Geometry-Aware Post-Hoc Uncertainty Quantification in Operator Learning

Oriol Vendrell-Gallart **Nima Negarandeh** **Ramin Bostanabad**
Department of Mechanical and Aerospace Engineering
University of California, Irvine
Irvine, CA 92617
{ovendrel,nnegaran,raminb}@uci.edu

Abstract

Neural operators provide fast surrogates for PDEs but their deterministic predictions limit their use in tasks requiring uncertainty quantification (UQ), especially under geometric variability. Existing approaches primarily model uncertainty in network parameters, largely overlooking the geometry-aware representations learned by the operator itself. We propose REEF-GP (Residual on Embedded Features Gaussian Process), a post-hoc UQ framework that fits a GP to the residuals of a frozen neural operator whose internal embeddings define the kernel feature space. Rather than learning a separate feature map, REEF-GP adapts the operator's intrinsic coordinate-feature representations to construct geometry-aware uncertainties. To ensure stability and scalability on unstructured domains, REEF-GP incorporates spectral-normalized projections, heteroscedastic geometry-aware noise, and efficient subset-based training that avoids restrictive low-rank approximations. Across five PDE benchmarks with varying geometries, REEF-GP preserves predictive accuracy while achieving calibrated uncertainty estimates competitive with deep ensembles but at a fraction of their cost. Our approach remains robust under geometric distribution shift, with uncertainty concentrating in physically meaningful regions (e.g., shock fronts). Our results demonstrate that accurate and scalable post-hoc UQ for neural operators can be achieved directly in their learned feature space, offering a practical alternative to parameter-centric approaches. We release our code at: https://github.com/Bostanabad-Research-Group/GP-GeometryAware-OL.

1 Introduction

Neural operators are increasingly used to approximate the solution of partial differential equations (PDEs) in domains such as climatology [1, 2], fluid dynamics [3, 4], plasma physics [5, 6], and solid mechanics [7, 8, 9]. Driven by this growing adoption, recent years have seen substantial advances in their architectural design and training strategies to bridge the accuracy gap with traditional solvers while improving scalability and accommodating PDE solutions over varying geometries. In this regard, Transolver and its extensions [10, 11, 12] stand out as prominent examples that leverage transformer-based backbones to achieve high accuracy, scale effectively to large problems, and natively operate on point clouds or mesh-based datasets.

Despite recent successes, most neural operators are built deterministically and provide point estimates that overlook uncertainties. This limitation hinders their adoption in scientific applications where UQ is critical [13, 14] or in downstream tasks that rely on probabilistic predictions (e.g., adaptive data collection [15]). Existing methods for addressing this issue broadly fall into train-time and post-hoc categories. Train-time approaches include the gold-standard deep ensembles [16], where predictions of multiple independent models are combined together; Bayesian approximations via stochastic regularization techniques such as MC Dropout [17]; and inherently probabilistic models

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

like DINOZAUR [18], which uses a diffusion multiplier to introduce stochasticity to Fourier neural operators (FNOs) [3].

In contrast to train-time UQ, post-hoc methods add probabilistic components to a pretrained model at inference time. This alternative is particularly attractive since training neural operators can be an expensive and resource-intensive process, especially for inherently probabilistic variants. Last-layer Laplace approximation [19] is a popular and generic post-hoc technique that has been recently applied to operator learning [20]. Our method belongs to this post-hoc category but, unlike Laplace approximation, it builds Gaussian processes (GPs) [21] whose mean and kernel functions are constructed around a pretrained neural operator, see Figure 1.

Kernel methods and GPs [22, 23, 24] have recently been used for operator learning and benefit from theoretical guarantees and connections to Bayesian inference [25]. However, they suffer from scalability issues related to dataset size and dimensionality, especially in the case of mesh-based data. Additionally, Bayesian inference in infinite-dimensional spaces is inherently *brittle* [26]: unlike in finite settings, if the assumed prior is even slightly misspecified regarding the operator's regularity, the posterior may converge to an incorrect solution with high confidence rather than contracting to the truth. Consequently, standard approximations unaware of the underlying infinite dimensional nature of the problem introduce inductive biases that are liable to yield uncalibrated predictions. Reliable UQ therefore requires not just approximating the posterior of a fixed model, but actively learning from data the prior that best describes the geometry of the operator [27].

We introduce REEF-GP, a novel post-hoc UQ framework that models the discrepancy of a frozen pretrained neural operator with respect to the true solution as a function of both spatial coordinates and latent geometric features. Our main contributions are as follows:

- We design a geometry-aware deep kernel that leverages the internal representations of a pretrained operator. These hidden layers encode deformed representations of the input geometry and REEF-GP reuses them rather than imposing a prior of its own.
- We develop training and inference procedures based on stochastic subset optimization and product of experts to make REEF-GP practical at the scale of operator learning datasets.
- We demonstrate competitive calibration on five challenging 2D and 3D benchmarks, including settings with geometric distribution shift and extrapolation. We further show that REEF-GP extends to vector-valued outputs.

2 Background and Related Work

2.1 Geometry-Aware Operator Learning

Let $\mathcal{U}$ and $\mathcal{V}$ be two Banach spaces of functions defined on a reference bounded domain $\Omega \subset \mathbb{R}^d$, where d denotes the spatial dimension. Each PDE instance is associated with a continuous geometric descriptor $a \in \mathcal{A}$, which corresponds to a specific physical domain $D_a \subseteq \Omega$. For any input function and geometry pair $(u, a) \in \mathcal{U} \times \mathcal{A}$, the corresponding PDE solution $v \in \mathcal{V}$ is defined on D_a and satisfies the governing equations:

$$\begin{aligned} \mathcal{P}(v, u)(\mathbf{x}) &= 0, \quad \mathbf{x} \in D_a, \\ \mathcal{B}(v, u)(\mathbf{x}) &= 0, \quad \mathbf{x} \in \partial D_a, \end{aligned} \tag{1}$$

where $\mathcal{P}$ and $\mathcal{B}$ denote the differential and boundary operators, respectively. Assuming a unique solution operator $\mathcal{G}^\dagger$ exists, we can write:

$$\mathcal{G}^\dagger : \mathcal{U} \times \mathcal{A} \to \mathcal{V}. \tag{2}$$

Neural Operators. Under this framework, the objective is to learn the infinite-dimensional operator $\mathcal{G}^\dagger$ given some data. To this end, these models construct a parametric operator $\mathcal{G}_\theta$ that approximates $\mathcal{G}^\dagger$ by minimizing the empirical risk over the dataset $\mathcal{D}$ containing M triplets $\{(u_i, a_i, v_i)\}_{i=1}^M$, where $v_i = \mathcal{G}^\dagger(u_i, a_i)$. While $\mathcal{G}^\dagger$ acts on continuous spaces, in practice we only have access to discrete numerical evaluations $\{(\mathbf{u}_i, \mathbf{a}_i, \mathbf{v}_i)\}_{i=1}^M$. For a given instance, the continuous domain is discretized as a mesh or point cloud $\mathbf{a}_i = \{\mathbf{x}_j\}_{j=1}^N \subset D_{a_i}$ and $\mathbf{v}_i = \{v_i(\mathbf{x}_{i,j})\}_{j=1}^N \in \mathbb{R}^N$ collects the corresponding nodal solution values.

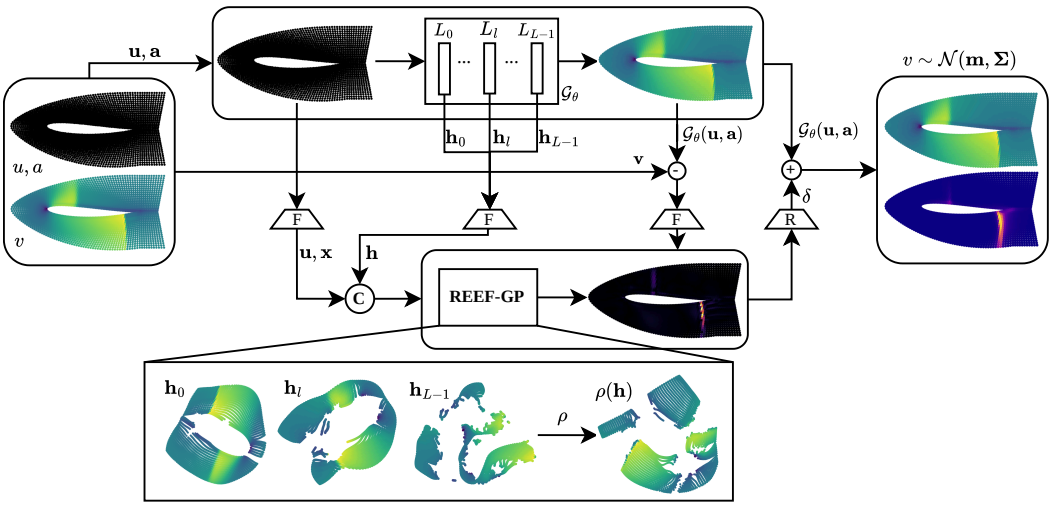

Figure 1: **REEF-GP architecture.** $\mathcal{G}_\theta$ is a frozen neural operator that maps the input functions u, a to the output function v. REEF-GP models $\mathcal{G}_\theta$'s residual discrepancy by operating on the concatenation (C) of $\mathbf{u}, \mathbf{x}, \mathbf{h}$ in the functional regression form (F). Internally, it transforms (ρ) the learned geometry-aware embeddings of $\mathcal{G}_\theta$'s internal layers $(\mathbf{h}_0, \mathbf{h}_l, \mathbf{h}_{L-1})$ to a new space where the kernel operates. In this example, the embedding in the feature-coordinate kernel space tears off the geometry exactly at the shock wave locations. The discrepancy δ is added to the base prediction after recovering (R) the point cloud form to obtain the output distribution.

A large body of work in operator learning has focused on designing architectures that can represent solution operators across discretizations and geometries. Early spectral approaches such as FNOs are particularly effective on regular discretizations [3] but more recent geometry-aware variants accommodate irregular domains, meshes, and point clouds [9, 28, 10]. Transformer-based architectures such as Transolver are especially relevant in this setting as they natively operate on general geometries and preserve spatially aligned hidden states that can later be exploited for UQ.

