Neural Operators with Residual-Based Correction: Applications to Optimization and Bayesian Inference

Prashant K. Jha[0000−0003−2158−364X] and
Koffi Enakoutsa[0009−0008−7859−2030]

Abstract This chapter examines the reliability of neural operators as surrogates for parametric partial differential equations in optimization and Bayesian inference. We consider DeepONet, PCANet, and FNO and evaluate their accuracy within and outside the training distribution. Despite small in-distribution prediction errors, the uncorrected DeepONet produces approximately 57% error in the minimizer of the optimization problem. In Bayesian inference, all three operators closely reproduce the finite element result for an in-distribution problem. Under distribution shift, however, errors in the posterior mean increase to 57%–112%, compared with 27% using the finite element model. To improve reliability outside the training distribution, we apply a residual-based correction (RBC) that uses the PDE residual to correct neural operator predictions. RBC substantially improves both optimization and Bayesian-inference results without retraining. These results show that in-distribution accuracy alone does not ensure reliable downstream calculations and that residual correction can provide the necessary physical grounding.

Keywords Neural operators · Residual-based correction · Scientific computing · Optimization · Bayesian inference · Nonlinear diffusion · Elasticity

1 Introduction

Optimization and Bayesian inference often require repeated solutions of the same parametric partial differential equation (PDE). In design optimization, the material

Prashant K. Jha (corresponding author)
Department of Mechanical Engineering, South Dakota School of Mines and Technology, Rapid City, SD 57701, USA, e-mail: prashant.jha@sdsmt.edu; alternate email: pjha.sci@gmail.com

Koffi Enakoutsa
Department of Mathematics, University of California, Los Angeles, Los Angeles, CA 90095, USA, e-mail: koffi@math.ucla.edu

or geometric parameters change as the objective is reduced subject to physical and manufacturing constraints. In Bayesian inference, the forward problem is evaluated for a sequence of proposed parameter fields so that the posterior distribution can be sampled. Similar repeated evaluations arise in uncertainty quantification, control, experimental design, and digital-twin calculations. Classical numerical methods provide the reference solutions for these problems, but a finite element solve at every design iterate or Markov chain Monte Carlo (MCMC) proposal can make the complete calculation expensive, particularly when the governing PDE is nonlinear.

Neural operators reduce this cost by learning the map from an input function to the corresponding PDE solution. Deep Operator Networks (DeepONets), principal-component-based neural operators, and Fourier neural operators (FNOs) construct this map using different representations of the input and output fields [3, 25, 27]. Once trained, they can evaluate a new parameter field much more rapidly than a complete numerical solve.

The computational advantage of a neural operator is useful only if the surrogate remains sufficiently accurate at the inputs encountered during the downstream calculation. Accuracy is commonly reported as an average over test samples drawn from the same distribution as the training data. Such a test does not quantify the error at an arbitrary input. Analyses of DeepONet and FNO identify contributions from finite-dimensional representation, network approximation, and statistical estimation [22, 24]. Bounds developed specifically for distribution shifts further show that in-distribution accuracy does not imply comparable performance under a different input distribution [2]. Benchmark studies have reached a similar conclusion for changes in PDE parameters, spatial resolution, and temporal scales [16, 30]. Section 6 of [20] evaluates the three neural operators used in this chapter for forward accuracy under distribution shifts and finds that the error can increase by an order of magnitude or more.

This limitation is particularly important in optimization and Bayesian inference because the inputs evaluated during these calculations need not resemble independent samples from the training distribution. An optimizer deliberately searches for parameter fields that reduce the objective and may generate spatial features that are poorly represented in the training data. A Bayesian posterior can similarly concentrate in a region that received little training coverage. The mismatch is more direct when the inference prior or the true parameter field differs from the distribution used to train the operator. In either case, a small average testing error does not establish that the surrogate is accurate where the minimizer or posterior is determined.

Errors in the predicted state can also affect a downstream result more strongly than a standalone prediction test suggests. In optimization, parameter-dependent errors alter the objective and its sensitivities and can redirect the optimization path. In Bayesian inference, errors in the predicted observations perturb the likelihood and therefore the posterior distribution. Neural-operator accuracy must consequently be assessed not only through forward predictions, but also through the quantities ultimately computed using those predictions. This distinction also underlies goal-oriented estimation of model error through its effect on quantities of interest, including estimated parameters [21].

Several approaches have been developed to improve or control this accuracy. Physics-informed neural operators incorporate the PDE residual during training, while derivative-informed neural operators use derivative information relevant to optimization

and inference [29, 31]. Multilevel and multistage methods train successive networks to resolve errors left by earlier approximations [1, 32]. Multifidelity methods combine comparatively inexpensive low-fidelity data with a smaller amount of high-fidelity data [17, 28]. For Bayesian inverse problems, adaptive operator learning can refine a pre-trained surrogate in regions explored by the posterior [14]. A posteriori approaches instead estimate or reduce the error of a particular prediction after training [4, 11, 19]. These methods differ in whether the additional cost is incurred during data generation, training, or online evaluation.

Two companion studies motivate the correction developed here. Enakoutsa [8] reads PDE operator learning from finite, noisy data as a projected linear–Gaussian inverse problem, proving posterior well-posedness, an error decomposition into discretization, approximation, and statistical terms, and posterior contraction, while showing across advection, Burgers, Darcy, and Helmholtz benchmarks that the resulting predictive variance is coefficient-driven and largely global rather than spatially adaptive. Enakoutsa *et al.* [10] applies Bayesian operator learning to nonlocal gradient-damage mechanics, using a Monte-Carlo-dropout surrogate to show that field-level uncertainty concentrates near active damage zones, but that the internal length scale can remain practically nonidentifiable from global reaction data in the pre-critical regime because the Fisher information associated with it is small. Together these studies indicate that neither a global predictive-uncertainty estimate nor an accurate global fit guarantees that a surrogate or forward model is reliable at the specific inputs a downstream calculation visits. This motivates the online, physics-grounded, residual-based correction pursued in the remainder of this chapter.

This chapter focuses on residual-based correction as an online approach. The neural-operator prediction is substituted into the PDE residual, and one linearized problem is solved for the correction [4, 19]. For a nonlinear PDE, this update corresponds to one Newton step initialized by the neural-operator prediction. Under local regularity and stability assumptions, the corrected error is quadratic in the original prediction error. The procedure requires no retraining and can be applied to DeepONet, PCANet, FNO, or another surrogate. Its additional cost is one linearized solve, which can remain substantially smaller than that of a complete nonlinear solution. Residual-based correction thus grounds the surrogate prediction in the governing PDE and can improve its reliability outside the training distribution. However, the correction is local. If the initial prediction lies outside the regime in which the linearization is valid, one correction step may not reduce the error. Its effectiveness therefore requires the neural operator to provide a reasonably accurate initial prediction.

The rest of the article is organized as follows: Section 2 introduces the solution operator, its finite element approximation, the training data, and the three neural operators used in the examples. Residual-based correction is developed in Section 3. The nonlinear diffusion and elasticity problems are presented in Section 4, followed by a comparison of the trained neural operators in Section 5. The optimization and Bayesian calculations are given in Sections 6 and 7. The final sections discuss the computational cost, practical limitations, and conclusions.

1.1 Code availability

The neural-operator implementations are available in the `neural_operators` repository[1] and are documented in [20]. Code for the examples considered in this chapter is available in the `applications_neural_operators` repository.[2]

2 Parametric PDEs and neural operators

Before introducing the individual neural operators, it is useful to distinguish three objects that appear throughout the chapter: the solution operator defined by the PDE, the finite element map used to generate data, and the learned approximation used in downstream calculations. This distinction also clarifies the role of residual correction. The neural operator is trained using finite-dimensional data, while the correction is computed by returning its prediction to the finite element space and evaluating the governing residual there. For a detailed discussion of finite-dimensional representations, singular-value decomposition, and sampling from a Gaussian distribution, we refer to [20, Section 2]. Throughout the chapter, notation from [20, Table 1] is adapted for consistency.

2.1 Solution operator

Let $m \in M$ denote a spatially varying parameter field and $u \in U$ the corresponding solution field, where M and U are appropriate function spaces defined on domains $D_M \subset \mathbb{R}^{q_M}$ and $D_U \subset \mathbb{R}^{q_U}$, respectively. The parametric boundary-value problems considered in this chapter can be written in residual form as

$$R(m, u) = 0 \quad \text{in } U^*, \tag{1}$$

where $R : M \times U \to U^*$ represents the governing equation and boundary conditions. When Equation (1) has a unique solution for every admissible m, it defines the solution operator

$$F : M \to U, \qquad u = F(m). \tag{2}$$

Thus, both the input and output of F are functions. Neural operator learning seeks to approximate this map over a class of admissible parameter fields rather than compute the solution for one prescribed field.

Remark 1 Consider the nonlinear reaction–diffusion problem on the unit interval $D_M = D_U = (0, 1)$,

[1] https://github.com/CEADpx/neural_operators

[2] https://github.com/CEADpx/applications_neural_operators

$$-\frac{\mathrm{d}}{\mathrm{d}x}\left(m(x)\frac{\mathrm{d}u}{\mathrm{d}x}\right)+u^3=f(x),\qquad u(0)=u(1)=0, \tag{3}$$

where f is a prescribed source term. We take $U=H_0^1(0,1)$ and

$$M=\{m\in L^\infty(0,1):m(x)\geq m_{\min}>0 \text{ almost everywhere in } (0,1)\}\,.$$

The residual operator $R : M\times U\to U^*$ is defined using the variational form of Equation (3):

$$\langle R(m,u),v\rangle:=\int_0^1 m(x)\frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x+\int_0^1 u^3v\,\mathrm{d}x-\int_0^1 f(x)v(x)\,\mathrm{d}x\qquad\forall v\in U. \tag{4}$$

Here, we note that due to homogeneous Dirichlet boundary conditions, the space of test functions is the same as the solution space. The solution $u=F(m)$ therefore satisfies

$$\langle R(m,F(m)),v\rangle=0\qquad\forall v\in U.$$

2.2 Finite element map and training data

To generate training data and evaluate neural operator predictions in the governing equations, the parameter and solution fields are represented in finite element spaces $M_h\subset M$ and $U_h\subset U$. Let $\{\phi_i^M\}_{i=1}^{p_M}$ and $\{\phi_j^U\}_{j=1}^{p_U}$ denote bases of these spaces. The discrete fields can then be written as

$$m_h(\boldsymbol{x})=\sum_{i=1}^{p_M}\mathrm{m}_i\phi_i^M(\boldsymbol{x}),\qquad u_h(\boldsymbol{x})=\sum_{j=1}^{p_U}\mathrm{u}_j\phi_j^U(\boldsymbol{x}), \tag{5}$$

where $\mathrm{m}\in\mathbb{R}^{p_M}$ and $\mathrm{u}\in\mathbb{R}^{p_U}$ are the corresponding coefficient vectors. The finite element discretization of the residual problem gives

$$\mathrm{R}(\mathrm{m},\mathrm{u})=0,\qquad[\mathrm{R}(\mathrm{m},\mathrm{u})]_j:=\langle R(m_h,u_h),\phi_j^U\rangle,\quad j=1,\ldots,p_U. \tag{6}$$

Assuming that this system has a unique solution for every admissible m, it defines the finite element map

$$\mathrm{F}:\mathbb{R}^{p_M}\to\mathbb{R}^{p_U},\qquad \mathrm{u}=\mathrm{F}(\mathrm{m}). \tag{7}$$

For a nonlinear PDE, evaluating F generally requires several nonlinear iterations and may also require incremental loading.

The finite element map is used to generate data for approximating the solution operator $F : M\to U$. Although these data are represented by finite-dimensional vectors, the neural operator approximates the map between the function spaces:

$$F_{NOp}(\,\cdot\,;\theta):M\to U,\qquad \widetilde{u}=F_{NOp}(m;\theta), \tag{8}$$

where θ denotes the trainable parameters. For an input m_h, let $\widetilde{u}_h$ denote the representation of $F_{NOp}(m_h;\theta)$ in U_h, with coefficient vector $\widetilde{\mathrm{u}}$. Using the selected finite-dimensional representations, we write

$$\widetilde{\mathrm{u}} = \mathrm{F}_{NOp}(\mathrm{m};\theta). \tag{9}$$

Thus, F generates the finite element solutions used for training, while F_{NOp} gives the corresponding representation of the neural operator prediction.

