank study	65^2 (513^2)	9	2 / 40	4–96	$\sigma = 0.2$
Table 8	33^d	8	2 / 40	64	$\alpha_\Omega = 0.45$

C.3 Baselines

The baselines use the settings of their original papers and are trained on one NVIDIA RTX A6000 GPU each. PI-FNO follows Li et al. (2024), with four Fourier layers of width 64 and 20 Fourier modes in two dimensions (26.2 M parameters) and 12 modes in three dimensions (226.5 M parameters). On the diffusion, Helmholtz and nonlinear diffusion problems, its output is multiplied by the mollifier $\sin(\pi x)\sin(\pi y)$ of Li et al. (2024), with one sine factor per direction in three dimensions, so that its predictions vanish on the boundary. It is trained for 15,001 Adam steps with a batch size of 20 and a learning rate of 10^{-3}, halved at steps 5k, 7.5k and 10k. PI-DeepONet follows Wang et al. (2021), with branch and trunk networks of five 50-unit layers. It is trained for 120,000 steps with batches of 10,000 points (4,000 in three dimensions) and a learning rate decayed by 0.9 every 2,000 steps. The diffusion–reaction benchmark uses the architecture and budget of Wang et al. (2021). The moving-source problem has no published setup, so PI-DeepONet uses the diffusion–reaction settings, and PI-FNO uses the Burgers configuration of Li et al. (2024) on a 513×513 grid that resolves the heat spot; both are trained on samples of the same Gaussian random field as the test samples. PI-FNO receives the input field on the grid on which it is trained, and PI-DeepONet reads it at 65^2 sensors in two dimensions and 17^3 in three dimensions. To compute the errors, PI-DeepONet is evaluated directly at the quadrature points, and the grid output of PI-FNO is interpolated to them with the same C-HiDeNN basis as INO.

PI-DeepONet is trained for a fixed number of steps, and each step evaluates the PDE residual at randomly sampled points instead of on the mesh. Therefore, its training time is short and does not

Table 3: Rank error with respect to a reference solution of the same discretization, averaged over test samples within the parametric domain. Diffusion uses $\sigma = 0.2$ and a parametric domain of ± 1 standard deviation, and Helmholtz uses $\bar{\kappa}^2 = 10$. Repeated training runs of the same operator differ by up to 1.4×, since the initial guesses of the modes are random.

problem	mesh	S	rank M	unknowns	rank error
2D diffusion	65^2	9	96	47,904	4.035×10^{-4}
2D diffusion	513^2	9	96	133,920	4.599×10^{-4}
2D Helmholtz	65^2	9	64	31,936	4.98×10^{-5}
2D Helmholtz	513^2	9	96	133,920	4.97×10^{-6}
3D diffusion	129^3	8	128	91,520	3.916×10^{-4}
3D Helmholtz	129^3	8	128	91,520	3.736×10^{-4}

depend on the mesh. However, on the moving-source, Helmholtz and three-dimensional problems, its error is larger than that of the mean-input solution. Since its error changes by less than 10% over the last five evaluations, the training budget is not the cause. The larger error comes from two features of its original setup in Wang et al. (2021). First, the boundary conditions are imposed only by a penalty term, so they are not satisfied exactly; on two-dimensional diffusion, for example, the error on the boundary is about 12% of the solution maximum. Second, the residual is evaluated at only 100 random interior points per input function, which are sparse in a three-dimensional domain and rarely fall on the narrow heat spot of the moving source (3.7% of the points).

D Additional experiments

D.1 Rank and mesh resolution

In this section, we study how the rank error of INO depends on the number of modes M and on the mesh (Table 3). For two-dimensional diffusion at 65^2 with $S = 9$, the rank error decreases from 2.9×10^{-2} at $M = 4$ to 4.0×10^{-4} at $M = 96$. At $M = 96$, the rank error is between 4.0×10^{-4} and 7.8×10^{-4} on the meshes from 65^2 to 513^2, so it hardly depends on the mesh. In two dimensions, Helmholtz needs fewer modes than diffusion, since its input enters only the $\kappa^2 u$ term and not the Laplacian. In three dimensions, the operators at 129^3 with $S = 8$ and $M = 128$ are trained in 70 s (diffusion) and 40 s (Helmholtz) on one CPU core, and each stores 91,520 numbers. This is 23 times fewer than the $129^3 = 2{,}146{,}689$ nodal values of a single solution on the same mesh, although the operator gives the solution for every input.