2.2 Gaussian Processes

GPs [21] provide a principled Bayesian framework for regression, yielding closed-form posteriors with calibrated uncertainty estimates. We apply them to operator learning via a *functional regression* perspective [22]: instead of directly learning $\mathcal{G}^\dagger$, which outputs an infinite-dimensional function, we learn its evaluation functional, so the pointwise evaluation of the operator is represented by a finite-dimensional surrogate

$$\widetilde{\mathcal{G}}^\dagger : \mathbb{R}^{d_u} \times \mathbb{R}^{d_a} \times \Omega \to \mathbb{R}, \qquad (\mathbf{u}, \mathbf{a}, \mathbf{x}) \mapsto v(\mathbf{x}), \tag{3}$$

where $\mathbf{u} \in \mathbb{R}^{d_u}$ and $\mathbf{a} \in \mathbb{R}^{d_a}$ are discrete encodings of the input function u and geometry a, and $\mathbf{x} \in D_a$ is a query coordinate. Operator learning thus reduces to scalar regression over an augmented input $\mathbf{z} = (\mathbf{u}, \mathbf{a}, \mathbf{x})$. The complete derivation is provided in Appendix A. Given M samples each evaluated at N spatial points, the training set contains MN pairs $\{(\mathbf{z}_i, v_i)\}_{i=1}^{MN} = \{\mathbf{Z}, \mathbf{v}\}$, where each $\mathbf{z}_i = (\mathbf{u}_{s(i)}, \mathbf{a}_{s(i)}, \mathbf{x}_i)$ and $s(i)$ identifies the sample to which point i belongs. A GP prior over the latent function $f(\mathbf{z})$ takes the form

$$f(\mathbf{z}) \sim \mathcal{GP}\big(m(\mathbf{z}; \boldsymbol{\beta}),\ k(\mathbf{z}, \mathbf{z}'; \boldsymbol{\phi})\big), \tag{4}$$

with parametric mean and covariance functions $m(\cdot)$ and $k(\cdot, \cdot)$. Given noisy observations $v_i = f(\mathbf{z}_i) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \lambda^2)$, the hyperparameters $\{\boldsymbol{\beta}, \boldsymbol{\phi}, \lambda^2\}$ are typically optimized via maximum likelihood estimation. The posterior mean and covariance at a test input $\mathbf{z}_*$ then admit the closed forms $\bar{m}(\mathbf{z}_*) = m(\mathbf{z}_*) + k(\mathbf{z}_*, \mathbf{Z})(\mathbf{K} + \lambda^2\mathbf{I})^{-1}(\mathbf{v} - m(\mathbf{Z}))$ and $\bar{k}(\mathbf{z}_*, \mathbf{z}'_*) = k(\mathbf{z}_*, \mathbf{z}'_*) - k(\mathbf{z}_*, \mathbf{Z})(\mathbf{K} + \lambda^2\mathbf{I})^{-1}k(\mathbf{Z}, \mathbf{z}'_*)$.

Scalability and Kernel Design. Standard GPs are fundamentally limited by their $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory complexity, making exact inference intractable for operator learning where datasets can easily contain millions of discretization points. Moreover, commonly used scalable GP approximations can introduce structural assumptions that are poorly matched to geometry-varying PDE surrogates. For instance, tensor-product kernels [29, 30] are naturally tied to regular grids and impose separable covariance structure, which can be restrictive for non-stationary and anisotropic solution fields on irregular domains. Likewise, inducing-point approximations such as Nyström [31], SGPR [32], and SVGP [33] rely on low-rank approximations that can limit the GP's flexibility.

Prior GP-based approaches to operator learning involve fixed-geometry settings [23, 22] where the input dimensionality remains moderate and Kronecker structure can be exploited for scalability. These advantages disappear with geometric variability. A natural alternative is deep kernel learning (DKL) which, when combined with scalable approximations, has enabled GPs to address large-scale and high-dimensional problems. However, in settings governed by strong structural constraints (e.g., PDE-driven problems) kernel design is critical for achieving both expressivity and well-calibrated uncertainty [34]. In this work, we address this challenge by leveraging the learnt embeddings of a pretrained neural operator, thus encoding geometry and physics-aware structure directly into the GP.

2.3 Scalable Uncertainty Quantification for Neural Operators

From a Bayesian perspective, UQ in deep networks such as neural operators amounts to characterizing the predictive distribution induced by uncertain model parameters or representations. In principle, this requires marginalizing over a posterior involving millions of parameters. Since this is computationally intractable, practical UQ relies on scalable approximations that are either train-time or post-hoc.

Train-time UQ. These methods consider uncertainties during training, typically via weight-space stochasticity or repeated retraining as in deep ensembles [16]. Computationally efficient alternatives include SWAG [35], which performs approximate Bayesian model averaging using weight trajectories collected during training, and Monte Carlo (MC) dropout [17], where uncertainty is estimated via repeated stochastic forward passes with dropout activated at inference time. These ideas have also been adapted to operator learning; for example, Probabilistic Neural Operator (PNO) [36, 37] uses dropout during training to generate functional samples and optimize an energy-score objective. In broader deep-learning settings, related architectural modifications such as Spectral-normalized GPs [38] have also been proposed to improve uncertainty estimation.

Post-hoc UQ. To preserve the computational efficiency of deterministic training, post-hoc methods augment a frozen pretrained base model for UQ. The most widely used approach is the Laplace approximation [39, 40, 19], which fits a local Gaussian posterior around the maximum a posteriori (MAP) estimate of the weights (for scalability, this strategy is often restricted to the last layer). Laplace approximation has also been explored for neural operators [41, 42], and more recently LUNO [20] applies it to the internal spectral representations and propagates them through a linearized upstream network. Beyond operator learning, Deep Vecchia Ensembles (DVE) [43] move from weight-space to feature-space by fitting a GP on internal neural representations.

While train-time methods benefit from rich epistemic exploration, they are prohibitively expensive for large-scale operator learning on unstructured meshes. In contrast, post-hoc UQ methods are cheaper but beyond DVE they mostly rely on local weight-space approximations instead of fully exploiting the geometry-aware structure encoded in the pretrained operator. These limitations motivate uncertainty models that operate directly in the coordinate and feature space of frozen neural operators.

3 Method: REEF-GP

Most existing UQ methods for neural operators act in weight space, either during training or through post-hoc local approximations. With REEF-GP (Figure 1) we pursue a complementary perspective: rather than perturbing the model parameters, we model the structural inadequacy of a pretrained neural operator directly in the input-coordinate space induced by the operator and its internal representations. More specifically, we approach the problem from the perspective of the Kennedy–O'Hagan (KOH) discrepancy framework [44]. While KOH was originally introduced for Bayesian calibration of computer models and fusing high- and low-fidelity datasets [45, 46], we reformulate it for post-hoc

UQ in operator learning. We treat the pretrained neural operator $\mathcal{G}_\theta$ as a deterministic computer model whose parameters remain strictly frozen. Since $\mathcal{G}_\theta$ is an approximation to $\mathcal{G}^\dagger$, there exists a residual discrepancy between the two which we interpret as *model inadequacy*.

Note that our goal is not to improve the accuracy of $\mathcal{G}_\theta$ but to convert its residuals into a calibrated uncertainty field. Accordingly, REEF-GP preserves the base operator's rL2 by design and does not remove its systematic biases; instead, it exposes them through inflated predictive uncertainty in the regions where they occur (Appendix D.2).

Building on the functional regression view introduced in Section 2.2, we write the true solution evaluated at a spatial query coordinate $\mathbf{x} \in D_a$ as:

$$v(\mathbf{x}) = \mathcal{G}_\theta(\mathbf{u}, \mathbf{a})(\mathbf{x}) + \delta(\mathbf{u}, \mathbf{a}, \mathbf{x}) + \epsilon, \tag{5}$$

where $\delta(\cdot)$ denotes the model inadequacy and ϵ captures stochastic variability or observational noise. In practice, δ is observed only at the training points, and the residual $\delta_i = v(\mathbf{x}_i) - \mathcal{G}_\theta(\mathbf{u}_{s(i)}, \mathbf{a}_{s(i)})(\mathbf{x}_i)$ is obtained with a single forward pass of the frozen operator over the training set (Section 3.3).

Unlike classical KOH formulations which treat the computer model as a black box, we exploit its internal structure. A neural operator can be written as a composition of L latent transformations:

$$\mathcal{G}_\theta(\mathbf{u}, \mathbf{a}) = \mathcal{L}_{L-1} \circ \cdots \circ \mathcal{L}_0(\mathbf{u}, \mathbf{a}). \tag{6}$$

Therefore, $\delta(\cdot)$ is the result of cumulative approximation errors propagated through the network's internal states (this observation is consistent with the representational limits and spectral bias in deep learning [47]). Additionally, recent analyses indicate that compared to the highly constrained output layer, intermediate representations often contain strictly richer and less-compressed information regarding the underlying physics [48, 43]. In the case of operator learning, we demonstrate this *multiscale* representation with a detailed layer-wise analysis in Appendix C, where we also illustrate the layer-wise geometry deformation using t-SNE [49].

Motivated by this multiscale error structure, we enrich the discrepancy model with hidden representations extracted from the frozen $\mathcal{G}_\theta$. In point-based operators such as Transolver [10], each mesh point keeps its own token throughout the transformer blocks, so the output of layer l is a tensor $\mathbf{H}_l \in \mathbb{R}^{N \times D_h}$ whose j-th row corresponds to point $\mathbf{x}_j$. We denote this row by $\mathbf{h}_l(\mathbf{u}, \mathbf{a}, \mathbf{x}_j)$ and call it the *spatially aligned* hidden feature at $\mathbf{x}_j$; we write $\mathbf{h}_l(\mathbf{u}, \mathbf{a}, \mathbf{x})$ for a generic query point $\mathbf{x}$. Operators whose hidden states live on a latent grid or in spectral space require an additional resampling step (Appendix K). We select $L' \leq L$ informative layers (the selection procedure is described in Appendix B.1 with ablations in Appendix D.1) and concatenate them into the augmented state $\mathbf{h}(\mathbf{u}, \mathbf{a}, \mathbf{x}) = \bigoplus_{l \in S} \mathbf{h}_l(\mathbf{u}, \mathbf{a}, \mathbf{x})$ with $S \subseteq \{0, \ldots, L-1\}$. We then refine the discrepancy model by replacing the black-box correction term with a feature-aware alternative defined on the coordinate-feature state:

$$v(\mathbf{x}) = \mathcal{G}_\theta(\mathbf{u}, \mathbf{a})(\mathbf{x}) + \delta(\mathbf{u}, \mathbf{x}, \mathbf{h}(\mathbf{u}, \mathbf{a}, \mathbf{x})) + \epsilon. \tag{7}$$

This yields a discrepancy model conditioned on location and the local regime given by the operator.