The training dataset consists of parameter-solution pairs

$$\mathcal{D}_N = \left\{\left(\mathrm{m}^I, \mathrm{u}^I = \mathrm{F}(\mathrm{m}^I)\right)\right\}_{I=1}^{N}. \tag{10}$$

The parameter samples are generated from a prescribed probability measure. In the examples considered here, a Gaussian field w is transformed pointwise to obtain a positive material field:

$$w \sim N(0, C), \qquad m = \alpha_m \exp(w) + \beta_m. \tag{11}$$

The covariance operator is $C = L_\Delta^{-2}$, where

$$\begin{aligned} L_\Delta w &= -\mathsf{a}_c \nabla \cdot (\mathsf{b}_c \nabla w) + \mathsf{c}_c w \quad && \text{in } D_M, \\ \mathsf{a}_c \mathsf{b}_c \nabla w \cdot n &= 0 && \text{on } \partial D_M, \end{aligned} \tag{12}$$

where n is the outward unit normal on ∂D_M. The coefficients $(\mathsf{a}_c, \mathsf{b}_c, \mathsf{c}_c)$ control the spatial variation of w, while (α_m, β_m) determine the scale and offset of the physical parameter m. Each realization of m is represented in M_h, and the corresponding solution is computed using Equation (7). These quantities, together with the finite element discretization, specify the distribution represented by the training data. Consequently, the terms *in distribution* and *out of distribution* are meaningful only relative to this specification. [20, Section 2.4] provides a detailed discussion of the Gaussian field and its finite-dimensional representation.

Given data, the trainable parameters are obtained by minimizing an empirical loss. A representative form written using the finite-dimensional representations is

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{I=1}^{N} \left\| \mathrm{u}^I - \mathrm{F}_{NOp}(\mathrm{m}^I;\theta) \right\|_{\ell^2}^2. \tag{13}$$

The precise form of the loss depends on the neural operator architecture. DeepONet is evaluated at prescribed output coordinates, PCANet is trained using reduced coordinates, and FNO acts on fields represented on a structured grid. In every case, however, the loss is evaluated over a finite collection of training samples.

Remark 2 For a given parameter field, the predicted solution can be returned to the finite element space and evaluated using the discrete residual:

$$\mathrm{R}(\mathrm{m}, \widetilde{\mathrm{u}}) = \mathrm{R}\big(\mathrm{m}, \mathrm{F}_{NOp}(\mathrm{m};\theta)\big). \tag{14}$$

The finite element solution satisfies $\mathrm{R}(\mathrm{m}, \mathrm{u}) = 0$, whereas a neural operator prediction does not generally satisfy the discrete governing equations. The residual in Equation (14) measures this discrepancy for the particular parameter field under consideration. The correction developed in Section 3 is constructed using this residual and the tangent of the discrete residual with respect to the solution.

2.3 Reduced representations

PCANet approximates the finite element map using low-dimensional coordinates constructed from the training snapshots. Let

$$\overline{\mathrm{m}} = \frac{1}{N}\sum_{I=1}^{N} \mathrm{m}^I, \qquad \overline{\mathrm{u}} = \frac{1}{N}\sum_{I=1}^{N} \mathrm{u}^I \tag{15}$$

denote the sample means. Let $\mathcal{N}_M$ and $\mathcal{N}_U$ denote the affine normalization maps that subtract these means and apply the selected input and output scalings. The normalized samples are

$$\widehat{\mathrm{m}}^I = \mathcal{N}_M(\mathrm{m}^I), \qquad \widehat{\mathrm{u}}^I = \mathcal{N}_U(\mathrm{u}^I). \tag{16}$$

Collecting these vectors as columns gives the snapshot matrices

$$\widehat{\mathrm{X}}_M = \left[\widehat{\mathrm{m}}^1 \cdots \widehat{\mathrm{m}}^N\right] \in \mathbb{R}^{p_M \times N}, \qquad \widehat{\mathrm{X}}_U = \left[\widehat{\mathrm{u}}^1 \cdots \widehat{\mathrm{u}}^N\right] \in \mathbb{R}^{p_U \times N}. \tag{17}$$

Their singular-value decompositions are

$$\widehat{\mathrm{X}}_M = \mathrm{U}_M \mathrm{S}_M \mathrm{V}_M^T, \qquad \widehat{\mathrm{X}}_U = \mathrm{U}_U \mathrm{S}_U \mathrm{V}_U^T. \tag{18}$$

For prescribed reduced dimensions r_M and r_U, let $\mathrm{U}_{M,r} \in \mathbb{R}^{p_M \times r_M}$ and $\mathrm{U}_{U,r} \in \mathbb{R}^{p_U \times r_U}$ contain the leading left singular vectors. The maps from the normalized coefficient spaces to the reduced spaces are

$$\widetilde{\mathrm{P}}_M = \mathrm{U}_{M,r}^T \in \mathbb{R}^{r_M \times p_M}, \qquad \widetilde{\mathrm{P}}_U = \mathrm{U}_{U,r}^T \in \mathbb{R}^{r_U \times p_U}. \tag{19}$$

The reduced coordinates of an input and its corresponding output are therefore

$$\widetilde{\mathrm{m}} = \widetilde{\mathrm{P}}_M \widehat{\mathrm{m}} \in \mathbb{R}^{r_M}, \qquad \widetilde{\mathrm{u}} = \widetilde{\mathrm{P}}_U \widehat{\mathrm{u}} \in \mathbb{R}^{r_U}. \tag{20}$$

PCANet learns the map from $\widetilde{\mathrm{m}}$ to $\widetilde{\mathrm{u}}$. The corresponding full coefficient vectors are reconstructed as

$$\mathrm{m}_r = \mathcal{N}_M^{-1}\left(\widetilde{\mathrm{P}}_M^T \widetilde{\mathrm{m}}\right), \qquad \mathrm{u}_r = \mathcal{N}_U^{-1}\left(\widetilde{\mathrm{P}}_U^T \widetilde{\mathrm{u}}\right). \tag{21}$$

Remark 3 The discarded singular vectors introduce a projection error before the reduced neural network is evaluated. For a normalized output sample, this error is

$$e_{\mathrm{proj},U}(\widehat{\mathrm{u}}) = \frac{\left\| \widehat{\mathrm{u}} - \widetilde{\mathrm{P}}_U^T \widetilde{\mathrm{P}}_U \widehat{\mathrm{u}} \right\|_{\ell^2}}{\|\widehat{\mathrm{u}}\|_{\ell^2}}. \tag{22}$$

Increasing the size of the reduced neural network cannot remove an error caused by an inadequate basis. This distinction becomes particularly important outside the training distribution because the reduced spaces remain fixed and are constructed entirely from the training snapshots.

2.4 Three neural operator architectures: DeepONet, PCANet, and FNO

The three architectures considered here use different representations of the same discrete map Equation (7). Their detailed implementations are given in [20, Section 4]. The descriptions below introduce the constructions and the notation used in the numerical sections.

DeepONet

DeepONet [26, 27] constructs an output by combining two different networks, branch and trunk, that act on the input field and output coordinates, respectively; see schematics Figure 1. The branch network acts as the input-dependent coefficient vector m and the trunk network acts as the basis function values $\phi_i^U(y)$ at specified coordinates y. The output of the neural operator then combines the two outputs to give a prediction of the solution field at the specified coordinates:

$$F_{NOp}(m)(y) = \sum_{i=1}^{N_{br}} \alpha_i(\mathrm{m};\theta_{br})\phi_i^U(y;\theta_{tr}) + b. \tag{23}$$

For a vector-valued output, separate sets of branch coefficients are used for the components of the field. The trunk network retains the coordinate dependence, so the trained model can be evaluated at prescribed locations in the output domain. In the present computations, the complete nodal parameter vector is supplied to the branch network and the trunk network is evaluated at the finite element nodes.

PCANet

PCANet constructs reduced-dimensional representations of the input and output using the SVD-based projectors in Equation (19) and learns the map between these representations [3, 12]. For a given $\mathrm{m} \in \mathbb{R}^{p_M}$, the PCANet approximation of the discrete map can be written as

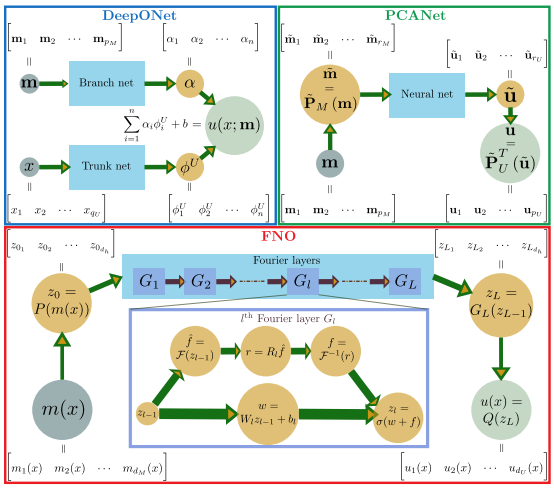


Fig. 1 Schematics of DeepONet, PCANet, and FNO. The trainable map is shown in blue. DeepONet combines input-dependent coefficients with coordinate-dependent basis functions. PCANet learns a map between fixed reduced coordinates. FNO alternates pointwise transformations with truncated Fourier multipliers. Figure based on [20].

$$\mathbb{R}^{p_U} \ni \mathrm{u} = \mathrm{F}(\mathrm{m}) \;\approx\; \mathrm{F}_{NOp}(\mathrm{m};\theta) := \mathcal{N}_U^{-1}\left[\widetilde{\mathrm{P}}_U^T\left(\underbrace{\widetilde{\mathrm{F}}_\theta\left(\widetilde{\mathrm{P}}_M\mathcal{N}_M(\mathrm{m})\right)}_{\text{map between reduced spaces}}\right)\right] \in \mathbb{R}^{p_U}, \tag{24}$$

where

$$\widetilde{\mathrm{F}}_\theta : \mathbb{R}^{r_M} \to \mathbb{R}^{r_U} \tag{25}$$

is a neural network. The normalization maps and SVD-based projectors are constructed from the training data and remain fixed. The only trainable part of PCANet is therefore the map $\widetilde{\mathrm{F}}_\theta$ between the reduced input and output coordinates.

PCANet reduces the dimensions of both the input and output spaces before training the neural network. Its effectiveness therefore depends on the decay of the singular values of the corresponding snapshot matrices. The approximation contains both the error introduced by truncating the input and output representations and the error in learning the reduced map. In particular, the predicted output is restricted to the space spanned by the retained output basis, so increasing the size of $\widetilde{\mathrm{F}}_\theta$ cannot recover components discarded by the output projection.

FNO

FNO considers a composition of layers, where each layer consists of a local affine transformation, a nonlocal integral operator, and a nonlinear activation function [25]. Consider $D_M = D_U = D \subset \mathbb{R}^q$. The FNO approximation of the solution operator is

written as

$$u(x) = F(m)(x) \approx F_{NOp}(m)(x) := Q\left(G_L\left(G_{L-1}\left(\cdots G_1\left(P(m(x))\right)\right)\right)\right), \tag{26}$$

or, more concisely,

$$F_{NOp} = Q \circ G_L \circ G_{L-1} \circ \cdots \circ G_1 \circ P. \tag{27}$$

Here, P lifts the input $m(x)$ to a hidden representation $z_0(x) \in \mathbb{R}^{d_h}$, and Q projects the final hidden representation $z_L(x) \in \mathbb{R}^{d_h}$ onto the output space. The operator G_l defining layer l is given by

$$z_l(x) = G_l(z_{l-1})(x) = \sigma_l\left(W_l z_{l-1}(x) + b_l + K_l(z_{l-1})(x)\right), \qquad l = 1, \ldots, L, \tag{28}$$

where $W_l z_{l-1} + b_l$ is a local affine transformation and K_l is a nonlocal integral operator. In FNO, this operator is defined using the Fourier transform:

$$K_l(z) = \mathcal{F}^{-1}\left(R_l \mathcal{F}(z)\right), \tag{29}$$

where R_l is a trainable complex-valued tensor acting on a prescribed number of retained Fourier modes. The Fourier transform is applied componentwise, the retained modes are transformed using R_l, and the inverse transform returns the result to the physical domain. For more details, including implementation and data generation, we refer to [7, 20].