D.2 Evaluation on every test sample

Every model is evaluated on every test sample, including the samples outside the parametric domain. For these samples, INO clips the coordinates to the parametric domain; for the nonlinear diffusion problem, which has no bounded parametric domain, the samples that fail the domain check are evaluated without change. Including these samples increases the error of INO by 1.1–7.5×, whereas the error of PI-FNO increases by at most 18% (Table 4).

The domain check uses only the leading coordinates, since the KL modes are scaled by the square roots of their eigenvalues, which decrease rapidly. For the two-dimensional problems ($\ell = 0.3$), the amplitude of the seventh mode is 0.29 times that of the first, and that of the 25th mode is 0.007 times, so the first six of the $S = 25$ modes carry 88% of the variance. Therefore, clipping a trailing coordinate hardly changes the input field, whereas a check of all 25 coordinates would keep only $0.9545^{25} \approx 31\%$ of the samples.

D.3 Training PI-FNO on fine meshes

As shown in Fig. 7, with its published settings, the training loss of PI-FNO decreases steadily at 65^2. At 129^2, it increases after a few thousand steps on diffusion and Helmholtz and hardly decreases on nonlinear diffusion. At 513^2, neither the loss nor the error changes after the first 2,000 steps, and the

Table 4: Relative errors on every test sample, including the samples outside the parametric domain. "mean input" is the error of the mean-input solution. The cells are defined as in Table 1; **bold** marks the most accurate model.

problem (mesh)	mean input	INO	PI-FNO	PI-DeepONet
2D diffusion 65^2	1.32e-1	1.07e-2 (1.01)	**6.06**e-3 (1.43)	5.74e-2 (1.05)
2D diffusion 129^2	1.32e-1	**1.05**e-2 (1.03)	5.48e-2 (1.16)	6.19e-2 (1.13)
2D diffusion 513^2	1.32e-1	**1.05**e-2 (1.02)	1.74e-1 (1.01)	5.61e-2 (1.12)
2D Helmholtz 65^2	7.07e-2	**2.96**e-3 (1.01)	7.29e-3 (1.33)	1.19e-1 (1.47)
2D Helmholtz 129^2	7.07e-2	**2.96**e-3 (1.01)	7.52e-2 (1.60)	1.32e-1 (1.27)
2D Helmholtz 513^2	6.65e-2	**2.80**e-3 (1.02)	9.07e-2 (1.00)	1.20e-1 (1.12)
3D diffusion 65^3	1.13e-1	1.19e-2 (1.02)	**2.14**e-3 (1.48)	2.12e-1 (1.06)
3D diffusion 129^3	1.14e-1	1.41e-2 (1.07)	**3.54**e-3 (3.28)	2.05e-1 (1.06)
3D Helmholtz 65^3	2.99e-2	1.81e-3 (1.02)	**1.23**e-3 (1.78)	4.16e-1 (1.31)
3D Helmholtz 129^3	2.80e-2	1.76e-3 (1.02)	**8.22**e-4 (1.53)	4.20e-1 (1.25)
2D nonlinear diffusion 65^2	9.60e-1	**3.08**e-3 (1.08)	9.13e-3 (2.10)	2.18e-1 (1.06)
2D nonlinear diffusion 129^2	8.91e-1	**2.84**e-3 (1.08)	5.56e-2 (4.11)	2.19e-1 (1.15)