3.1 Probabilistic Discrepancy Modeling

To obtain calibrated uncertainties, we model the feature-aware discrepancy probabilistically by first decomposing it into two parts: a latent noiseless residual correction function $f(\cdot)$ and a heteroscedastic noise term $\varepsilon(\cdot)$ that accounts for the unexplained variability:

$$\delta(\mathbf{u}, \mathbf{x}, \mathbf{h}(\mathbf{u}, \mathbf{a}, \mathbf{x})) = f(\mathbf{u}, \mathbf{x}, \mathbf{h}(\mathbf{u}, \mathbf{a}, \mathbf{x})) + \varepsilon(\mathbf{u}, \mathbf{x}, \mathbf{h}(\mathbf{u}, \mathbf{a}, \mathbf{x})). \tag{8}$$

Then, we place a zero-mean GP prior over the residual, that is $f \sim \mathcal{GP}(0, k_{\boldsymbol{\phi}}(\bar{\mathbf{z}}, \bar{\mathbf{z}}'))$ where $\bar{\mathbf{z}}$ is the augmented state $\bar{\mathbf{z}} = (\mathbf{u}, \mathbf{x}, \mathbf{h}(\mathbf{u}, \mathbf{a}, \mathbf{x}))$ and $k_{\boldsymbol{\phi}}$ is the kernel with hyperparameters $\boldsymbol{\phi}$. Next, we jointly model the unresolved error and the independent observational noise as:

$$\varepsilon(\bar{\mathbf{z}}) + \epsilon \sim \mathcal{N}\left(0, \sigma^2_{\boldsymbol{\psi}}(\bar{\mathbf{z}}) + \sigma_n^2\right), \tag{9}$$

where σ_n^2 is a global observation-noise term modeled as $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Although many PDE benchmarks are effectively deterministic, retaining σ_n^2 allows our framework to accommodate stochastic simulators or measurement noise in experimental settings. Moreover, we parametrize $\sigma^2_{\boldsymbol{\psi}}$ by a spectral-normalized [38] dense network that maps $\bar{\mathbf{z}}$ to a strictly positive value. The spectral normalization mitigates feature collapse which, in turn, enables a GP to surrogate $\delta(\cdot)$.

With the independent Gaussian priors and Equations 8 and 9, the marginal prior for the solution value at a query point $\mathbf{x}$ is:

$$v(\mathbf{x}) \sim \mathcal{N}\Big(\mathcal{G}_\theta(\mathbf{u}, \mathbf{a})(\mathbf{x}),\ k_{\boldsymbol{\phi}}(\bar{\mathbf{z}}, \bar{\mathbf{z}}) + \sigma^2_{\boldsymbol{\psi}}(\bar{\mathbf{z}}) + \sigma^2_n\Big). \tag{10}$$

The kernel design, hyperparameter estimation, and inference are detailed in the next sections.

3.2 Kernel Design for Infinite-dimensional Inference

The rationale behind defining the discrepancy δ as a function of the operator's internal state is rooted in a fundamental theoretical challenge: the inherent brittleness of Bayesian inference in infinite-dimensional spaces [26]. In finite-dimensional settings, posterior contraction is often robust to moderate prior misspecification, but in function spaces even mild mismatch between the prior regularity and the target operator can lead to highly confident yet incorrect inference.

To avoid imposing a misspecified prior, we draw inspiration from Kernel Flows (KF) [27] which advocates for learning the kernel geometry from data without the need of relying on physics informed priors [50]. Our key modification is to avoid learning the manifold from scratch: we assume that the pretrained neural operator has already deformed the original input-geometry space into a physics-aware latent representation through its hidden layers. In Appendix C we provide a more formal motivation for this latent geometry and its connection to KF and DKL.

We therefore construct a deep kernel [34] over the augmented state (see Appendix B.2 for architecture details). To respect the distinct topological roles of spatial location, input constraints, and latent operator state, we leverage kernel closure properties and define the covariance as the product kernel:

$$k_{\boldsymbol{\phi}}(\bar{\mathbf{z}}, \bar{\mathbf{z}}') = k_{\text{space}}(\mathbf{x}, \mathbf{x}') \cdot k_{\text{fun}}(\mathbf{u}, \mathbf{u}') \cdot k_{\text{latent}}(\mathbf{h}, \mathbf{h}'). \tag{11}$$

Here, k_{space} acts on physical coordinates, k_{fun} acts on the input function representation when present, and k_{latent} acts on the hidden operator embeddings. We use a product rather than a sum so that space, input function, and latent state are modeled jointly rather than additively: two points are strongly correlated only when they are similar along all factors. Through the latent factor, the resulting kernel is non-stationary in space, allowing the spatial correlation to vary with the local solution behavior. For instance, adjacent points on opposite sides of a shock front are expected to have dissimilar latent states, so the product attenuates their spatial correlation. For the latent component, we further define:

$$k_{\text{latent}}(\mathbf{h}, \mathbf{h}') = k_{\text{base}}(\rho(\mathbf{h}), \rho(\mathbf{h}'))\,, \tag{12}$$

where k_{base} is a stationary kernel and $\rho(\cdot)$ is a warping function parameterized with a spectral-normalized MLP. This construction yields a non-stationary prior in the original coordinate space: spatial correlations are modulated by similarity in the learned latent state, allowing the discrepancy model to adapt to different local physical regimes while remaining anchored to the pretrained operator's internal geometry.

The geometric structure itself originates in the frozen operator and the role of REEF-GP is to turn it into a metric suited to uncertainty rather than to prediction. Three components achieve this: (i) CKA-based layer selection (Appendix B.1), which retains the representations that are jointly most distinct; (ii) the spectral-normalized projection ρ, which maps embeddings shaped by predictive accuracy into a space where kernel distances are meaningful while preventing feature collapse; and (iii) the heteroscedastic noise network, which learns where residual variance is irreducible.

3.3 Scalable Training

Under the functional regression formulation of Section 2.2, the training set contains MN elements. In operator learning MN is often in the order of millions and so exact GP training is infeasible due to $\mathcal{O}((MN)^3)$ time and $\mathcal{O}((MN)^2)$ memory requirements. Hence, again inspired by KF [27], we develop a stochastic subset strategy that jointly optimizes all model hyperparameters. At each optimization step, we randomly sample a subset of $N_s \ll MN$ points and evaluate $\boldsymbol{\delta}$ defined as the N_s-dimensional vector of structural residuals with entries $\delta_i = v(\mathbf{x}_i) - \mathcal{G}_\theta(\mathbf{u}_{s(i)}, \mathbf{a}_{s(i)})(\mathbf{x}_i)$. Then, we update the hyperparameters using mini-batch negative log marginal likelihood as the loss:

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\psi}) = \frac{1}{2}\boldsymbol{\delta}^\top \mathbf{K}^{-1}_{\boldsymbol{\phi}\boldsymbol{\psi}}\boldsymbol{\delta} + \frac{1}{2}\log|\mathbf{K}_{\boldsymbol{\phi}\boldsymbol{\psi}}| + \frac{N_s}{2}\log(2\pi), \tag{13}$$

where $\mathbf{K}_{\phi\psi} = k_\phi(\bar{\mathbf{Z}}, \bar{\mathbf{Z}}) + \mathrm{diag}\Big(\sigma^2_\psi(\bar{\mathbf{Z}})\Big) + \sigma_n^2\mathbf{I}$ is the tractable $N_s \times N_s$ joint covariance matrix evaluated at the augmented states $\bar{\mathbf{Z}}$. By drawing random N_s-sized subsets at each epoch, we strictly decouple the memory constraints from the true mesh resolution N. In our studies, this training procedure combined with spectral normalization and heteroscedastic noise remains stable. The implementation, empirical validation and optimization settings are detailed in Appendix B.4.

We highlight that the KF loss introduced in [27] discards UQ and focuses on building kernel *interpolants* whose accuracy is robust to the training data size. We borrow the notion of robustness behind KF loss and extend it to accommodate UQ. In Appendix C we further elaborate in this regard and detail how REEF-GP differs from standard DKL with inducing points.

3.4 Inference via Generalized Product of Experts

Conditioning the GP on MN training points is infeasible for fast inference. To address this issue, we use generalized Product of Experts (gPoE) [51] where we first randomly draw K support subsets of the data to construct K independent GP experts that share the same hyperparameters. We then aggregate their predictive means and variances in precision space to obtain the posterior of our discrepancy surrogate. Finally, this posterior is added to the frozen base operator's prediction to form the final predictive distribution. This formulation strictly bounds the test-time memory footprint to $\mathcal{O}(N_s^2)$ per expert. Full mathematical details of the gPoE aggregation are provided in Appendix B.5.

4 Experiments

Base Neural Operator. We use Transolver [10] as the base deterministic neural operator in all experiments. Unlike operators with global spectral filters (e.g., FNOs), Transolver preserves pointwise spatial correspondence between blocks. This allows us to extract the spatially-aligned hidden features $\mathbf{h}(\mathbf{u}, \mathbf{a}, \mathbf{x})$ required for our augmented state. By keeping the base architecture fixed, we isolate the effects of our UQ formulation. Hyperparameter configurations are detailed in Appendix E. Appendix K shows that REEF-GP also applies to F-FNO [52] with the learned deformation of Geo-FNO [9], whose hidden states are not point-aligned.

Benchmarks. We use a combination of 2D (*Elasticity, Airfoil*, and *Pipe*) and 3D (*ShapeNet Car* and *Ahmed Body*) datasets involving point clouds. In the main text we present results on the representative 2D *Airfoil* and 3D *ShapeNet Car* benchmarks; full results across all five datasets are reported in Appendix I. Dataset details are included in Appendix F.

Baselines. We compare REEF-GP against six UQ baselines: (1) Deep Ensembles, (2) MC Dropout, (3) PNO, (4) Input Perturbation, (5) Laplace Approximation (LUNO-LA), and (6) Deep Vecchia Ensemble (DVE-spatial). All baselines use the same pretrained Transolver architecture. Implementation details are provided in Appendix G and evaluation metrics are defined in Appendix H.

4.1 Results

Table 1 reports the evaluation metrics on the two representative benchmarks. Results on the remaining three cases are deferred to Appendix I (Table 10) and are consistent with the trends discussed here.