Next, we describe the residual correction that can be applied to any of the three neural operators as a post-processing step.

3 Residual-based correction

The residual in Equation (14) is more than a measure of whether a neural operator prediction satisfies the governing problem. Evaluated at the prediction, it identifies the governing-equation defect, while the tangent determines the state update needed to eliminate its leading-order contribution. This provides a physics-based correction at inference without modifying or retraining the neural operator.

Let $u = F(m)$ satisfy $R(m, u) = 0$, let $\widetilde{u} = F_{NOp}(m)$, and define the prediction error $e = u - \widetilde{u}$. Expanding the residual about $\widetilde{u}$ gives

$$\begin{aligned} 0 = R(m,u) = R(m,\widetilde{u}) + \delta_u R(m,\widetilde{u})(e) \\ + \int_0^1 (1-s)\delta_u^2 R\big(m, \widetilde{u} + se\big)(e,e)\,\mathrm{d}s, \end{aligned} \tag{30}$$

where $\delta_u R$ and $\delta_u^2 R$ denote the first and second derivatives of R with respect to the state. Retaining the linear terms gives the correction $e^C \in U$:

$$\delta_u R(m,\widetilde{u})(e^C) = -R(m,\widetilde{u}). \tag{31}$$

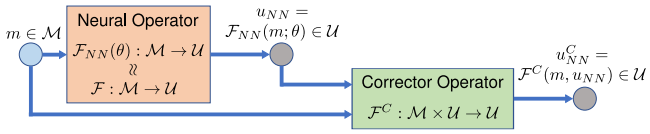


Fig. 2 Predictor-corrector construction. A trained neural operator provides the prediction, and the governing residual and its tangent determine the physics-based correction [4, 19].

The corrected state is therefore

$$\begin{aligned} u^C = F^C(m,\widetilde{u}) &:= \widetilde{u} + e^C \\ &= \widetilde{u} - [\delta_u R(m,\widetilde{u})]^{-1} R(m,\widetilde{u}), \end{aligned} \tag{32}$$

where F^C denotes the corrector operator.

Remark 4 For the nonlinear reaction–diffusion problem introduced in Remark 1, the tangent of the residual at $\widetilde{u} = F_{NOP}(m)$ is

$$\langle \delta_u R(m,\widetilde{u})(w), v\rangle = \int_0^1 m(x)\frac{\mathrm{d}w}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x + \int_0^1 3\widetilde{u}^2 wv\,\mathrm{d}x \qquad \forall w, v \in U. \tag{33}$$

Substitution into Equation (31) shows that the correction $e^C \in U$ satisfies

$$\begin{aligned} \int_0^1 m(x)\frac{\mathrm{d}e^C}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x + \int_0^1 3\widetilde{u}^2 e^C v\,\mathrm{d}x = \int_0^1 f(x)v(x)\,\mathrm{d}x - \int_0^1 m(x)\frac{\mathrm{d}\widetilde{u}}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x \\ - \int_0^1 \widetilde{u}^3 v\,\mathrm{d}x \qquad \forall v \in U. \end{aligned} \tag{34}$$

Thus, although the governing problem is nonlinear, the corrector problem is linear in e^C, with the reaction coefficient $3\widetilde{u}^2$ determined by the neural operator prediction. The corrected state is $u^C = \widetilde{u} + e^C$.

For the finite element problem, let $\widetilde{\mathrm{u}}$ be the coefficient vector associated with $\widetilde{u}$ and define the discrete tangent

$$\mathrm{A}(\mathrm{m},\widetilde{\mathrm{u}}) := D_{\mathrm{u}}\mathrm{R}(\mathrm{m},\widetilde{\mathrm{u}}). \tag{35}$$

The discrete correction is obtained from

$$\mathrm{u}^C = \widetilde{\mathrm{u}} + \mathrm{e}^C, \qquad \text{where} \quad \mathrm{A}(\mathrm{m},\widetilde{\mathrm{u}})\mathrm{e}^C = -\mathrm{R}(\mathrm{m},\widetilde{\mathrm{u}}). \tag{36}$$

The effectiveness of the correction depends on three quantities: the initial prediction error, the stability of the tangent, and the variation of the tangent between the prediction and the exact solution. The following result makes this dependence explicit [19].

Theorem 1 (Quadratic error reduction) *Fix $m \in M$, and let u satisfy $R(m,u) = 0$. Assume that $R(m,\cdot)$ is twice continuously differentiable on a neighborhood containing the segment joining $\widetilde{u}$ and u, and that $\delta_u R(m,\widetilde{u}) : U \to U^*$ is boundedly invertible. Define*

$$
\begin{aligned}
C_{\mathrm{stab}} &:= \left\| [\delta_u R(m, \widetilde{u})]^{-1} \right\|_{\mathcal{L}(U^*;U)}, \\
C_{\mathrm{nl}} &:= \sup_{s\in[0,1]} \left\| \delta_u^2 R\big(m, \widetilde{u} + s(u-\widetilde{u})\big) \right\|_{\mathcal{L}(U\times U;U^*)}.
\end{aligned}
\tag{37}
$$

If $u^C = F^C(m, \widetilde{u})$*, then*

$$
\left\| u - u^C \right\|_U \le \frac{1}{2} C_{\mathrm{stab}} C_{\mathrm{nl}} \left\| u - \widetilde{u} \right\|_U^2 . \tag{38}
$$

Proof. Combining Equations (30) and (31) and using $u - u^C = e - e^C$ gives

$$
\delta_u R(m, \widetilde{u})(u - u^C) = -\int_0^1 (1-s)\delta_u^2 R\big(m, \widetilde{u} + s(u-\widetilde{u})\big)\big(u-\widetilde{u}, u-\widetilde{u}\big)\, \mathrm{d}s. \tag{39}
$$

Applying the inverse tangent, taking norms, and using $\int_0^1 (1-s)\,\mathrm{d}s = 1/2$ yields Equation (38). □

Remark 5 The theorem gives a concrete condition for improvement. If

$$
\frac{1}{2} C_{\mathrm{stab}} C_{\mathrm{nl}} \left\| u - \widetilde{u} \right\|_U < 1, \tag{40}
$$

then the corrected state is guaranteed to be more accurate than the original prediction. The role of the neural operator is therefore to place the prediction within a locally stable neighborhood of the solution. Within this neighborhood, the correction removes the leading-order error and leaves a second-order remainder.

The constants in Equation (37) also identify the limits of the approach. The correction can deteriorate near a singular or poorly conditioned tangent, where C_{stab} becomes large, or in strongly nonlinear regimes, where C_{nl} becomes large. Consequently, the magnitude of the residual alone does not determine the accuracy of the prediction or the effectiveness of the correction. The same residual can correspond to substantially different state errors depending on the stability of the linearized problem.

At the discrete level, Equation (36) is one Newton step initialized by the neural operator prediction. For nonlinear reaction-diffusion and hyperelasticity problems, a reference solution may require several Newton iterations and, in some cases, incremental loading. The correction replaces that nonlinear solution process with one tangent solve. For a linear problem, the second derivative of the residual vanishes, so one correction recovers the finite element solution exactly. However, the correction then requires the same linear solve as the reference problem and provides no computational advantage. Repeated corrections recover the standard Newton iteration and progressively remove the cost advantage of using the neural operator predictor. The construction is independent of the neural operator architecture. Once a DeepONet, PCANet, or FNO prediction is represented in U_h, the same discrete residual R and tangent A determine the correction.

Remark 6 One correction is sufficient in the examples considered here. For the nonlinear diffusion problem, it reduces the mean relative error from approximately 1% to the

order of 10^{-4}%, while the design error in Section 6 falls from 56.86% to 0.05%. If the corrected residual or correction norm remains large, additional Newton steps may be applied; if the correction fails to reduce these quantities or the tangent is poorly conditioned, the reference solver is the safer choice.

4 Model problems and computational setup

Three boundary-value problems are considered. The nonlinear diffusion problem provides a scalar nonlinear setting for demonstrating residual correction and serves as the state equation in the diffusivity-optimization problem. The linear-elasticity and hyperelasticity problems describe a heterogeneous solid with a spatially varying Young's modulus. These two problems use the same domain, loading, material samples, and spatial discretization, allowing the effect of constitutive nonlinearity to be studied within a common computational setting.

4.1 Nonlinear diffusion

The nonlinear diffusion problem considered here and the associated diffusivity-optimization problem in Section 6 are motivated by [19]. Related diffusion–reaction models arise in continuum descriptions of evolving materials, including micropolar models of bone remodeling [23]. Let $D_U \subset \mathbb{R}^2$ be a bounded domain whose boundary is divided into a Dirichlet part Γ_{u_d} and a flux part Γ_{u_q}. The input field $m(x) > 0$ is the diffusivity, and the output $u(x)$ is the temperature. Given m, the temperature satisfies

$$\begin{aligned} -\nabla \cdot (m\nabla u) + \alpha u^3 &= f \quad \text{in } D_U, \\ u &= 0 \quad \text{on } \Gamma_{u_d}, \\ m\nabla u \cdot n &= g \quad \text{on } \Gamma_{u_q}. \end{aligned} \tag{41}$$

The cubic reaction term makes the solution operator nonlinear. With

$$U = \left\{ v \in H^1(D_U) : v = 0 \text{ on } \Gamma_{u_d} \right\}, \tag{42}$$

the residual is defined by

$$\begin{aligned} \langle R(m,u), v \rangle = &\int_{D_U} m\nabla u \cdot \nabla v \, \mathrm{d}x + \int_{D_U} \alpha u^3 v \, \mathrm{d}x \\ &- \int_{D_U} f v \, \mathrm{d}x - \int_{\Gamma_{u_q}} g v \, \mathrm{d}s, \qquad \forall v \in U. \end{aligned} \tag{43}$$

4.2 Linear elasticity

The second model describes a heterogeneous elastic solid whose stiffness varies over the domain. The input $m(x) > 0$ is the Young's modulus, and the output $u(x) \in \mathbb{R}^2$ is the displacement. Let the boundary be divided into Γ_{u_d}, where displacement is prescribed, and Γ_{u_q}, where traction is applied. The balance law and boundary conditions are

$$\begin{aligned} -\nabla \cdot \sigma(u) &= b \quad \text{in } D_U, \\ u &= 0 \quad \text{on } \Gamma_{u_d}, \\ \sigma(u) \cdot n &= t \quad \text{on } \Gamma_{u_q}. \end{aligned} \tag{44}$$

For isotropic linear elasticity,

$$e(u) = \frac{1}{2}\left(\nabla u + \nabla u^T\right), \qquad \sigma(u) = \lambda \text{tr}(e(u))\mathbf{I} + 2\mu e(u), \tag{45}$$

where $\mathbf{I}$ is the identity tensor and the Lamé parameters (λ, μ) depend on m and the fixed Poisson ratio ν through

$$\lambda = \frac{m\nu}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{m}{2(1+\nu)}. \tag{46}$$

With

$$U = \left\{v \in H^1(D_U; \mathbb{R}^2) : v = 0 \text{ on } \Gamma_{u_d}\right\}, \tag{47}$$

the weak form is: given m, find $u \in U$ such that

$$\langle R(m,u), v\rangle = \int_{D_U} \sigma(u) : e(v)\,\mathrm{d}x - \int_{D_U} b \cdot v\,\mathrm{d}x - \int_{\Gamma_{u_q}} t \cdot v\,\mathrm{d}s = 0 \qquad \forall v \in U. \tag{48}$$

4.3 Hyperelasticity

This problem extends the linear elasticity model in Section 4.2 to account for finite deformation. To keep presentation simple, the domain, boundary conditions, and model parameter – spatially varying Young's modulus – are the same as in the linear-elasticity problem. Let

$$\mathbf{F} = \mathbf{I} + \nabla u, \qquad J = \det \mathbf{F}, \qquad I_1 = \text{tr}(\mathbf{F}^T\mathbf{F}). \tag{49}$$

The compressible Neo-Hookean strain-energy density and first Piola-Kirchhoff stress are

$$\begin{aligned} W(\mathbf{F}) &= \frac{\mu}{2}(I_1 - 3 - 2\ln J) + \frac{\lambda}{2}(\ln J)^2, \\ \mathbf{P} &= \frac{\partial W}{\partial \mathbf{F}} = \mu\mathbf{F} + (\lambda \ln J - \mu)\mathbf{F}^{-T}. \end{aligned} \tag{50}$$

Equilibrium is written in the reference configuration as

$$\begin{aligned} -\nabla \cdot \mathbf{P} &= b \quad \text{in } D_U, \\ u &= 0 \quad \text{on } \Gamma_{u_d}, \\ \mathbf{P} \cdot n &= t \quad \text{on } \Gamma_{u_q}, \end{aligned} \tag{51}$$

or, equivalently, in weak form: given m, find $u \in U$ such that

$$\langle R(m,u), v\rangle = \int_{D_U} \mathbf{P} : \nabla v \, \mathrm{d}\boldsymbol{x} - \int_{D_U} b \cdot v \, \mathrm{d}\boldsymbol{x} - \int_{\Gamma_{u_q}} t \cdot v \, \mathrm{d}s = 0 \qquad \forall v \in U. \tag{52}$$

4.4 Computational setup, data generation, and neural-operator configurations

The computational setup for the elasticity problems and the neural-operator configurations follow [20]. For nonlinear diffusion, only DeepONet is trained, using the same architecture and training parameters as the DeepONet for the linear Poisson problem in that work. For each elasticity problem, DeepONet, PCANet, and FNO are trained.