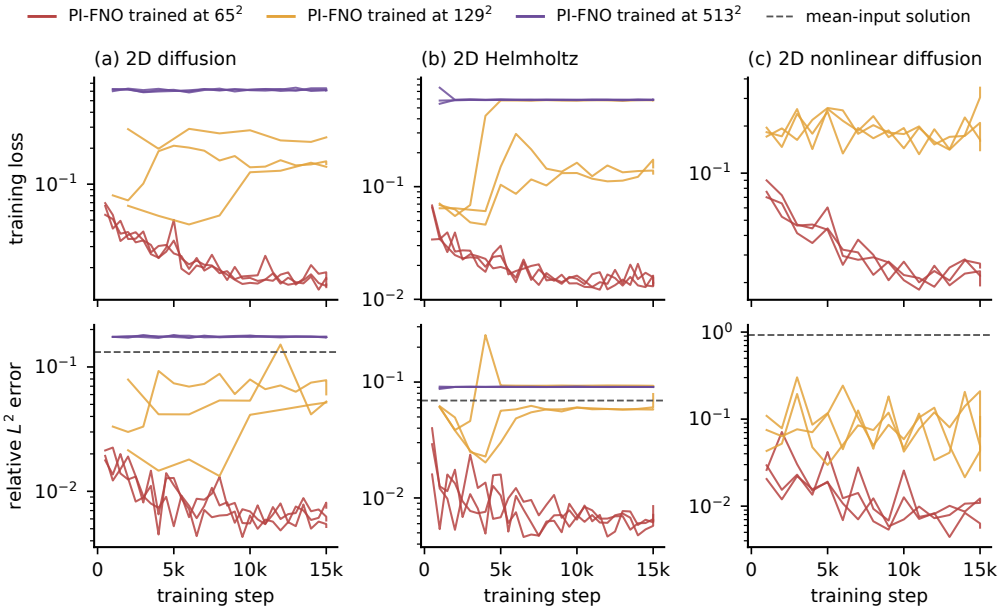


Figure 7: Training histories of PI-FNO on the grid of each problem, for the three initializations of Table 1. Top: training loss. Bottom: relative L^2 error on all test samples. The dashed line is the error of the mean-input solution.

error stays above that of the mean-input solution. For the same initialization, a smaller learning rate of 3×10^{-4} reduces the diffusion error at 129^2 from 6.0×10^{-2} to 2.6×10^{-2}, which is still larger than at 65^2. Since the Fourier neural operator is resolution invariant, the network trained at 65^2 can also be applied on the finer grid. As shown in Table 5, it is then more accurate than the network trained on the fine mesh, and on two-dimensional diffusion it is also more accurate than INO. This use of the network is not included in Tables 1 and 4, where every model is trained on the mesh of the problem for fair comparison.

D.4 Baseline variability

Across three initializations, the error of PI-FNO changes by 1.0–4.5× and that of PI-DeepONet by 1.0–1.9×, whereas that of INO changes by at most 1.4× (Tables 1 and 4). The error of PI-

Table 5: Relative errors of PI-FNO on every test sample when it is trained on the fine mesh and when it is trained at 65^2 and applied on the fine mesh (range over three initializations), compared with INO (geometric mean over three initializations, as in Table 4).

problem	mesh	trained at mesh	trained at 65^2, applied	INO
2D diffusion	129^2	5.18–6.02e-2	2.53–3.98e-3	1.05e-2
2D diffusion	513^2	1.73–1.75e-1	3.14–3.74e-3	1.05e-2
2D Helmholtz	129^2	5.81–9.29e-2	5.13–6.09e-3	2.96e-3
2D Helmholtz	513^2	9.06–9.07e-2	4.06–4.69e-3	2.80e-3
2D nonlinear diffusion	129^2	2.57e-2–1.06e-1	4.02e-3–1.01e-2	2.84e-3

Table 6: Inference cost of INO and of a sparse five-point finite difference solve for two-dimensional diffusion ($S = 9$, $M = 96$, single-threaded medians). The rows with a bound use $M = 64$ and inputs given by their KL coordinates.