Predictive accuracy is preserved. REEF-GP does not degrade the base operator's predictive accuracy in any of the five datasets (Tables 1, 10), and rL2 remains identical or very similar to the deterministic baseline. LUNO-LA also has this attractive property but Perturbation and all the train-time methods (MC Dropout, PNO, and DVE-spatial) incur substantial rL2 penalties relative to Base. The only method that (expectedly) improves rL2 over Base is Deep Ensembles but at the cost of substantial additional computation.

UQ matches Deep Ensembles on 3D benchmarks. On *ShapeNet Car*, REEF-GP achieves NLL of 2.84, outperforming even Deep Ensembles (4.16) and reducing NLL by an order of magnitude relative to the strongest post-hoc competitor LUNO-LA (33.55). On *Ahmed Body*, our NLL is tied with the Deep Ensembles (Table 10). Regarding the rest of the metrics and datasets, REEF-GP is competitive to either the best or second-best performing baselines. The remaining gap on the 2D benchmarks

Table 1: **Evaluation metrics on representative 2D (Airfoil) and 3D (ShapeNet Car) benchmarks.** Lower is better (↓). Best in **bold**, second best underlined. Base is the deterministic Transolver without UQ; rankings exclude Base. REEF-GP is our approach. Metrics are defined in Appendix H.

Method	**rL2** ↓	**NLL** ↓	**CRPS** ↓	**NIS** ↓	**ES** ↓
Airfoil (2D)	(%)		($\times 10^{-4}$)	($\times 10^{-2}$)	
Base (no UQ)	1.24 ±0.12	–	–	–	–
Ensemble	**1.06**	−3.46	**28.63**	**3.89**	**0.35**
MC Dropout	1.58 ±0.21	−3.38 ±0.15	44.88 ±5.66	5.55 ±0.91	0.55 ±0.08
PNO	2.32 ±1.40	−3.28 ±0.65	74.27 ±56.12	8.26 ±6.39	0.79 ±0.47
Perturbation	3.33 ±0.19	−3.32 ±0.05	81.05 ±5.67	9.71 ±0.66	1.08 ±0.06
LUNO-LA	1.24 ±0.12	−3.07 ±0.82	38.70 ±2.50	6.19 ±0.54	0.47 ±0.04
DVE-spatial	2.09 ±0.25	19.98 ±7.32	88.41 ±12.53	25.89 ±3.09	0.92 ±0.10
REEF-GP	1.24 ±0.12	**−3.51** ±0.06	39.80 ±2.35	5.69 ±0.47	0.45 ±0.05
ShapeNet Car (3D)	(%)				
Base (no UQ)	8.62 ±0.13	–	–	–	–
Ensemble	**7.73**	4.16	**1.88**	30.28	**194.57**
MC Dropout	9.04 ±0.59	17.21 ±4.21	2.52 ±0.26	67.96 ±9.04	267.68 ±17.66
PNO	13.76 ±1.74	4.98 ±1.52	3.53 ±0.50	51.10 ±12.17	337.16 ±43.04
Perturbation	8.62 ±0.13	13.44 ±1.14	2.28 ±0.03	50.82 ±1.10	224.73 ±3.97
LUNO-LA	8.62 ±0.13	33.55 ±23.43	2.47 ±0.13	73.47 ±14.91	271.32 ±12.10
DVE-spatial	13.08 ±0.26	11.22 ±3.03	3.76 ±0.10	96.17 ±10.43	396.70 ±6.79
REEF-GP	8.52 ±0.14	**2.84** ±0.09	2.05 ±0.03	**28.81** ±1.65	224.31 ±4.18

and on CRPS, NIS, and ES is expected, since an ensemble combines several independently trained operators at five times the training cost. Fitting REEF-GP to each ensemble member closes this gap on *ShapeNet Car*, and it matches or outperforms Deep Ensemble on all five metrics (Appendix M).

Spatially coherent uncertainty maps. Aggregate metrics can be inconclusive when best values shift across datasets and scoring rules. In geometric operator learning, spatial consistency between predicted uncertainty and prediction error is itself a meaningful signal. Figure 2 shows per-method standard deviation fields on *Airfoil* and *ShapeNet Car* alongside the base operator's prediction and error. Our uncertainty concentrates around the shock fronts on *Airfoil* and the front bumper region on *Car*, closely tracking where the operator's error is largest. Calibration is therefore not just numerically tight but spatially meaningful.

Cost-quality tradeoff. Figure 3 compares compute cost across all UQ methods on *Airfoil* and *ShapeNet Car*. REEF-GP achieves training overhead an order of magnitude lower than Deep Ensembles and competitive with all post-hoc baselines. Eval time and train memory are competitive across methods, with REEF-GP sitting close to the median. The only tradeoff is in eval memory, where REEF-GP carries an $\mathcal{O}(N_s^2)$ cost from per-expert kernel matrices; absolute values nonetheless remain well within commodity GPU memory.

4.2 Ablation Studies

We conduct five ablations using the *Airfoil* benchmark to dissect the impact of each architectural and methodological choice; full results, tables, and visualizations are deferred to Appendix D. (i) **Layer selection** (Appendix D.1): our CKA-based three-layer selection $\{L_0, L_{l^*}, L_{L-1}\}$ matches concatenating the entire 8-layer Transolver stack at significantly lower memory cost, and substantially outperforms single-layer variants. (ii) **Size of training data** (Appendix D.2): REEF-GP adapts gracefully to base operators of varying quality, with the predicted uncertainty correctly inflating to track the operator's deteriorating accuracy as M_{train} decreases from 750 to 50, while the GP's rL2 closely follows the base operator's across two orders of magnitude in training-set size (Figure 11). (iii) **Stochastic subset size** (Appendix D.3): performance is essentially insensitive to the per-expert support size N_s across two orders of magnitude (from 500 to 25,000), with overlapping error bars on every probabilistic metric. We adopt $N_s = 5{,}000$ as a middle-of-the-range default that balances conditioning quality against the cubic Cholesky cost. (iv) **Number of gPoE experts** (Appendix D.4):

calibration is essentially insensitive to the expert count $K \in \{1, 5, 10\}$. (v) **Heteroscedastic vs. homoscedastic noise** (Appendix D.5): both noise models yield comparable aggregate metrics, but the heteroscedastic variant produces sharper, error-localized uncertainty fields, motivating its use as our default.

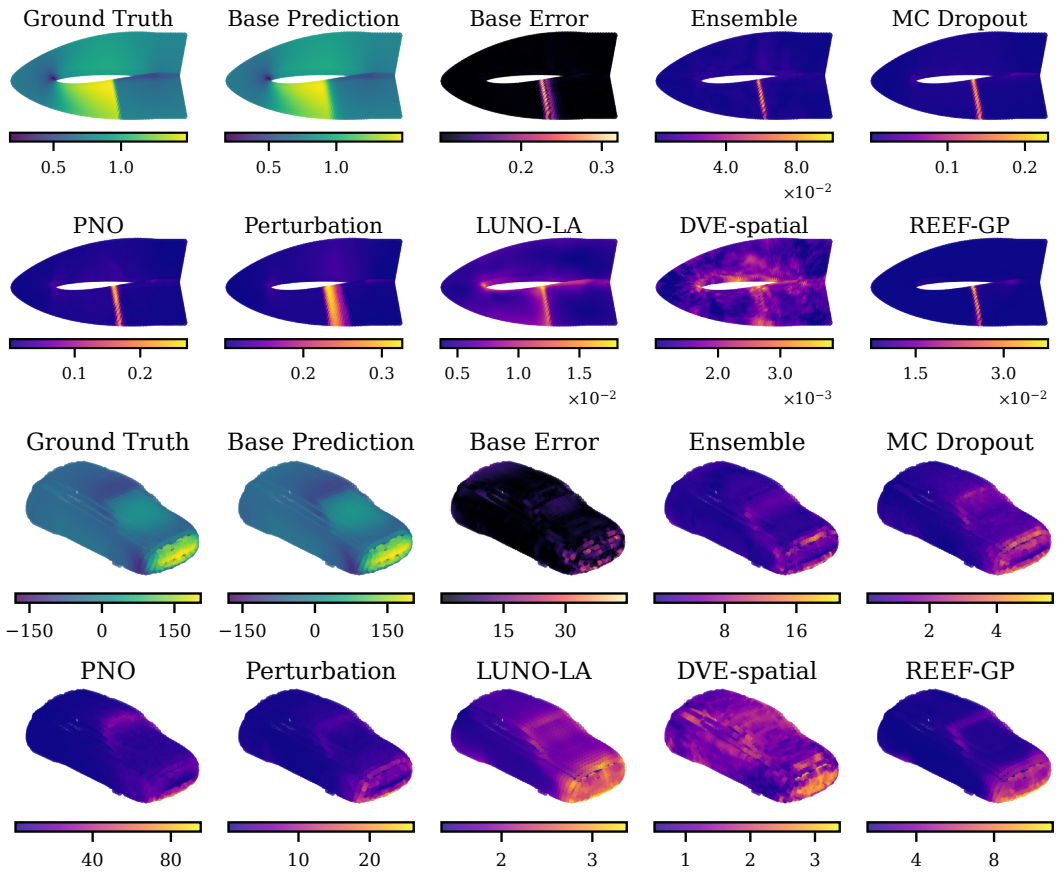

Figure 2: **Predictive standard deviation fields on representative 2D (*Airfoil*) and 3D (*ShapeNet Car*) benchmarks.** Top two rows show an airfoil sample and bottom two rows a car sample. For each sample the rows show in order: ground truth, base prediction, base error and the standard deviation of a sample for each of the UQ baselines.

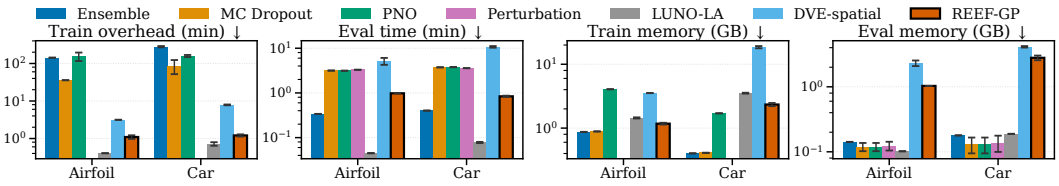

Figure 3: **Compute costs on representative 2D and 3D benchmarks:** Train overhead, evaluation time, peak train memory, and peak evaluation memory on *Airfoil* (2D) and *ShapeNet Car* (3D). All axes use log scale. REEF-GP's train and evaluation times are competitive but its memory requirements are higher. We define training overhead as the additional wall-clock time required to add UQ capabilities to a pretrained deterministic base Transolver. For REEF-GP it includes feature extraction from the frozen operator, CKA layer selection, and GP fitting, and excludes base-operator training.