Nonlinear diffusion

The computational domain is the unit square with two circular holes,

$$D_U = (0,1)^2 \setminus \overline{B((0.2,0.8),0.1)} \setminus \overline{B((0.7,0.3),0.2)}. \tag{53}$$

The hole boundaries constitute Γ_{u_d} and are held at zero temperature. The external boundary is Γ_{u_q} and carries the uniform flux $g = 0.1$. The remaining parameters are $\alpha = 1$ and $f = 0$.

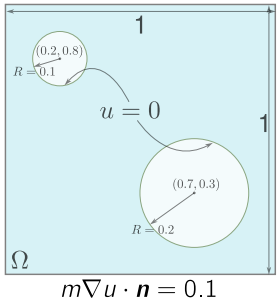


Fig. 3 Domain and boundary conditions for the nonlinear diffusion problem. The temperature is fixed at zero on the boundaries of the two holes, and a uniform flux is applied on the external boundary.

The domain is discretized using 5,025 linear triangular elements and 2,659 vertices. Continuous piecewise-linear interpolation is used for both m and u, giving $p_M = p_U =$

2,659. The diffusivity samples are generated from

$$m = 0.25\exp(w), \qquad w \sim N(0, L_{\Delta}^{-2}), \tag{54}$$

with

$$\mathsf{a}_c = 0.08, \qquad \mathsf{b}_c = 1, \qquad \mathsf{c}_c = 2 \tag{55}$$

in Equation (12). A total of 4,000 parameter-solution pairs are generated, of which 3,200 are used for training and 800 for testing.

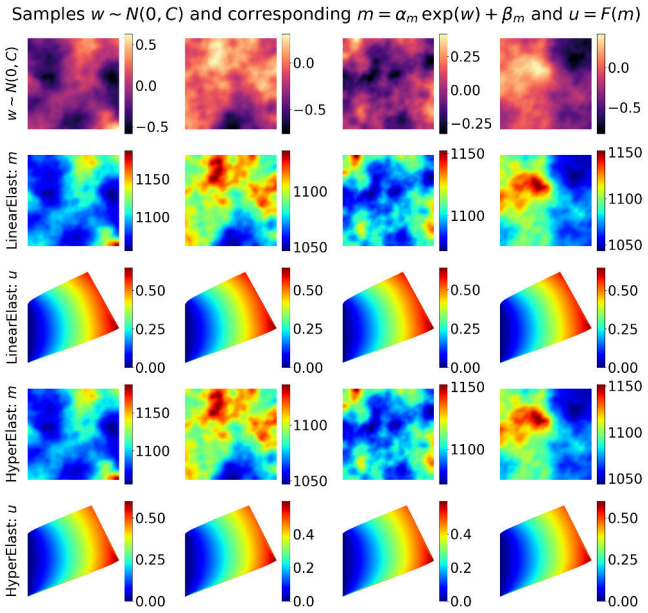


Fig. 4 Representative auxiliary Gaussian fields w, Young's modulus fields m, and solutions u for the linear-elasticity and hyperelasticity problems. The displacement is shown through the deformed configuration, colored by its magnitude.

Linear elasticity and hyperelasticity

Both elasticity problems are posed on $D_U = (0,1)^2$. The left edge is fixed, the body force is zero, and the constant traction

$$t = 20\boldsymbol{e}_1 + 50\boldsymbol{e}_2 \tag{56}$$

is applied on the remaining boundary. The Poisson ratio is $\nu = 0.45$. The Young's modulus samples are generated from

$$m = 100\exp(w) + 1000, \qquad w \sim N(0, L_{\Delta}^{-2}), \tag{57}$$

with $\mathsf{a}_c = 0.005$, $\mathsf{b}_c = 1$, and $\mathsf{c}_c = 0.2$ in Equation (12). The unit square is discretized using 5,000 linear triangular elements and 2,601 vertices. Thus, the scalar modulus vector has dimension $p_M = 2{,}601$, while the two-component displacement vector has dimension $p_U = 5{,}202$. The same 4,000 modulus samples are used for the two elasticity problems, with 3,200 parameter-solution pairs used for training and 800 for testing in each case.

For hyperelasticity, the traction is applied over 20 increments, with a Newton solve at each increment. The neural operators predict the displacement at the final load directly. Residual correction in the Bayesian calculations requires one tangent solve at the final load rather than the complete incremental nonlinear solution.

Neural-operator configurations

The complete training and architecture parameters are summarized in Table 1. For each elasticity problem, DeepONet, PCANet, and FNO have 194,540, 180,700, and 187,508 trainable parameters, respectively. Results are from the neural operator repository [18][3] ([20] discusses the implementation details).

[3] https://github.com/CEADpx/neural_operators

Table 1 Summary of parameters used for neural-operator training and testing. Parameters specific to PCANet and FNO apply only to the elasticity problems.

Parameter	Value	Description
L	3	Number of layers in each DeepONet branch and trunk network, PCANet, and FNO
Width	64	Number of neurons per hidden layer in DeepONet
Width	300	Number of neurons per hidden layer in PCANet
d_h	22	Dimension of the Fourier-layer output
N_{train}	3,200	Number of parameter-solution pairs used for training
N_{test}	800	Number of parameter-solution pairs used for testing
N_m	3,200	Number of input functions used for DeepONet training; $N_m = N_{\text{train}}$
N_x	2,659, 2,601	Number of coordinates used to evaluate the DeepONet output for nonlinear diffusion and elasticity, respectively
p_M, p_U	2,659, 2,659	Discretized input and output function dimensions (nonlinear diffusion)
p_M, p_U	2,601, 5,202	Dimensions of the discretized input and output functions (elasticity)
r_M, r_U	200, 100	Reduced input and output dimensions for PCANet
N_{br}, N_{tr}	100, 100	Numbers of DeepONet branch and trunk outputs (nonlinear diffusion)
N_{br}, N_{tr}	200, 100	Numbers of DeepONet branch and trunk outputs (elasticity)
Batch	20	Number of samples in each training batch
Epochs	1,000	Number of optimization steps for DeepONet and PCANet
Epochs	400	Number of optimization steps for FNO
LR	0.001	Learning rate used during training
Activation	ReLU	Activation function
$k_{\max}$	8	Number of retained modes in each spatial direction
n_1, n_2	51, 51	Number of FNO grid points in each spatial direction

5 Performance of the trained neural operators

The same DeepONet, PCANet, and FNO models are used for forward prediction and for the Bayesian calculations in Section 7. Their performance should therefore be viewed together rather than as unrelated numerical examples. This section first examines the accuracy of the displacement fields when the modulus samples follow the training distribution and when that distribution is changed. The results are then summarized alongside the Bayesian-inference errors obtained with the same trained models. Finally, the nonlinear diffusion problem is used to show the effect of residual correction on individual forward predictions.

Table 2 Performance of DeepONet, PCANet, and FNO on the linear-elasticity and hyperelasticity problems. The prediction errors are mean relative percentage ℓ^2 errors. The Bayesian-inference error is the relative percentage ℓ^2 error in the posterior mean of the latent field w. The finite element row provides the reference inference error. Results from [20].

Forward model	**Quantity**	**Linear elasticity**	**Hyperelasticity**
Finite element	Bayesian-inference error	21.1%	19.7%
DeepONet	Trainable parameters	194,540	194,540
	In-distribution prediction error	0.034%	0.032%
	Prediction error, Case 1	18.064%	14.007%
	Prediction error, Case 2	66.778%	52.654%
	Bayesian-inference error	21.6%	21.9%
PCANet	Trainable parameters	180,700	180,700
	In-distribution prediction error	0.008%	0.007%
	Prediction error, Case 1	17.551%	14.118%
	Prediction error, Case 2	65.929%	52.973%
	Bayesian-inference error	23.8%	21.5%
FNO	Trainable parameters	187,508	187,508
	In-distribution prediction error	0.028%	0.027%
	Prediction error, Case 1	24.900%	13.238%
	Prediction error, Case 2	96.265%	69.212%
	Bayesian-inference error	22.2%	21.4%

For a parameter field m, the relative percentage error in the predicted coefficient vector is

$$e_{NOp}(m) = \frac{\left\|\mathrm{F}(\mathrm{m}) - \mathrm{F}_{NOp}(\mathrm{m})\right\|_{\ell^2}}{\|\mathrm{F}(\mathrm{m})\|_{\ell^2}} \times 100. \tag{58}$$

The in-distribution value reported in Table 2 is the mean over the 800 held-out samples generated from Equation (57). To test the accuracy against out-of-distribution (OOD) inputs, two additional cases are considered, by using the same sampling for w as in (57) but changing the transformation from w to m:

$$\begin{aligned} &\text{Case 1:} \quad (\alpha_m, \beta_m) = (100, 1500), \\ &\text{Case 2:} \quad (\alpha_m, \beta_m) = (1000, 1000). \end{aligned} \tag{59}$$

Comparing the above with the in-distribution case in (57) ($\alpha_m = 100$, $\beta_m = 1000$) shows that the OOD Case 1 shifts the modulus upward without changing the amplitude of its fluctuations, while OOD Case 2 increases the amplitude by a factor of ten. Each out-of-distribution error is averaged over 100 samples.

5.1 Prediction within the training distribution

All three neural operators are highly accurate when the testing samples follow the training distribution. DeepONet and FNO give mean errors near 0.03% for both elasticity problems, while the PCANet errors are below 0.01%. The smooth and strongly correlated modulus and displacement fields admit accurate low-dimensional representations, so the PCANet projectors in Equation (19) discard little of the relevant variation.

Figure 5 shows four representative hyperelasticity samples. The deformed configurations predicted by the three neural operators are visually indistinguishable from the finite element results. This agreement establishes that the architectures and training procedure are adequate for inputs represented by the training data. It does not yet establish their accuracy for the fields encountered during optimization or inference.