	65^2	129^2	257^2	513^2
degrees of freedom	4,225	16,641	66,049	263,169
inference	23 μs	54 μs	169 μs	623 μs
finite-difference solve	11.1 ms	53.3 ms	281.6 ms	1.80 s
ratio	484×	991×	1,662×	2,892×
inference + rank-error bound	0.54 ms	1.61 ms	—	—
inference + total-error bound	5.26 ms	12.5 ms	—	—

FNO also changes by up to 8× over its last five evaluations, so its final checkpoint is noisy. In a separate study of the nonlinear diffusion problem with a stronger nonlinearity ($\beta = 10$), three PI-FNO initializations differ by a factor of 10.9, and the best of them is more accurate than INO. Therefore, we report every result together with its spread.

D.5 Inference cost

Table 6 compares the inference cost of INO with that of a conventional numerical solve. The error bound of Eq. (3) additionally requires one matrix–vector product with $A(k)$ and one evaluation of the dual norm, which is inexpensive since K_0 is diagonal in the eigenbasis of its one-dimensional factors. For a true input field, $A(k)$ is not in separated form, so the field is first written in separated form, and this step dominates the cost of the bound.

D.6 Solving for the coefficients

In Eq. (2), the coefficient of each mode is the product of its parametric factors at the input coordinates, and the error of INO comes from these coefficients (Section 4.3). Since the weak form is a polynomial in $\boldsymbol{\xi}$ and in u, the coefficients can instead be obtained for a given input by solving the weak form projected onto the spatial modes, which is a system of size $r \leq M$, linear for diffusion and cubic for the nonlinear diffusion problem. The projected matrices are computed once after training, so INO remains data-free. The solved coefficients reduce the error by 21.5× at $M = 96$ and 124× at $M = 320$ on two-dimensional diffusion, and by 5.3× on the nonlinear diffusion problem, at an inference cost of 150 μs and 0.5 s, respectively (Table 7).

D.7 Scaling with spatial dimension

In this example, we study how the training cost of INO grows with the spatial dimension d. The parametric problem is fixed, and the spatial dimension is varied from $d = 1$ to $d = 5$ with 33 nodes per direction. The training time increases only from 4.2 s at $d = 2$ to 7.0 s at $d = 5$ (Table 8), while the spatial grid grows by more than four orders of magnitude, since each spatial dimension only adds one one-dimensional solve per sweep over the directions. The ratio of the bound to the true

Table 7: Errors of the same operator evaluated with the product form of Eq. (2) and with solved coefficients. The errors are rank errors for diffusion and total errors for the nonlinear diffusion problem. For diffusion, $S = 9$ and the errors are averaged over 30 samples, which differ from the samples of the text.

problem	product form	solved coefficients	gain	inference time, product / solved
2D diffusion 65^2, M=96	3.21×10^{-4}	1.49×10^{-5}	21.5×	27 μs / 150 μs
2D diffusion 65^2, M=320	9.45×10^{-5}	7.63×10^{-7}	124×	62 μs / 1.08 ms
nonlinear diffusion 65^2	3.08×10^{-3}	5.79×10^{-4}	5.3×	146 μs / 0.5 s

Table 8: Cost versus spatial dimension with $n = 33$ nodes per direction and $M = 64$, with $S = 8$ for $d \geq 2$ and $S = 4$ for $d = 1$, and σ chosen such that $\alpha_\Omega = 0.45$. Five test samples within the parametric domain are used, and all timings are on one CPU core. Errors and bounds are relative energy-norm errors, and "ratio" is the mean ratio of the bound to the error. †No reference solution was computed at $d = 5$.

d	spatial DOF n^d	unknowns	terms T	training time (s)	error	bound	ratio
1	33	12,608	5	2.0	1.09×10^{-4}	1.27×10^{-4}	1.16
2	1,089	25,216	18	4.2	1.71×10^{-3}	2.15×10^{-3}	1.25
3	35,937	27,328	27	5.5	5.21×10^{-3}	6.35×10^{-3}	1.21
4	1,185,921	29,440	36	6.0	2.24×10^{-3}	3.19×10^{-3}	1.42
5	39,135,393	31,552	45	7.0	n/a†	3.03×10^{-3}	—

error stays between 1.16 and 1.42. At $d = 5$, the spatial grid has 3.9×10^7 points and no reference solution was computed, but the bound gives a relative energy-norm error of at most 3.03×10^{-3}.