4.3 Geometric Out-Of-Distribution (OOD) Robustness

To assess whether REEF-GP remains well-calibrated under geometric distribution shift, we partition each test set into four quartiles by maximum mean discrepancy (MMD) distance from the training distribution, ranging from Q1 (closest) to Q4 (most geometrically novel). A reliable UQ method should produce stable calibration across quartiles, inflating its predicted uncertainty as test geometries drift away from training. Across all benchmarks, REEF-GP preserves the base operator's rL2 and its NLL increases only modestly from Q1 to Q4. Full per-quartile rL2 and NLL metrics, MMD construction details, and per-method comparisons are reported in Appendix J and Table 11.

Extrapolation. We further consider two splits in which every test sample lies outside the training range: Reynolds-number extrapolation on *Ahmed Body* and shape-novelty extrapolation on *ShapeNet Car* (which increases the average geometric MMD to the training set by about $8.9\times$). All evaluated methods degrade, but REEF-GP degrades gracefully. Details are reported in Appendix J.1.

5 Conclusions

Our results show that the coordinate–feature space induced by a pretrained neural operator can be used for post-hoc UQ. By fitting REEF-GP directly on the neural operator's internal representations, we obtain calibrated uncertainty maps that are competitive with deep ensembles across five benchmarks but at a fraction of the training cost of a neural operator. The resulting uncertainty is also spatially interpretable: it concentrates around shock fronts and other regions where the operator's approximation is known to break down, and it remains stable as test geometries drift away from the training distribution. We view this as evidence that the geometry a neural operator learns while solving a PDE already encodes much of what is needed to quantify its own uncertainty, opening a complementary axis to parameter-centric approaches and a natural starting point for UQ on increasingly large geometry-aware operators.

Limitations and future work. REEF-GP relies on neural operators whose hidden states preserve pointwise spatial correspondence. Backbones with latent-grid or spectral representations need a resampling step, which we demonstrate for factorized Geo-FNO on Airfoil (Appendix K); a full comparison against other backbones is left for future work. Our main experiments involve stationary problems with single-output responses. We extend REEF-GP to vector-valued outputs via an intrinsic coregionalization model (Appendix L), whereas time-dependent problems, where time can enter as an additional coordinate in k_{space}, remain to be studied. Our shift studies cover geometry and Reynolds number; shifts in boundary conditions or mesh resolution are untested. As a post-hoc method, REEF-GP cannot correct systematic biases of the base operator, although it exposes them through inflated uncertainty. Finally, training memory is independent of dataset size and mesh resolution thanks to subset sampling, but peak evaluation memory scales as $\mathcal{O}(N_s^2)$ per expert, which is the dominant cost on large 3D meshes.

Acknowledgments and Disclosure of Funding

We appreciate the support from the Office of Naval Research (award number N000142312485) and National Science Foundation (award number 2238038).

References

[1] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators, 2022.

[2] Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: learning stable dynamics on the sphere. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[3] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[4] Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A. Yeh, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Pretraining codomain attention neural operators for solving multiphysics PDEs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[5] Vignesh Gopakumar, Stanislas Pamela, Lorenzo Zanisi, Zongyi Li, Anima Anandkumar, and MAST Team. Fourier neural operator for plasma modelling, 2023.

[6] N. Carey, L. Zanisi, S. Pamela, V. Gopakumar, J. Omotani, J. Buchanan, J. Brandstetter, F. Paischer, G. Galletti, and P. Setinek. Neural operator surrogate models of plasma edge simulations: feasibility and data efficiency, 2025.

[7] Mohammad S. Khorrami, Pawan Goyal, Jaber R. Mianroodi, Bob Svendsen, Peter Benner, and Dierk Raabe. A physics-encoded fourier neural operator approach for surrogate modeling of divergence-free stress fields in solids, 2025.

[8] Dhananjeyan Jeyaraj, Hamidreza Eivazi, Jendrik-Alexander Tröger, Stefan Wittek, Stefan Hartmann, and Andreas Rausch. A neural operator based hybrid microscale model for multiscale simulation of rate-dependent materials, 2025.

[9] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *J. Mach. Learn. Res.*, 24(1), January 2023.

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

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

[12] Hang Zhou, Haixu Wu, Haonan Shangguan, Yuezhou Ma, Huikun Weng, Jianmin Wang, and Mingsheng Long. Transolver-3: Scaling up transformer solvers to industrial-scale geometries, 2026.

[13] S. Chandra Mouli, Danielle C. Maddix, Shima Alizadeh, Gaurav Gupta, Andrew Stuart, Michael W. Mahoney, and Yuyang Wang. Using uncertainty quantification to characterize and improve out-of-domain learning for pdes, 2024.

[14] Apostolos F. Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902, 2023.

[15] Daniel Musekamp, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert. Active learning for neural PDE solvers. In *The Thirteenth International Conference on Learning Representations*, 2025.

[16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc.

[17] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1050–1059. JMLR.org, 2016.

[18] Albert Matveev, Sanmitra Ghosh, Aamal Hussain, James-Michael Leahy, and Michalis Michaelides. Light-weight diffusion multiplier and uncertainty quantification for fourier neural operators, 2025.

[19] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux - effortless bayesian deep learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20089–20103. Curran Associates, Inc., 2021.

[20] Emilia Magnani, Marvin Pförtner, Tobias Weber, and Philipp Hennig. Linearization turns neural operators into function-valued gaussian processes. In *Forty-second International Conference on Machine Learning*, 2025.

[21] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[22] Carlos Mora, Amin Yousefpour, Shirin Hosseinmardi, Houman Owhadi, and Ramin Bostanabad. Operator learning with gaussian processes. *Computer Methods in Applied Mechanics and Engineering*, 434:117581, 2025.

[23] Pau Batlle, Matthieu Darcy, Bamdad Hosseini, and Houman Owhadi. Kernel methods are competitive for operator learning. *Journal of Computational Physics*, 496, 2024.

[24] Matthew Lowery, John Turnage, Zachary Morrow, John D Jakeman, Akil Narayan, Shandian Zhe, and Varun Shankar. Kernel neural operators (knos) for scalable, memory-efficient, geometrically-flexible operator learning. *arXiv preprint arXiv:2407.00809*, 2024.

[25] Pau Batlle, Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Error analysis of kernel/gp methods for nonlinear and parametric pdes. *Journal of Computational Physics*, 520:113488, 2025.

[26] Houman Owhadi, Clint Scovel, and Tim Sullivan. On the brittleness of bayesian inference. *SIAM Review*, 57(4):566–582, 2015.

[27] Houman Owhadi and Gene Ryan Yoo. Kernel flows: From learning kernels from data into the abyss. *Journal of Computational Physics*, 389:22–47, 2019.

[28] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d PDEs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[29] Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, ICML'15, page 1775–1784. JMLR.org, 2015.

[30] Shandian Zhe, Wei Xing, and Robert M. Kirby. Scalable high-order gaussian process regression. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2611–2620. PMLR, 16–18 Apr 2019.

[31] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[32] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In David van Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[33] James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable Variational Gaussian Process Classification. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 351–360, San Diego, California, USA, 09–12 May 2015. PMLR.

[34] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain, 09–11 May 2016. PMLR.

[35] Wesley J. Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. *A simple baseline for Bayesian uncertainty in deep learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[36] Christopher Bülte, Philipp Scholl, and Gitta Kutyniok. Probabilistic predictions with fourier neural operators. In *NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty*, 2024.

[37] Christopher Bülte, Philipp Scholl, and Gitta Kutyniok. Probabilistic neural operators for functional uncertainty quantification. *Transactions on Machine Learning Research*, 2025.

[38] Jeremiah Zhe Liu, Shreyas Padhy, Jie Ren, Zi Lin, Yeming Wen, Ghassen Jerfel, Zachary Nado, Jasper Snoek, Dustin Tran, and Balaji Lakshminarayanan. A simple approach to improve single-model deep uncertainty via distance-awareness. *Journal of Machine Learning Research*, 24(42):1–63, 2023.

[39] David J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 05 1992.

[40] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

[41] Emilia Magnani, Nicholas Krämer, Runa Eschenhagen, Lorenzo Rosasco, and Philipp Hennig. Approximate bayesian neural operators: Uncertainty quantification for parametric pdes. *CoRR*, abs/2208.01565, 2022.

[42] Tobias Weber, Emilia Magnani, Marvin Pförtner, and Philipp Hennig. Uncertainty quantification for fourier neural operators. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, 2024.

[43] Felix Jimenez and Matthias Katzfuss. Vecchia gaussian process ensembles on internal representations of deep neural networks. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan, editors, *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 3403–3411. PMLR, 03–05 May 2025.

[44] Marc C. Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 63(3):425–464, 01 2002.

[45] Dave Higdon, Marc Kennedy, James C. Cavendish, John A. Cafeo, and Robert D. Ryne. Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing*, 26(2):448–466, 2004.

[46] J Loeppky, Derek Bingham, and W Welch. Computer model calibration or tuning in practice. Technical report, Department of Statistics, University of British Columbia, 2006.

[47] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, 09–15 Jun 2019.

[48] Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*, 2025.

[49] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[50] Michail Spitieris and Ingelin Steinsland. Bayesian calibration of imperfect computer models using physics-informed priors. *Journal of Machine Learning Research*, 24(108):1–39, 2023.

[51] Yanshuai Cao and David J. Fleet. Generalized product of experts for automatic and principled fusion of gaussian process predictions, 2015.

[52] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023.

[53] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. *CoRR*, abs/1905.00414, 2019.

[54] B. MacDonald, P. Ranjan, and H. Chipman. Gpfit: An r package for fitting a gaussian process model to deterministic simulator outputs. *Journal of Statistical Software*, 64(12):1–23, 2015.

[55] Amin Yousefpour, Zahra Zanjani Foumani, Mehdi Shishehbor, Carlos Mora, and Ramin Bostanabad. Gp+: A python library for kernel-based learning via gaussian processes. *Advances in Engineering Software*, 195:103686, 2024.

[56] Hao Chen, Lili Zheng, Raed Al Kontar, and Garvesh Raskutti. Gaussian process parameter estimation using mini-batch stochastic gradient descent: Convergence guarantees and empirical benefits. *Journal of Machine Learning Research*, 23(227):1–59, 2022.

[57] Xianjin Yang and Houman Owhadi. A mini-batch method for solving nonlinear pdes with gaussian processes, 2024.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[60] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[61] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[62] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015.