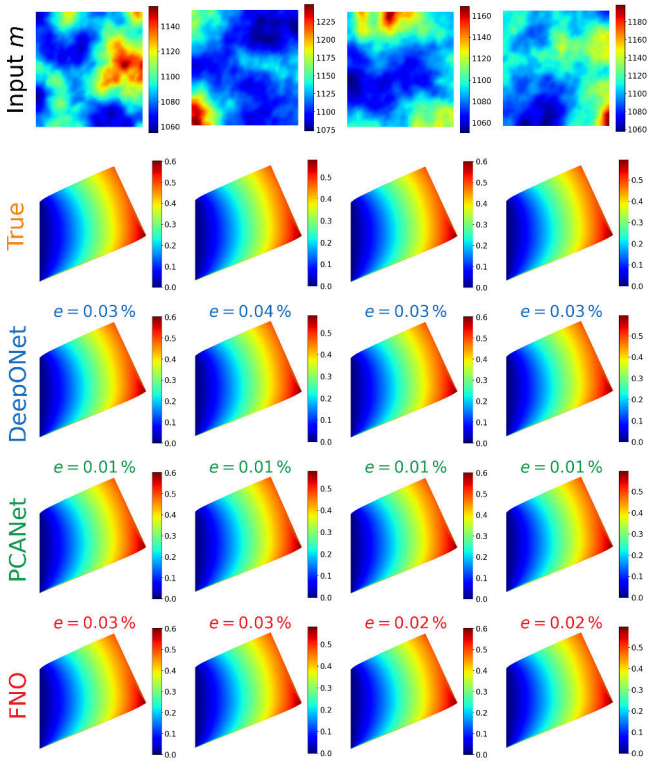


Fig. 5 Finite element and neural-operator solutions for four hyperelasticity inputs drawn from the training distribution. The deformed configurations are colored by displacement magnitude, and e is the relative percentage ℓ^2 error in Equation (58). Results from [20].

5.2 Prediction outside the training distribution (OOD performance)

The out-of-distribution results are substantially different. Under Case 1, the mean errors range from approximately 13% to 25%. Under Case 2, they exceed 50% for every architecture and reach 96.265% for FNO in linear elasticity. The degradation is therefore not peculiar to a learned coordinate basis, a fixed SVD basis, or a Fourier representation. All three models were trained to minimize an average loss over the original material distribution, and none was trained on the shifted fields. Figure 6 shows four representative hyperelasticity predictions for Case 2.

The performance table also includes the Bayesian-inference errors obtained with the same trained models when the inference prior and synthetic truth follow the training distribution. These errors remain close to the finite element result despite the large errors in the separate out-of-distribution tests. The comparison is useful because it shows that a neural operator can be adequate for one downstream calculation and inadequate for another. What matters is the relation between the training distribution and the parameter fields that the calculation actually evaluates. The Bayesian setup and results are discussed in detail in Section 7.

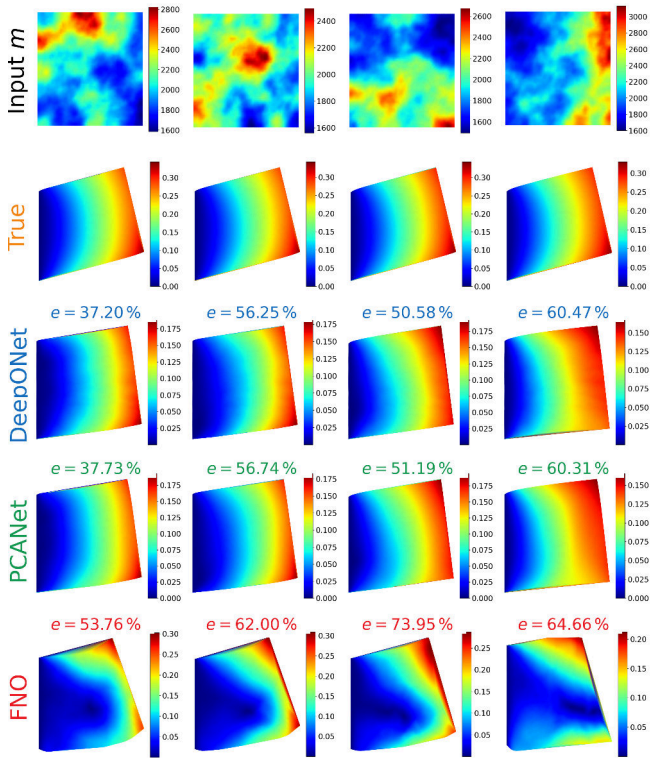


Fig. 6 Representative hyperelasticity predictions for Case 2 in Equation (59). The trained operators are unchanged from Figure 5, but the input distribution has been shifted. Results from [20].

5.3 Residual correction of the nonlinear diffusion predictions

The DeepONet for the nonlinear diffusion problem shows an error of approximately 1% on testing data. Applying one correction Section 3 reduces the mean error to the order of 10^{-4}%. The reduction is consistent with the quadratic estimate in Equation (38): once the DeepONet places the predicted field within the local convergence region, the residual and tangent recover most of the remaining accuracy. Figure 7 shows the finite element solution, the DeepONet prediction, and the residual-corrected prediction for four representative inputs drawn from the training distribution.

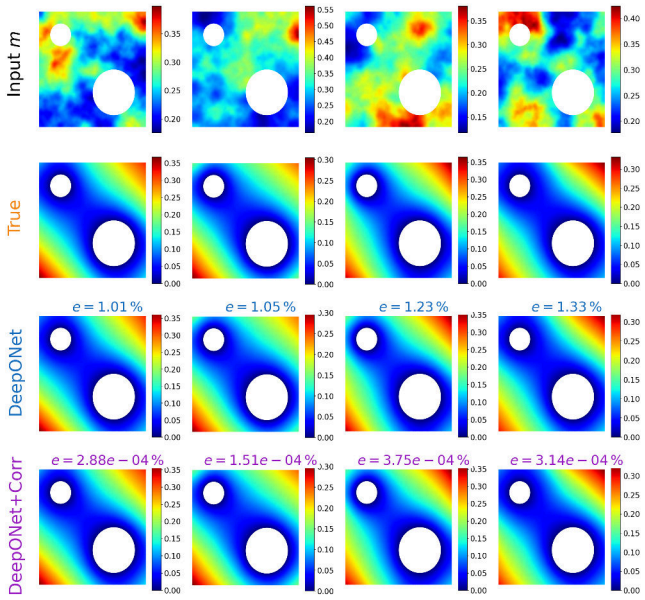


Fig. 7 Finite element solution, DeepONet prediction, and residual-corrected prediction for four nonlinear diffusion inputs drawn from the training distribution.

These forward tests provide the necessary baseline for the two applications that follow. For the nonlinear diffusion problem, the DeepONet is evaluated using a PDE-constrained optimization problem. For elasticity problems, Bayesian inference is uesd to test the performance of neural operators and we report main results from [20, Section 5]. The inference problem in [20, Section 5] can be considered in-distribution. Restricting to hyperelasticity problem, we also consider a shifted inference problem with ground truth and prior that are different from the training distribution, to test how the neural operators perform on an inference problem that is out-of-distribution. The shifted inference problem is described in Section 7. In this case, neural operators show large errors and residual correction is shown to improve the performance significantly.

Through the applications in the next two sections, we highlight both the strengths and limitations of neural operators.

6 Optimization of a diffusivity field

We consider the design of a diffusivity field for the nonlinear diffusion problem in Section 4.1. The objective is to distribute a prescribed amount of additional diffusivity so as to minimize the external thermal work. The optimization is performed using the finite element solution, the DeepONet prediction, and the residual-corrected DeepONet prediction as three alternative state evaluations. Material-distribution optimization also arises in other settings, including topology optimization of lattice metamaterials [15] and joint material–structural optimization of hard-magnetic soft materials [13]. In such iterative calculations, a neural-operator surrogate may be evaluated at parameter fields outside its training distribution.

6.1 Design problem and numerical procedure

The state equation is Equation (41) on the perforated domain in Equation (53). The two hole boundaries are held at zero temperature, and a uniform flux $g = 0.1$ is prescribed on the external boundary, as shown in Figure 3. The objective is the external thermal work

$$J(m) = \int_{\Gamma_{uq}} g\, u(m)\, \mathrm{d}S, \qquad u(m) = F(m). \tag{60}$$

Let v denote the design field. The physical diffusivity is defined as

$$m = m_0 + \widetilde{v}, \qquad \widetilde{v} = H(v), \tag{61}$$

where H is the Helmholtz filter obtained by solving

$$\begin{aligned} -r^2\nabla^2\widetilde{v} + \widetilde{v} &= v \quad \text{in } D_U, \\ \nabla\widetilde{v}\cdot n &= 0 \quad \text{on } \partial D_U. \end{aligned} \tag{62}$$

Here, m_0 is the diffusivity of the host material, while $\widetilde{v}$ describes the additional diffusivity distributed within the domain. The filter regularizes the design field and introduces a length scale controlled by r.

The admissible design fields satisfy

$$v_{lw} \le v(x) \le 1, \qquad \frac{1}{|D_U|}\int_{D_U} v\,\mathrm{d}x = \vartheta, \tag{63}$$

where ϑ is the prescribed mean of v. The diffusivity-optimization problem is

$$v^* = \arg\min_{v} J\big(m_0 + H(v)\big) \quad \text{subject to Equations (41) and (63)}, \qquad m^* = m_0 + H(v^*). \tag{64}$$

The parameters are

$$m_0 = 0.2, \qquad r = 0.012, \qquad v_{lw} = 0.001, \qquad \vartheta = 0.4. \tag{65}$$

Because the Helmholtz filter in Equation (62) preserves the spatial mean, the mean physical diffusivity is $m_0 + \vartheta$.

Three optimization calculations are performed. The reference calculation uses the finite element solution $F(m)$, the second uses the DeepONet prediction $F_{NOp}(m)$, and the third uses the corrected prediction $F^C\left(m, F_{NOp}(m)\right)$ from Equation (32). The corresponding optimized diffusivity fields are denoted by m^*, m^*_{NOp}, and $m^*_{NOp^C}$, respectively.

All three calculations use the same filtered optimality-criteria iteration and begin from the uniform design $v = \vartheta$. At each iteration, the current state u is held fixed while the physical diffusivity m is updated, following the optimality-condition approximation derived in [19, Appendix C]. This gives

$$\frac{dJ}{dm} = -|\nabla u|^2. \tag{66}$$

No derivatives of the DeepONet are needed. The three calculations differ only in the state used to evaluate this expression: the finite element solution, the DeepONet prediction, or the corrected DeepONet prediction. The sensitivity is mapped from the physical diffusivity to the design field through the Helmholtz filter. Since the filter is self-adjoint, this requires one additional solution of Equation (62). The same bounds, mean constraint, update parameters, and stopping criterion are used in all three calculations.

The DeepONet is trained using the log-normal diffusivity fields in Equation (54). Since the optimization iterates are not sampled from this distribution, the DeepONet is evaluated on out-of-distribution inputs and therefore is expected to make larger errors in solving the optimization problem.

6.2 Results

Let $m^*, u^* = F(m^*)$ denote the optimized diffusivity and corresponding state obtained using the finite element model. Let $m^*_{NOp}, u^*_{NOp} = F_{NOp}(m^*_{NOp})$ denote the optimized diffusivity and corresponding state obtained using the DeepONet, and let $m^*_{NOp^C}, u^*_{NOp^C} = F^C\left(m^*_{NOp^C}, F_{NOp}(m^*_{NOp^C})\right)$ denote the optimized diffusivity and corresponding state obtained using the corrected DeepONet. For $a \in \{NOp, NOp^C\}$, the design error is defined as

$$\varepsilon_a = \frac{\left\|m^* - m^*_a\right\|_{\ell^2}}{\|m^*\|_{\ell^2}} \times 100. \tag{67}$$

The state error is evaluated at the optimized diffusivity produced by the corresponding surrogate:

$$e_a = \frac{\left\|u^* - u^*_a\right\|_{\ell^2}}{\|u^*\|_{\ell^2}} \times 100. \tag{68}$$

The resulting compliance gap for $a \in \{NOp, NOp^C\}$ is defined as

$$\Delta J_a = \frac{|J(m^*) - J(m_a^*)|}{|J(m^*)|} \times 100. \tag{69}$$

The errors are summarized in Table 3, and the optimized diffusivity fields and corresponding states are shown in Figure 8.

Table 3 Design, state, and compliance errors for the DeepONet and corrected DeepONet optimizations relative to the optimization solution obtained using the finite element model.

Method	**Design error** ε_a	**State error** e_a	**Compliance gap** ΔJ_a
DeepONet	56.86%	59.18%	28.79%
Corrected DeepONet	0.05%	0.04%	0.003%

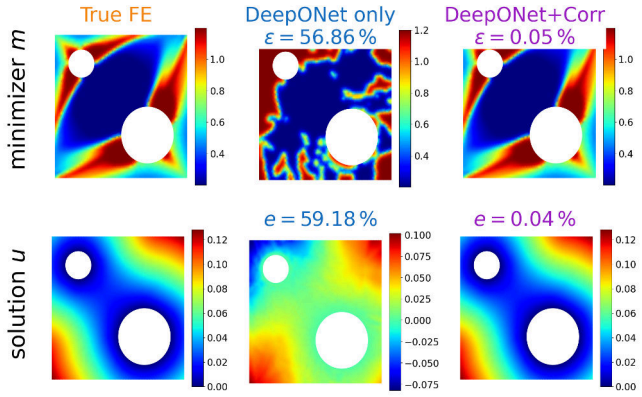


Fig. 8 Top row: optimized diffusivity field obtained using the finite element model, DeepONet, and corrected DeepONet. Bottom row: the corresponding state, evaluated with the same forward map that produced the minimizer above it.

The DeepONet optimization gives a design error of 56.86%. When this design is re-evaluated using the finite element model, its compliance is 28.79% higher than the reference value. The optimized field also contains a noisy, speckled pattern, and the DeepONet state at this design has an error of 59.18% relative to the reference state u^* and includes negative temperatures. These negative values are inconsistent with the prescribed positive flux and zero temperature on the hole boundaries.

The corrected DeepONet gives a substantially different result. Its optimized diffusivity is nearly indistinguishable from the finite element design, with a design error of 0.05%. The corrected state error is 0.04%, and the finite element re-evaluation gives a compliance within 0.003% of the reference value.

The mean error of the DeepONet over the held-out testing fields is approximately 1%, yet its error at the design obtained from the DeepONet optimization is 59.18%. The held-out error therefore does not characterize its performance along the optimization

path. At each iteration, the objective and its derivative determine the next diffusivity field, so errors in these quantities can redirect the subsequent iterates. In the present problem, residual correction provides an accurate state at the diffusivity selected by the optimizer and recovers both the reference design and its compliance.

7 Bayesian inference of Young's modulus

The second application is a Bayesian inference of the Young's modulus field from noisy displacement observations. Because the forward model is evaluated at every MCMC proposal, errors in the predicted displacement directly affect the likelihood and the resulting posterior. We first consider linear elasticity and hyperelasticity when the inference prior and true field are consistent with the neural-operator training distribution. We then repeat the hyperelasticity calculation using a shifted prior and a true field outside the training distribution. The trained neural operators from Section 4.4 are used.

7.1 Bayesian formulation

We first provide concise formulation of the Bayesian inference problem. For more details including algorithm and python implementation, see [20, Section 5]. Let $\mathrm{g} \in \mathbb{R}^{d_{\mathrm{g}}}$ denote the observations and let $\bar{\mathbf{B}} : U \to \mathbb{R}^{d_{\mathrm{g}}}$ be the state-to-observable map. The data model is

$$\mathrm{g} = \bar{\mathbf{B}}(F(m(w))) + \eta, \qquad \eta \sim \mathrm{N}(0, \Gamma_{\mathrm{g}}), \tag{70}$$

where $m(w)$ is the positive transformation in Equation (11). The parameter-to-observable map is

$$\mathbf{B}(w) = \bar{\mathbf{B}}(F(m(w))). \tag{71}$$

For Gaussian noise, the likelihood potential is

$$\Phi(w) = \frac{1}{2}\big(\mathrm{g} - \mathbf{B}(w)\big)^T \Gamma_{\mathrm{g}}^{-1}\big(\mathrm{g} - \mathbf{B}(w)\big). \tag{72}$$

Given the prior measure μ^{o}, the posterior measure μ^{g} satisfies

$$\frac{\mathrm{d}\mu^{\mathrm{g}}}{\mathrm{d}\mu^{\mathrm{o}}}(w) = \frac{1}{Z}\exp\big(-\Phi(w)\big), \tag{73}$$

where Z is the normalization constant.

Replacing F by F_{NOp} gives the approximate observation map $\mathbf{B}_{NOp}$ and potential Φ_{NOp}. Define

$$\mathbf{r}(w) = \mathrm{g} - \mathbf{B}(w), \qquad \delta_{NOp}(w) = \mathbf{B}_{NOp}(w) - \mathbf{B}(w).$$

The error in the likelihood potential is

$$\Phi_{NOp}(w) - \Phi(w) = -\left(\Gamma_{\mathrm{g}}^{-1/2}\mathbf{r}(w), \Gamma_{\mathrm{g}}^{-1/2}\delta_{NOp}(w)\right) + \frac{1}{2}\left\|\delta_{NOp}(w)\right\|_{\Gamma_{\mathrm{g}}^{-1}}^2 . \tag{74}$$

Thus, the effect of the forward error depends on its value at the observation locations, the current data misfit, and the noise covariance. A small error over the full displacement field can still affect the posterior if it is concentrated at the observation locations. Its effect also increases as the assumed observation noise decreases.

Posterior sampling uses the preconditioned Crank–Nicolson method [5, 6]. Given $w^{(k)}$, the proposal is

$$v^{(k)} = \sqrt{1-\beta_{pCN}^2}\, w^{(k)} + \beta_{pCN}\xi, \qquad \xi \sim \mu^{\mathrm{o}}, \tag{75}$$

and is accepted with probability

$$\min\left\{1, \exp\left(\Phi(w^{(k)}) - \Phi(v^{(k)})\right)\right\} . \tag{76}$$

The finite element, neural-operator, and corrected neural-operator models enter the sampling procedure through the likelihood potential.

7.2 Inference using the training prior

The first calculations use the elasticity problems, prior, material transformation, and trained neural operators from [20]. The prior and transformation in Equation (57) are used for both neural-operator training and Bayesian inference. Synthetic displacement observations are collected on a 16×16 grid. Since the displacement has two components, $d_{\mathrm{g}} = 512$. The noise standard deviation is $\sigma_{\mathrm{g}} = 0.0411$, corresponding to 1% of the mean magnitude of the observations. Each chain contains 10,500 samples, of which the first 500 are discarded as burn-in, and $\beta_{pCN} = 0.15$.

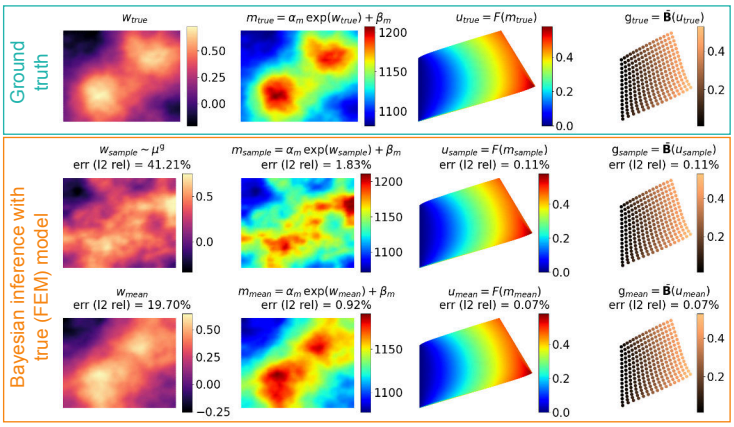


Fig. 9 Hyperelasticity inference using the finite element forward model. The top row contains the true latent field, transformed modulus, displacement, and observations. The lower row contains a posterior sample and the posterior mean with their forward quantities. Results from [20, Section 5].

The errors for the linear-elasticity and hyperelasticity problems are reported in Table 2. For both problems, the posterior errors obtained using the neural operators are within a few percentage points of those obtained using the finite element model. Residual correction is not used in these calculations. The neural operators provide sufficiently accurate forward solutions when the inference prior agrees with the training distribution. The ground truth and posterior sample and mean when forward model is based on the finite element model for the hyperelasticity problem are shown in Figure 9. The Bayesian inference results using the neural operator surrogates are shown in Figure 10.

7.3 Inference using a shifted prior

We next consider a setting in which the prior used for Bayesian inference differs from the prior used to generate the neural-operator training data. The hyperelasticity problem is repeated using a shifted prior for the latent field w and a shifted transformation to the material field m, while the trained DeepONet, PCANet, and FNO remain unchanged. The true field is also constructed using the shifted prior. Consequently, the pCN chains sample material fields that differ from those represented in the training data.

The finite element mesh, constitutive model, loading procedure, observation grid, noise level, pCN parameter, chain length, and burn-in remain unchanged. The neural operators remain those trained using the material transformation $(\alpha_m, \beta_m) = (100, 1000)$.

The shifted prior remains Gaussian, but the covariance of w is changed through the elliptic operator used to define its covariance operator. Specifically, the coefficients a_c and c_c are scaled by the same factor, while b_c remains unchanged. This preserves the ratio $\mathsf{a}_c/\mathsf{c}_c$ and, therefore, the nominal correlation length, while increasing the marginal variance of w by approximately a factor of five. The shifted prior thus produces latent fields with greater amplitude variation without changing their nominal spatial

correlation length. The material transformation used during inference is changed from Equation (57) to

$$m = 100\exp(w) + 1200. \tag{77}$$

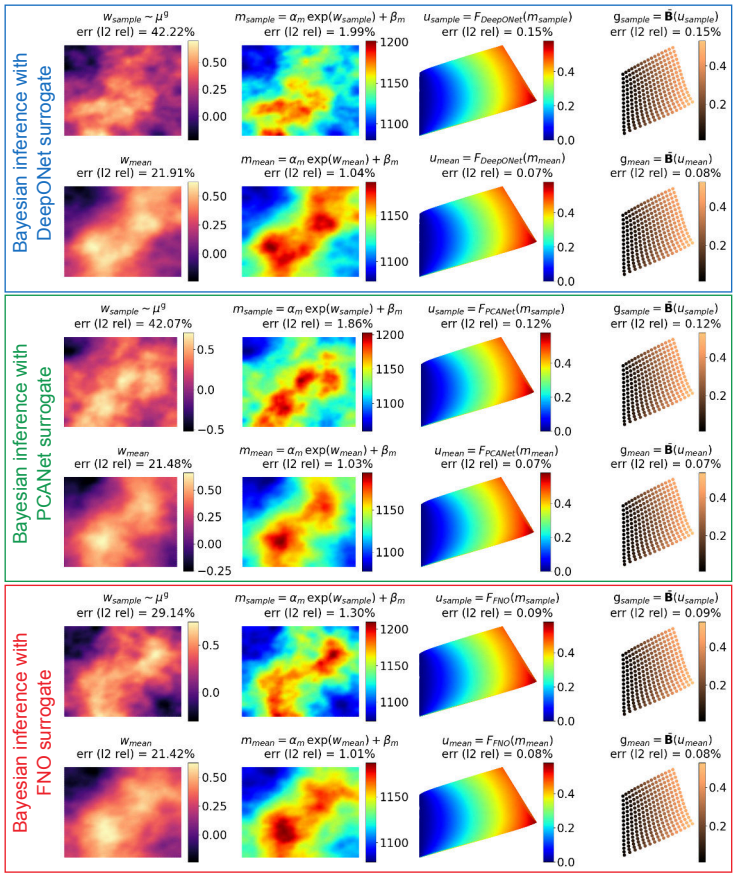


Fig. 10 Hyperelasticity inference using DeepONet, PCANet, and FNO as forward surrogates. The posterior means are close to the finite element result in Figure 9. Results from [20, Section 5].

The true latent field is obtained by amplifying the multimode field used in the first calculation and adding a larger perturbation sampled from the shifted prior. Synthetic observations are generated using the finite element solution for this shifted true field. The same observations, shifted prior, and shifted material transformation are used in all finite element and neural-operator chains. Thus, each forward model is used to solve the same inverse problem, while the neural operators are evaluated on material fields outside their training distribution.

For comparison, the neural-operator posteriors obtained when the inference prior agrees with the training prior are shown in Figure 10.

This setup combines two sources of mismatch that can arise in practice. The proposals are drawn from a prior different from the training measure, and the posterior is informed by data generated from a true field not represented during training. The finite element chain defines the reference posterior for this shifted inverse problem.