[63] Nobuyuki Umetani and Bernd Bickel. Learning three-dimensional flow for interactive aerodynamic design. *ACM Trans. Graph.*, 37(4), July 2018.

[64] Murat Seckin Ayhan and Philipp Berens. Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. In *Medical Imaging with Deep Learning*, 2018.

A Operator Learning with Gaussian Processes

We adopt the *functional regression* perspective of [22] as opposed to the operator-valued approach of [23] (we note that neither works study the UQ properties of GPs in the context of operator learning). Specifically, instead of learning the map $\mathcal{G}^\dagger$ directly (which outputs an infinite-dimensional function), we learn the evaluation functional associated with the operator. Formally, we define a real-valued bilinear form $\tilde{\mathcal{G}}^\dagger$ acting on the input space and the dual space of the output:

$$\tilde{\mathcal{G}}^\dagger : (\mathcal{U} \times \mathcal{A}) \times \mathcal{V}^* \to \mathbb{R}, \quad ((u, a), \varphi) \mapsto \langle \varphi, \mathcal{G}^\dagger(u, a) \rangle_{\mathcal{V}^* \times \mathcal{V}}, \tag{14}$$

where $\mathcal{V}^*$ is the dual space of $\mathcal{V}$ and $\langle \cdot, \cdot \rangle$ denotes the duality pairing. In particular, by choosing φ to be the Dirac delta functional $\delta_{\mathbf{x}}$ centered at $\mathbf{x} \in D_a$, the pairing recovers the pointwise evaluation of the solution field:

$$\tilde{\mathcal{G}}^\dagger((u, a), \delta_{\mathbf{x}}) = \langle \delta_{\mathbf{x}}, v \rangle = v(\mathbf{x}). \tag{15}$$

This transformation effectively converts the operator learning problem into a scalar regression task, where the spatial coordinate $\mathbf{x}$ becomes an input variable. The process can be extended to vector-valued cases by either repeating the above process or solving a multi-output regression problem (see Appendix L).

In practice, we only have access to finite discretizations. Let $\mathcal{E}_u : \mathcal{U} \to \mathbb{R}^{d_u}$ be a bounded linear observation operator that extracts discrete d_u dimensional features from the continuous input function (e.g., sensor measurements or nodes of a mesh in computer simulations) such that $\mathbf{u} = \mathcal{E}_u(u)$. Similarly, let $\mathcal{E}_a : \mathcal{A} \to \mathbb{R}^{d_a}$ denote a discretization operator that produces the discrete d_a-dimensional representation of the geometry such that $\mathbf{a} = \mathcal{E}_a(a)$. Consequently, the infinite-dimensional functional $\tilde{\mathcal{G}}^\dagger$ is approximated by a finite-dimensional surrogate:

$$f : \mathbb{R}^{d_u} \times \mathbb{R}^{d_a} \times \Omega \to \mathbb{R}, \quad (\mathbf{u}, \mathbf{a}, \mathbf{x}) \mapsto v(\mathbf{x}). \tag{3}$$

This formulation allows us to treat the triplet $\mathbf{z} = (\mathbf{u}, \mathbf{a}, \mathbf{x})$ as the joint input to a regression model. By placing a GP prior over f, i.e., $f(\mathbf{z}) \sim \mathcal{GP}(m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}'))$, we can infer the solution value $v(\mathbf{x})$ at any query point $\mathbf{x}$ within the irregular domain D_a, while naturally obtaining uncertainty estimates via the GP posterior.

B Implementation Details of REEF-GP

This appendix details our design choices as well as training and inference algorithms that can be used to instantiate REEF-GP and reproduce our results. Code is available at https://github.com/Bostanabad-Research-Group/GP-GeometryAware-OL.

B.1 Layer Selection

REEF-GP requires an augmented state $\bar{\mathbf{z}}$ that aggregates hidden features $\mathbf{h}_l(\mathbf{u}, \mathbf{a}, \mathbf{x})$ from L' selected layers of a frozen base neural operator (Transolver in this case). Naively concatenating all $L = 8$ Transolver layers would produce a very high-dimensional representation that increases the costs while providing negligible gains as these representations have redundancies. Internal Transolver layers exhibit strong feature correlation so consecutive blocks contribute overlapping information, see Figure 4. We therefore select a small subset of layers that jointly span the network's representational hierarchy: an early layer (preserving raw geometric structure), a late layer (encoding the predicted physics), and a middle layer that is maximally distinct from both. Below, we provide a heuristic for choosing the middle layer which has been used in all our studies. Unlike DVE, we observe that not all layers are needed, see ablation studies in Table 3.

CKA-based middle layer selection. For the middle layer, we use a centered kernel alignment (CKA) [53] criterion to identify the block whose representation is jointly most distinct from the first and last layers. We pass a batch of B samples through the frozen Transolver and extract the hidden state at each candidate layer l. Flattening across the batch and spatial dimensions yields $\mathbf{H}_l \in \mathbb{R}^{(B \cdot N) \times D_h}$. We randomly subsample $P_s = 5{,}000$ rows and center each subsampled tensor before computing CKA. For two centered matrices $\mathbf{X}$ and $\mathbf{Y}$, linear CKA is:

$$\mathrm{CKA}(\mathbf{X}, \mathbf{Y}) = \frac{\|\mathbf{Y}^\top \mathbf{X}\|_F^2}{\|\mathbf{X}^\top \mathbf{X}\|_F \cdot \|\mathbf{Y}^\top \mathbf{Y}\|_F}. \tag{16}$$

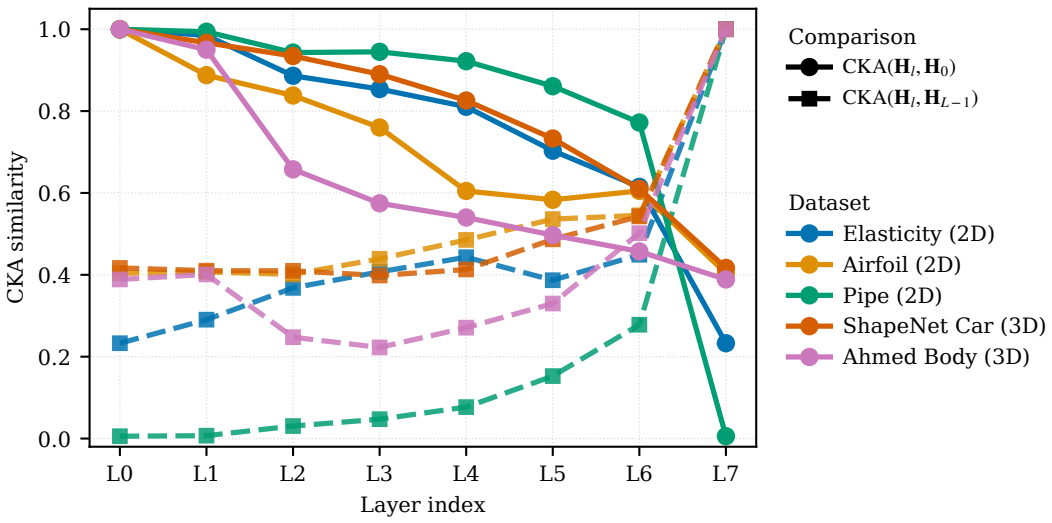

Figure 4: **Layer-wise CKA similarity across benchmarks.** For each candidate Transolver layer $\mathbf{H}_l$, we plot CKA similarity to the input block $\mathbf{H}_0$ (solid) and to the output block $\mathbf{H}_{L-1}$ (dashed) as a function of depth. Intermediate layers are jointly distinct from both endpoints and our middle-layer selection chooses the maximally distinct one.

The middle layer l^* is then chosen to minimize the combined similarity to the boundary layers:

$$l^* = \arg \min_{l \in \{1,\dots,L-2\}} \big[\mathrm{CKA}(\mathbf{H}_l, \mathbf{H}_0) + \mathrm{CKA}(\mathbf{H}_l, \mathbf{H}_{L-1}) \big]. \tag{17}$$

This procedure runs once per dataset and seed, takes negligible time relative to GP training, and selects three layers in total: $\{L_0, L_{l^*}, L_{L-1}\}$. Their outputs are concatenated along the feature dimension to form $\mathbf{h}(\mathbf{u}, \mathbf{a}, \mathbf{x})$. Hidden dimensions vary per dataset (Table 8), so the resulting augmented state has dimension $3D_h$.

B.2 Deep Kernel Design

The GP prior on f in Equation 3 has a deep kernel that operates on three input types (spatial coordinates $\mathbf{x}$, discretized input function $\mathbf{u}$, and the selected embeddings $\mathbf{h}$ from Transolver).

Latent feature projection. We project the high-dimensional augmented state $\mathbf{h}$ through a learned MLP $\rho : \mathbb{R}^{3D_h} \to \mathbb{R}^{D_l}$ that mixes the three layers and projects them into a compact latent space for the kernel. The network consists of two spectrally normalized linear layers with a ReLU activation:

$$\rho(\mathbf{h}) = W_2 \cdot \mathrm{ReLU}(W_1 \mathbf{h}) \tag{18}$$

Spectral normalization bounds the Lipschitz constant of ρ, preventing the projection from collapsing the input geometry manifold and preserving the latent distance metric for the kernel [38].

Kernel composition. The composite kernel is a product of three base kernels:

$$k_{\boldsymbol{\phi}}(\bar{\mathbf{z}}, \bar{\mathbf{z}}') = \sigma^2 \cdot k_{\text{space}}(\mathbf{x}, \mathbf{x}') \cdot k_{\text{fun}}(\mathbf{u}, \mathbf{u}') \cdot k_{\text{latent}}(\rho(\mathbf{h}), \rho(\mathbf{h}')), \tag{19}$$

where σ^2 is a learned global outputscale, k_{space} acts on physical coordinates ($d = 2$ or 3), k_{fun} acts on the discretized input function when present (e.g., tension traction on the top edge for *Elasticity*), and k_{latent} acts on the projected augmented state. The function-dimension factor k_{fun} is omitted when no input function is present. We use radial basis function (RBF) with automatic relevance determination (ARD) lengthscales as the base kernel across all the datasets. Additionally, as recommended in [54, 55], change of variable is used so that the lengthscales w are all optimized in the $\log_{10}$-space, e.g., $k_{\text{space}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\sum_{r=1}^{d} 10^{w_r}(x_r - x'_r)^2\right)$. To ensure numerical stability, we restrict the search interval to $[-5, 3]$.

Hyperprior Design. To avoid overfitting, we place Gaussian priors on the lengthscales and log-process variance:

$$w_{\text{space}} \sim \mathcal{N}(-1.5,\, 0.5^2),$$
$$w_{\text{fun}} \sim \mathcal{N}(0.0,\, 0.5^2),$$
$$w_{\text{latent}} \sim \mathcal{N}(-1.0,\, 0.5^2),$$
$$\log \sigma^2 \sim \mathcal{N}(0.0, 0.5^2).$$

These priors are fixed across all experiments. No prior is used for the parameters of $\rho(\cdot)$.

B.3 Heteroscedastic Noise Likelihood

The structural error ε and the observational noise ϵ are functionally indistinguishable from the GP's perspective as both contribute to the diagonal of the covariance matrix. They are jointly absorbed into a single input-dependent noise variance $\sigma^2_{\boldsymbol{\psi}}(\bar{\mathbf{z}}) + \sigma_n^2$ parametrized by an MLP as:

$$\sigma^2_{\boldsymbol{\psi}}(\bar{\mathbf{z}}) + \sigma_n^2 = \text{Softplus}\big(g_{\boldsymbol{\psi}}(\bar{\mathbf{z}})\big) + \sigma^2_{\min}, \tag{20}$$

where $g_{\boldsymbol{\psi}}$ is a 3-layer MLP with hidden dimensions $[64, 32]$, output dimension of 1, GELU activations, layer normalization at the input, and spectral normalization on each linear layer. We set $\sigma^2_{\min} = 10^{-3}$ to ensure numerical conditioning of the covariance matrix. The final linear layer's bias is initialized to -3.0 so that $\sigma^2_{\boldsymbol{\psi}}$ starts at a small value. The MLP receives the same input representation used by the kernel $\{\mathbf{x}, \mathbf{u}, \mathbf{h}\}$ and it returns a diagonal noise covariance:

$$p(v(\mathbf{x}) \mid f(\bar{\mathbf{z}})) = \mathcal{N}\big(f(\bar{\mathbf{z}}),\ \sigma^2_{\boldsymbol{\psi}}(\bar{\mathbf{z}}) + \sigma_n^2\big). \tag{21}$$

B.4 Training

The full functional regression problem has $MN \in [5 \times 10^5, 5 \times 10^6]$ training points across our benchmarks, making exact GP training intractable. To circumvent this, we draw inspiration from Kernel Flows (KF) [27] and employ stochastic subsampling. While KF loss typically minimizes an RKHS stability metric, we directly minimize the mini-batch negative marginal log-likelihood (NLL) which, unlike KF loss, also prioritizes uncertainty calibration. Although optimizing on mini-batches yields a biased estimator of the full-data NLL, recent works have successfully leveraged random mini-batches [27, 56, 57] and our empirical studies demonstrate similar trends, see Figure 5. We train using a fixed max-iteration budget and the configurations detailed in Table 2.

Table 2: **Hyperparameter configurations for our method across benchmarks.** Almost the same configuration is used for all five datasets with minimal per-task tuning, demonstrating robustness across varied geometries and physical regimes.

Hyperparameter	**Elasticity**	**Airfoil**	**Pipe**	**ShapeNet Car**	**Ahmed Body**
Latent dim D_l	64	64	64	64	64
Spatial kernel k_{space}	RBF	RBF	RBF	RBF	RBF
Input function kernel k_{fun}	RBF	–	–	–	RBF
Latent kernel k_{latent}	RBF	RBF	RBF	RBF	RBF
Subset size N_s	5,000	5,000	5,000	5,000	5,000
Number of experts K	5	5	5	5	5
Eval chunk size N_q	10,000	10,000	10,000	10,000	10,000
Max iterations	500	500	500	500	500
NN learning rate η_{nn}	2×10^{-3}	2×10^{-3}	2×10^{-4}	2×10^{-3}	2×10^{-3}
GP learning rate η_{gp}	5×10^{-2}	5×10^{-2}	1×10^{-3}	5×10^{-2}	5×10^{-2}
Weight decay (NN)	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}

Numerical stability. GP marginal likelihood and predictive computations are performed using exact Cholesky decomposition (jitter 10^{-6}) when the kernel remains positive definite, and we fall back to conjugate gradient (CG) approximation if it becomes mildly ill-conditioned during training. With a support set of $N_s = 5000$ points, Cholesky is computationally feasible ($O(N_s^3)$ cost). We additionally clip gradients to a maximum norm of 10 before each optimizer step.

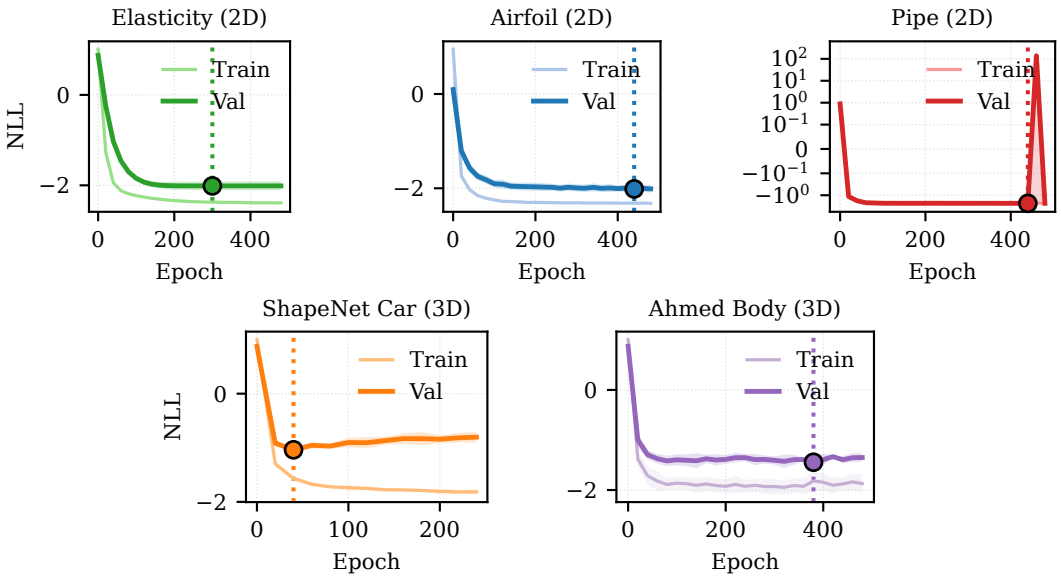

Figure 5: **NLL loss curves across benchmarks for five seeds.** Solid lines represent the mean NLL, with shaded regions indicating one standard deviation. To prevent overfitting of the GP hyperparameters, we continuously track the validation loss and employ early stopping. The optimal stopping point, where validation loss reaches its minimum, is marked by the dotted vertical line and point in each subplot.

Reproducibility details. Base Transolver models are previously trained using NVIDIA A100 GPUs (40 GB). All post-hoc UQ method fitting and inference for both our method and the baselines are performed on a single workstation equipped with an NVIDIA RTX 6000 Ada Generation GPU (48 GB). This split reflects realistic deployment: the deterministic operator is trained once leveraging HPC resources, while the post-hoc UQ component is fit and queried on a workstation. All experiments use `float32` precision. Baselines for MC Dropout, PNO, Perturbation, LUNO-LA, and DVE-spatial share the same Transolver backbone described in Appendix E. Random seeds $\{0, 1, 2, 3, 4\}$ are used across all experiments.

B.5 Inference

At inference time, conditioning on the full support set of MN training points remains computationally prohibitive. We employ a uniformly weighted generalized Product of Experts (gPoE) [51]: K random support subsets of size N_s condition K GP experts, each producing a posterior predictive mean $\bar{\delta}_k(\bar{\mathbf{z}}_*)$ and variance $\sigma_k^2(\bar{\mathbf{z}}_*)$ at a query point $\bar{\mathbf{z}}_* = (\mathbf{u}_*, \mathbf{x}_*, \mathbf{h}(\mathbf{u}_*, \mathbf{a}_*, \mathbf{x}_*))$. Unlike mixture aggregation, gPoE combines experts multiplicatively in precision space, yielding a closed-form Gaussian:

$$\frac{1}{\sigma_*^2(\bar{\mathbf{z}}_*)} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{\sigma_k^2(\bar{\mathbf{z}}_*)}, \qquad \bar{\delta}_*(\bar{\mathbf{z}}_*) = \sigma_*^2(\bar{\mathbf{z}}_*)\cdot\frac{1}{K}\sum_{k=1}^{K}\frac{\bar{\delta}_k(\bar{\mathbf{z}}_*)}{\sigma_k^2(\bar{\mathbf{z}}_*)}. \tag{22}$$

The aggregated discrepancy is added back to the frozen base operator to form the final predictive distribution:

$$v(\mathbf{x}_*) \sim \mathcal{N}\Big(\mathcal{G}_\theta(\mathbf{u}_*, \mathbf{a}_*)(\mathbf{x}_*) + \bar{\delta}_*(\bar{\mathbf{z}}_*),\ \sigma_*^2(\bar{\mathbf{z}}_*)\Big). \tag{23}$$

This formulation bounds the test-time memory footprint strictly to $\mathcal{O}(N_s^2)$ per expert, circumventing the $\mathcal{O}((MN)^2)$ requirement of the full support set, while remaining exact within each expert's local approximation. The GP experts use the same hyperparameters and only differ in their conditioning data.

Memory complexity at inference. The dominant memory cost is the $\mathcal{O}(N_s^2)$ kernel matrix stored once per expert (re-used across all query batches). Query points are processed in chunks of $N_q = 10{,}000$ to bound peak memory; this gives a worst-case per-batch peak of $\mathcal{O}(N_s \cdot N_q)$.

C Deformation Flow of the Geometry

In Section 3.2 we introduce a deep kernel formulation that relies on the frozen internal representations of a pretrained neural operator rather than learning a feature map from scratch. This design connects two established lines of work: deep kernel learning (DKL) [34], which composes a stationary kernel with a learned feature extractor, and KF [27], which constructs data-dependent kernels through iterative, flow-based deformations of the input space. Mechanically, REEF-GP is closest to DKL but conceptually several of our design choices are inspired by KF. The remainder of this appendix formalizes these connections and empirically analyzes the layer-wise geometry deformation of the Transolver backbone.