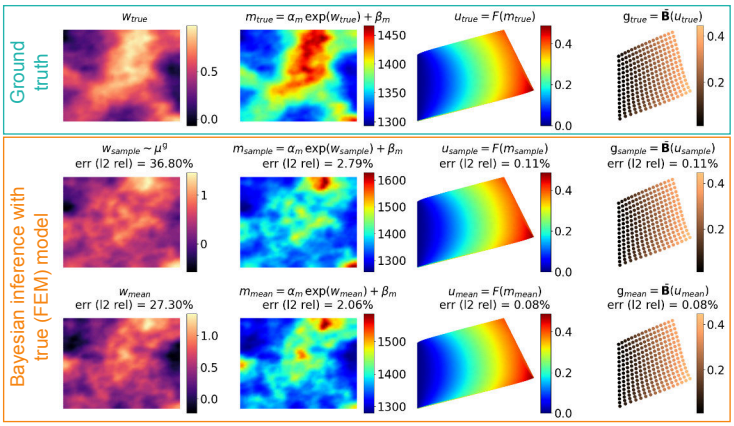


Fig. 11 Shifted hyperelasticity inference using the finite element forward model. The reference chain uses the shifted prior and the same transformation as the synthetic truth.

Results

The raw-surrogate error in the posterior mean of w ranges from 57.2% to 111.7%, compared with 27.3% for the finite element chain; see Table 4. This failure is not visible in the results obtained using the training prior in Section 7.2. It is consistent with the forward distribution-shift results in Table 2. Figure 11 shows the ground truth and the finite element posterior mean, while Figure 12 shows the raw-surrogate posterior means. The raw-surrogate posteriors depart substantially from the finite element result.

7.4 Residual-corrected inference

To determine whether residual correction can improve inference under this mismatch between the training and inference distributions, one correction is applied to the neural-operator prediction for each proposed field. The corrected observation map is

$$\mathbf{B}^{C}_{NOp}(w) = \bar{\mathbf{B}}\left[\mathcal{F}^{C}\left(m(w), F_{NOp}(m(w))\right)\right]. \tag{78}$$

Replacing $\mathbf{B}_{NOp}$ by $\mathbf{B}^{C}_{NOp}$ in Equation (72) requires one hyperelastic tangent solve for each pCN proposal.

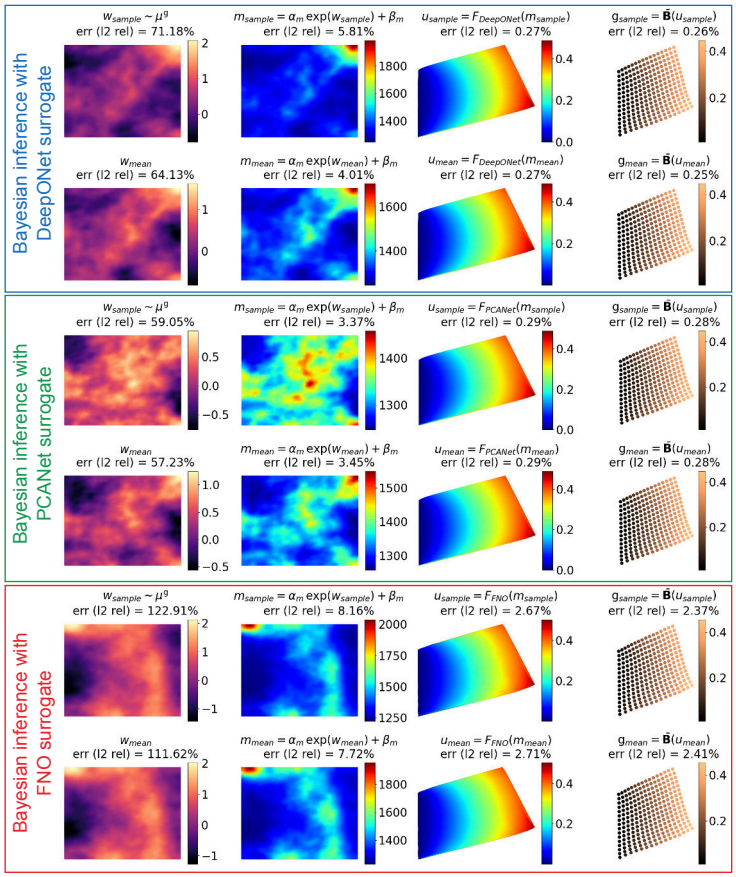


Fig. 12 Shifted hyperelasticity inference using the uncorrected DeepONet, PCANet, and FNO forward models. The posterior means differ substantially from the finite element result in Figure 11.

Let $\overline{\mathrm{w}}$ and $\overline{\mathrm{m}}$ denote the posterior means of the latent and transformed material fields, and let $\mathrm{w}_{\mathrm{true}}$ and $\mathrm{m}_{\mathrm{true}}$ denote their true values. The relative errors are

$$e_w = \frac{\|\overline{\mathrm{w}} - \mathrm{w}_{\mathrm{true}}\|_{\ell^2}}{\|\mathrm{w}_{\mathrm{true}}\|_{\ell^2}} \times 100, \qquad e_m = \frac{\|\overline{\mathrm{m}} - \mathrm{m}_{\mathrm{true}}\|_{\ell^2}}{\|\mathrm{m}_{\mathrm{true}}\|_{\ell^2}} \times 100. \tag{79}$$

From Table 4, for the finite element calculation, the errors in the posterior means of w and m are 27.3% and 2.06%, respectively. Without correction, the neural-operator errors in w range from 57.2% to 111.7%. Residual correction reduces these errors to 32.0%–38.4% and reduces the corresponding errors in m to 2.36%–2.88%. Relative to the finite element error, correction removes approximately 72%, 84%, and 87% of the additional error in w obtained using DeepONet, PCANet, and FNO, respectively.

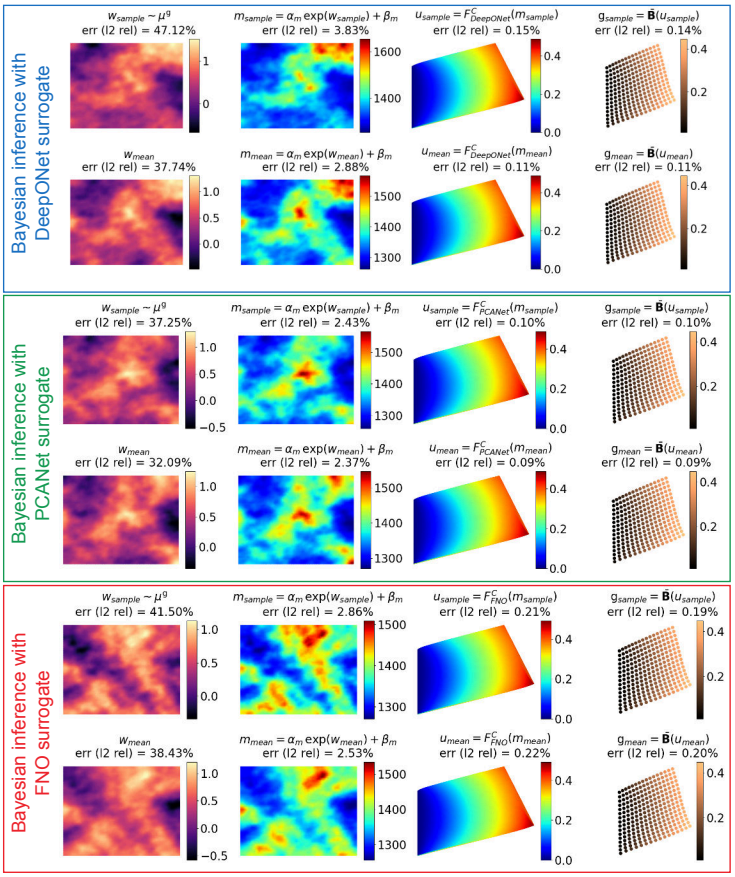


Fig. 13 Shifted hyperelasticity inference using one residual correction in each likelihood evaluation. The corrected posterior means are substantially closer to the finite element result.

Table 4 Relative percentage ℓ^2 errors in the posterior means for the shifted hyperelasticity problem.

	Without correction		With correction	
Forward model	**Error in w**	**Error in m**	**Error in w**	**Error in m**
Finite element	27.3	2.06	–	–
DeepONet	64.1	4.02	37.7	2.88
PCANet	57.2	3.44	32.0	2.36
FNO	111.7	7.72	38.4	2.52

One correction does not completely recover the finite element posterior, but it consistently reduces the additional error introduced by all three neural operators. The finite

element posterior itself differs from the true field because the parameter is inferred from limited and noisy displacement observations.

These results are based on one shifted prior, one true field, and one realization of the observation noise. For this problem, residual correction substantially improves the posterior when the prior used for inference differs from the prior represented in the neural-operator training data.

8 Practical considerations and limitations

The optimization and Bayesian examples show that residual correction can prevent surrogate error from dominating a downstream result. Its role is specific. It reduces the inconsistency between a predicted state and the discrete PDE at the parameter currently being evaluated. It does not address every source of numerical, statistical, or modeling error.

In particular, correction does not refine the finite element mesh or change the approximation space used to generate the reference data. It cannot remove discretization error, repair an incorrect constitutive model, or compensate for an incorrect boundary condition or source term. It also does not remove uncertainty arising from noisy and incomplete training data. The projected operator-learning framework of [8] separates discretization, approximation, and statistical contributions in a Bayesian setting. A residual corrector could be applied after such an operator estimate, but it would enforce the residual of the adopted PDE rather than eliminate uncertainty in the estimated operator or discrepancy between that PDE and the data-generating process.

The size of the residual must also be considered together with the stability of the tangent problem. Formally,

$$u - \tilde{u} \approx - \left[\delta_u R(m, \tilde{u})\right]^{-1} R(m, \tilde{u}). \tag{80}$$

A small residual need not imply a small state error when the inverse tangent has a large norm. Near loss of ellipticity, bifurcation, or another poorly conditioned regime, the correction norm and tangent stability are therefore more informative than the residual alone.

Forward accuracy must likewise be separated from parameter identifiability. As discussed in Section 7, improving the accuracy of the surrogate likelihood cannot make uninformative observations identify a material parameter. Residual correction improves the state used to evaluate the likelihood, but it does not increase the information contained in the observations.

Before a neural operator is used in optimization or inference, its validation set should include parameter fields representative of the intended calculation rather than only additional samples from the training distribution. For PCANet, the input and output projection errors should be reported separately from the error of the reduced neural network. During the downstream calculation, the residual norm, correction norm, tangent conditioning, and physical admissibility of the predicted field provide complementary

Table 5 Forward-evaluation choices within a downstream calculation.

Mode	Appropriate use	Cost	Relevant checks
Neural operator	Inputs are represented by validation data and the observed error is acceptable	One network evaluation	Distribution shift, observable error, physical admissibility
Corrected neural operator	Local accuracy is important and one tangent solve is cheaper than the nonlinear reference solve	Network evaluation, residual and tangent assembly, one linear solve	Correction norm, corrected residual, tangent stability
Reference solver	Correction checks fail or final verification is required	Complete linear or nonlinear solution	Numerical convergence and discretization error

checks. A reference solve remains appropriate when these checks fail and for final verification of an optimized design or posterior summary.

8.1 Computational cost

The computational value of correction depends on the cost of one tangent solve relative to the complete forward problem. The three available evaluation modes are summarized in Table 5.

Let C_{NOp} be the cost of one neural operator evaluation, and C_{asm} and C_{lin} the costs of assembling and solving the linearized system Equation (36). A corrected evaluation costs

$$C_{RBC} \approx C_{NOp} + C_{asm} + C_{lin}. \tag{81}$$

By comparison, a nonlinear finite element solution repeats the assembly and linear solve at every Newton iteration and, when incremental loading is used, at every load increment. Writing N_{nl} for the total number of linear solves,

$$C_{FE} \approx N_{nl} \left(C_{asm} + C_{lin} \right), \tag{82}$$

When the network-evaluation cost is small and the individual assembly and solve costs are comparable, the resulting speedup is approximately

$$\frac{C_{FE}}{C_{RBC}} \approx N_{nl}. \tag{83}$$

Thus, the speedup scales with the number of linear solves required by the reference calculation. For a linear problem, $N_{nl} = 1$, and correction provides little computational advantage. For the hyperelasticity problem, one correction replaces the repeated Newton solves over the load increments with one tangent solve. The actual speedup depends

Table 6 Computational cost for the hyperelasticity problem with the DeepONet surrogate. Online times are means over 20 parameter samples using one Apple M4 Pro CPU core.

Quantity	Value	Description
Online cost per evaluation, seconds		
C_{NOp}	0.000861	Neural-operator evaluation
C_{asm}	0.00224	Residual and tangent assembly, measured in place
C_{lin}	0.00552	Linear solve including LU factorization, measured in place
C_{RBC}	0.00862	Corrected evaluation, measured end to end
C_{FE}	0.605	Reference nonlinear solve, 20 load increments
N_{nl}	80	Linear solves per reference evaluation
Offline cost, seconds		
C_{data}	1936	Estimate for 3200 training pairs: $3200\,C_{FE}$
C_{train}	887	Measured training time, 1000 epochs
Speedup relative to the reference model		
C_{FE}/C_{NOp}	703	Online, uncorrected surrogate
C_{FE}/C_{RBC}	70.2	Online, corrected surrogate
$S_{NOp}(10{,}500)$	2.24	Amortized estimate, uncorrected
$S_{RBC}(10{,}500)$	2.18	Amortized estimate, corrected

on the nonlinear-solver configuration, the use of warm starts, and the extent to which data-generation and training costs are amortized over repeated evaluations.

For the hyperelasticity problem with DeepONet, the measured cost of one corrected evaluation is 0.00862 s, compared with 0.605 s for the reference solver used in the numerical experiments, giving an online speedup of 70.2. This reference calculation uses 20 load increments and 80 linear solves. With two load increments, the same 20 samples converge in 0.090 s on average, reducing the speedup to about 10. Thus, the measured advantage depends strongly on the reference-solver configuration. Including the data-generation and training costs, define the amortized speedup for evaluation mode $X \in \{NOp, RBC\}$ over n downstream evaluations as

$$S_X(n) = \frac{nC_{FE}}{C_{data} + C_{train} + nC_X}. \tag{84}$$

For the 10,500 evaluations used in Section 7, this gives $S_{NOp}(10{,}500) = 2.24$ without correction and $S_{RBC}(10{,}500) = 2.18$ with correction. The measurements are summarized in Table 6.

8.2 Relation to transfer learning and domain adaptation

Transfer learning adapts a pretrained neural operator to a shifted regime. For example, [33] fine-tunes an FNO-based operator using target data when the domain geometry or

material parameters change. Adaptive operator learning can similarly refine a pretrained surrogate in the region explored by a posterior [14]. These approaches modify the surrogate and require additional target-regime data or training. Residual correction instead leaves the surrogate unchanged and corrects an individual prediction using the PDE residual. Under the local assumptions of Equation (38), it provides quadratic error reduction and retains the residual as a diagnostic. The approaches are complementary: adaptation can improve the initial prediction, while correction reduces its remaining local PDE inconsistency.

The present numerical evidence has clear limits. The examples are restricted to two-dimensional domains and one finite element resolution. The elasticity and Bayesian examples use smooth Gaussian parameter fields, and the Bayesian observations are synthetic. The optimization study considers one Helmholtz-filtered material-field design problem governed by nonlinear diffusion. The shifted Bayesian study uses one change in the prior, one true field, and one noise realization. These results establish the mechanism by which local correction can improve optimization and inference, but they do not provide a broad statistical characterization.

Further work should examine changes in correlation length, non-Gaussian and discontinuous fields, multiple true fields and noise realizations, and observation layouts with different levels of identifiability. Problems near instability require additional care because the tangent may become poorly conditioned. Experimental inference would also require an explicit treatment of model discrepancy. These extensions do not change the role established here: residual correction controls local surrogate inconsistency with the adopted PDE, while other errors require separate analysis.

9 Conclusions

Neural operators are useful in optimization and Bayesian inference because they replace repeated PDE solutions with evaluations of a learned field-to-field map. Their measured accuracy, however, depends on the distribution used to generate the training and testing data. The inputs produced by an optimizer or posterior sampler can differ systematically from this distribution, allowing forward error to alter the downstream result.

This chapter presented and examined residual correction as a local connection between the learned map and the governing PDE. One linearized residual solve gives a Newton-type update to the neural-operator prediction. The method is independent of the learned architecture, requires no retraining, and is most attractive when one tangent solve is substantially cheaper than the complete nonlinear forward calculation.

In the diffusivity-optimization study, the raw DeepONet produced a 56.86% relative error in the optimized diffusivity field and a 28.79% increase in reference-evaluated compliance. One residual correction reduced these quantities to 0.05% and 0.003%, respectively. In Bayesian inference, the uncorrected DeepONet, PCANet, and FNO gave posterior means close to the finite element result when the inference prior and true field were consistent with the neural-operator training distribution. When the prior and true field were shifted outside this distribution, the errors in the posterior mean of w increased to 57.2–111.7%. Applying one hyperelastic correction within each likelihood

evaluation reduced these errors to 32.0–38.4%, compared with 27.3% for the finite element chain.

These results support a predictor-corrector use of neural operators in nonlinear many-query calculations. They do not imply that correction resolves every source of error. Discretization, model form, statistical operator uncertainty, parameter identifiability, and interpretation of learned representations [9] remain separate questions. A reliable workflow therefore combines representative training data, application-specific validation, residual and stability checks, corrected evaluations when they are cost-effective, and reference solutions for final verification.

References

1. Aldirany, Z., Cottereau, R., Laforest, M., Prudhomme, S.: Multi-level neural networks for accurate solutions of boundary-value problems. Computer Methods in Applied Mechanics and Engineering **419**, 116666 (2024)
2. Benitez, J.A.L., Furuya, T., Faucher, F., Kratsios, A., Tricoche, X., de Hoop, M.V.: Out-of-distributional risk bounds for neural operators with applications to the helmholtz equation. Journal of Computational Physics **513**, 113168 (2024)
3. Bhattacharya, K., Hosseini, B., Kovachki, N.B., Stuart, A.M.: Model reduction and neural networks for parametric PDEs. SMAI Journal of Computational Mathematics, Volume 7 (2021)
4. Cao, L., O'Leary-Roseberry, T., Jha, P.K., Oden, J.T., Ghattas, O.: Residual-based error correction for neural operator accelerated infinite-dimensional bayesian inverse problems. Journal of Computational Physics **486**, 112104 (2023)
5. Cotter, S.L., Roberts, G.O., Stuart, A.M., White, D.: MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster. Statistical Science **28**(3), 424 – 446 (2013). DOI 10.1214/13-STS421. URL https://doi.org/10.1214/13-STS421
6. Dashti, M., Stuart, A.M.: The bayesian approach to inverse problems. In: R. Ghanem, D. Higdon, H. Owhadi (eds.) Handbook of Uncertainty Quantification, pp. 311–428. Springer International Publishing, Cham (2017). DOI 10.1007/978-3-319-12385-1_7. URL https://doi.org/10.1007/978-3-319-12385-1_7
7. De Hoop, M., Huang, D.Z., Qian, E., Stuart, A.M.: The cost-accuracy trade-off in operator learning with neural networks. arXiv preprint arXiv:2203.13181 (2022)
8. Enakoutsa, K.: Bayesian inference of PDE solution operators: Projection-based learning, error decomposition, and posterior contraction. Computer Methods in Applied Mechanics and Engineering **461**, 119184 (2026). DOI 10.1016/j.cma.2026.119184
9. Enakoutsa, K., Bogard, P., Jha, P.K.: Do attention maps in neural operators learn the Green's function? (2026). Manuscript in preparation
10. Enakoutsa, K., Li, A., Bogard, P.: Bayesian operator learning for nonlocal damage mechanics and internal length scale identification. Mathematics and Mechanics of Solids (2026). DOI 10.1177/10812865261453597
11. Fanaskov, V., Rudikov, A., Oseledets, I.: Neural functional a posteriori error estimates. arXiv preprint arXiv:2402.05585 (2024)
12. Fresca, S., Manzoni, A.: POD-DL-ROM: Enhancing deep learning–based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. Computer Methods in Applied Mechanics and Engineering **388**, 114181 (2022)
13. Galloway, I., Jha, P.K.: Model-informed joint material–structural optimization of hard-magnetic soft materials. arXiv preprint arXiv:2607.14397 (2026). DOI 10.48550/arXiv.2607.14397
14. Gao, Z., Yan, L., Zhou, T.: Adaptive operator learning for infinite-dimensional bayesian inverse problems. SIAM/ASA Journal on Uncertainty Quantification **12**(4), 1389–1423 (2024)

15. Giorgio, I.: Lattice shells composed of two families of curved Kirchhoff rods: an archetypal example, topology optimization of a cycloidal metamaterial. Continuum Mechanics and Thermodynamics **33**(4), 1063–1082 (2021). DOI 10.1007/s00161-020-00955-4
16. Gupta, J.K., Brandstetter, J.: Towards multi-spatiotemporal-scale generalized pde modeling. arXiv preprint arXiv:2209.15616 (2022)
17. Howard, A.A., Perego, M., Karniadakis, G.E., Stinis, P.: Multifidelity deep operator networks for data-driven and physics-informed problems. Journal of Computational Physics **493**, 112462 (2023)
18. Jha, P.: Ceadpx: neural_operators (2026). DOI 10.5281/zenodo.21197314. URL https://doi.org/10.5281/zenodo.21197314
19. Jha, P.K.: Residual-based error corrector operator to enhance accuracy and reliability of neural operator surrogates of nonlinear variational boundary-value problems. Computer Methods in Applied Mechanics and Engineering **419**, 116595 (2024)
20. Jha, P.K.: From theory to application: A practical introduction to neural operators in scientific computing. Mathematics **14**(13), 2421 (2026). DOI 10.3390/math14132421
21. Jha, P.K., Oden, J.T.: Goal-oriented a-posteriori estimation of model error as an aid to parameter estimation. Journal of Computational Physics **470**, 111575 (2022). DOI 10.1016/j.jcp.2022.111575
22. Kovachki, N., Lanthaler, S., Mishra, S.: On universal approximation and error bounds for fourier neural operators. Journal of Machine Learning Research **22**(290), 1–76 (2021)
23. La Valle, G., Soize, C., Misra, A., Giorgio, I.: A nonlinear micropolar continuum model with diffusion–reaction equation for remodeling of bone with trabecular lattice microarchitecture. Mathematics and Mechanics of Solids p. 10812865251413553 (2026). DOI 10.1177/10812865251413553
24. Lanthaler, S., Mishra, S., Karniadakis, G.E.: Error estimates for DeepONets: A deep learning framework in infinite dimensions. Transactions of Mathematics and its Applications **6**(1), tnac001 (2022)
25. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895 (2020)
26. Lu, L., Jin, P., Karniadakis, G.E.: DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv preprint arXiv:1910.03193 (2019)
27. Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E.: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature machine intelligence **3**(3), 218–229 (2021)
28. Lu, L., Pestourie, R., Johnson, S.G., Romano, G.: Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. Physical Review Research **4**(2), 023210 (2022)
29. O'Leary-Roseberry, T., Chen, P., Villa, U., Ghattas, O.: Derivative-informed neural operator: an efficient framework for high-dimensional parametric derivative learning. Journal of Computational Physics **496**, 112555 (2024)
30. Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., Niepert, M.: PDEBench: An extensive benchmark for scientific machine learning. Advances in neural information processing systems **35**, 1596–1611 (2022)
31. Wang, S., Wang, H., Perdikaris, P.: Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Science advances **7**(40), eabi8605 (2021)
32. Wang, Y., Lai, C.Y.: Multi-stage neural networks: Function approximator of machine precision. Journal of Computational Physics **504**, 112865 (2024)
33. Xu, J., Zhou, Y., Liu, Q., Li, K., Yang, H.: Transfer learning of neural operators for partial differential equations based on sparse network λ-FNO. PLOS ONE **20**(5), e0321154 (2025). DOI 10.1371/journal.pone.0321154