Residual Networks as KF. KF constructs a deep kernel by evaluating a base kernel K_1 on a deformed input space. For two input states $\mathbf{z}$ and $\mathbf{z}'$, the kernel takes the form $K(\mathbf{z}, \mathbf{z}') = K_1(F_L(\mathbf{z}), F_L(\mathbf{z}'))$ where F_L is a flow map learned via a sequence of incremental deformations:

$$F_{l+1}(\mathbf{z}) = F_l(\mathbf{z}) + \tau\, G_{l+1}(F_l(\mathbf{z})), \tag{24}$$

with τ acting as a discrete step size and G_{l+1} a transformation in the span of kernel evaluations. This incremental update mirrors the residual blocks within Transolver [10] and motivates us to leverage its internal geometry-aware representations.

Transolver is built upon transformer blocks [58] with residual connections [59]. Let $\mathbf{h}_l$ denote the latent state at layer l corresponding to a spatial coordinate $\mathbf{x}$. The layer-wise update reads:

$$\mathbf{h}_{l+1} = \mathbf{h}_l + \mathcal{F}_{\text{attn}}(\mathbf{h}_l) + \mathcal{F}_{\text{MLP}}(\mathbf{h}_l + \mathcal{F}_{\text{attn}}(\mathbf{h}_l)), \tag{25}$$

where $\mathcal{F}_{\text{attn}}$ is the Physics-Attention mechanism. Structurally, Transolver executes a discrete flow mapping that deforms the original coordinate and input space $\mathcal{U} \times \mathcal{A} \times \Omega$ into a structured latent manifold. We do not claim that Transolver is a KF model (it is trained via empirical risk minimization against the PDE solution, not the KF stability loss) but the layer-wise update has the same residual-flow geometry, which is what suggests that the intermediate hidden states carry useful information beyond the final embedding alone.

Figures 6 and 7 show the t-SNE representation of the internal embeddings of all the Transolver layers for two airfoil samples. The network learns to deform the geometry into latent representations that resolve the physics of the problem.

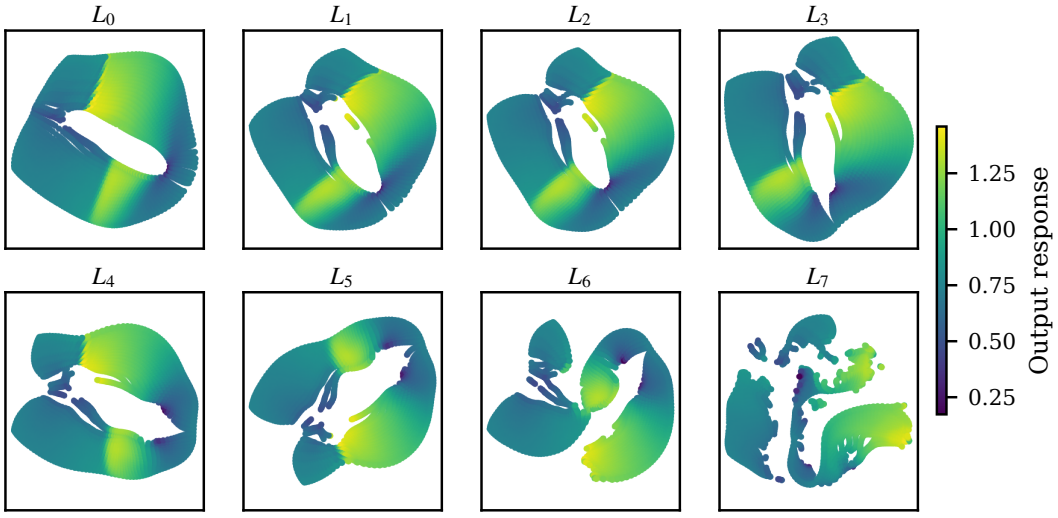

Figure 6: **Deformation flow of the geometry for an airfoil sample with shock waves above and below the airfoil.** Each panel shows the two-dimensional t-SNE representations of the pointwise embeddings of the initial geometry at each of the 8 layers of the Transolver backbone. Across depth, the network progressively deforms the original geometry into a latent manifold, supporting the view that intermediate layers consistently encode geometry-aware structure.

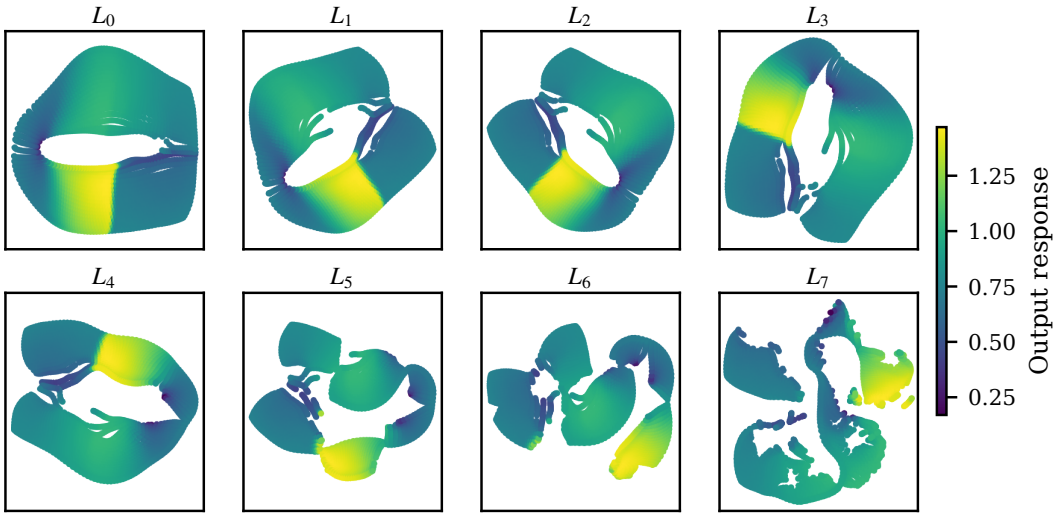

Figure 7: **Deformation flow of the geometry for an airfoil sample with a shock wave below the airfoil.** Each panel shows the two-dimensional t-SNE representations of the pointwise embeddings of the initial geometry at each of the 8 layers of the Transolver backbone.

Relation to DKL. Using learned internal representations inside a stationary kernel with learnable hyperparameters is essentially the same as DKL. However, two design choices distinguish REEF-GP from a standard DKL pipeline, and both follow from the post-hoc UQ goal. First, the feature extractor is not trained jointly with the GP: $\mathcal{G}_\theta$ has already produced a physics-aware deformation by solving

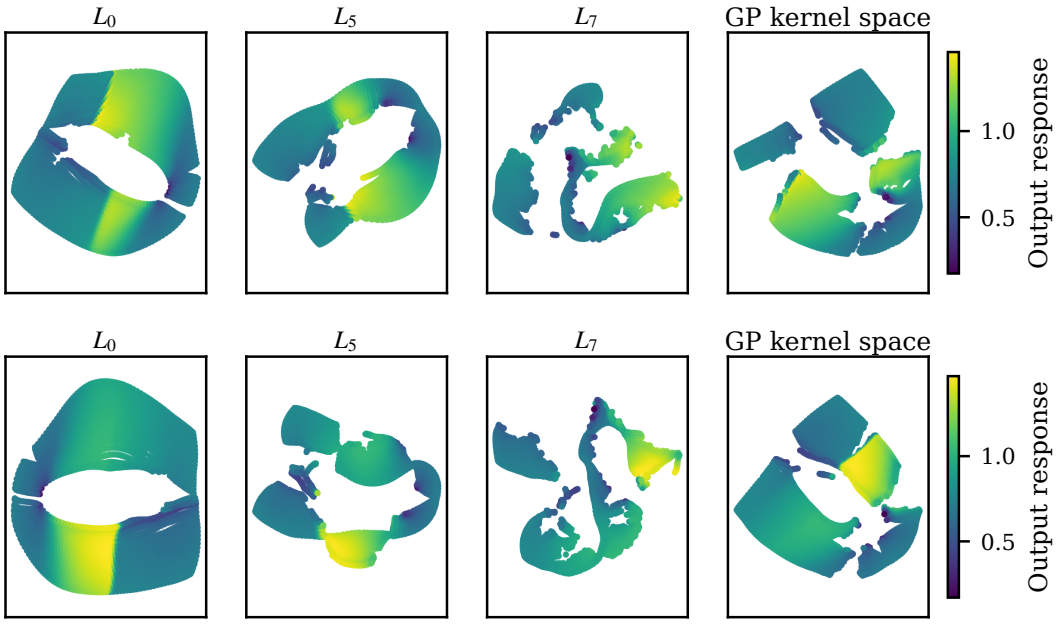

Figure 8: **Feature map from Transolver embeddings to the GP kernel space.** Top: airfoil sample with shock waves at both upper and lower surfaces. Bottom: airfoil sample with shock wave at the lower surface. The first three columns show t-SNE projections of the three CKA-selected Transolver layers used to construct the augmented state $\mathbf{h}(\mathbf{u}, \mathbf{a}, \mathbf{x})$; the rightmost column shows the embedding after the spectral-normalized projection $\rho(\cdot)$ used inside the kernel. The GP kernel space tears precisely at the shock wave region, indicating that the projection reshapes the operator's latent geometry into a metric space where uncertainty can localize around physically meaningful error regions.

the PDE under empirical risk minimization, and we reuse that deformation rather than relearning one. The internal layers deform the geometry into a manifold where the physical dynamics are smoothly resolvable, but the resulting embeddings are not, on their own, adapted to provide UQ.

Second, scaling DKL to operator learning datasets ($MN \sim 10^6$) typically relies on sparse approximations such as inducing points [32, 33], which impose a low-rank covariance structure that can constrain calibration in ways that are difficult to control without additional regularization. We take a different route, again inspired by KF: rather than committing to a fixed low-rank structure, we condition on random subsets of the training data at every step (Appendix B.4). KF itself was designed for interpolation and explicitly discards UQ information, so we keep its stochastic-subset philosophy while replacing the RKHS-stability loss with a mini-batch negative log marginal likelihood that retains the calibration signal.

Adapting the Flow for UQ. The Transolver embeddings are shaped by predictive accuracy, not by the geometry a stationary kernel would prefer for UQ. We therefore need to compress them into a new latent space whose role is to calibrate uncertainty rather than to relearn the flow. Figure 8 illustrates the mechanism. Through the layer selection algorithm we choose three internal representations that are maximally distinct, project them through a spectral-normalized MLP, and use the resulting embedding to augment the input features before feeding them to the stationary kernel. The projection yields a geometry-aware embedding suited for UQ: as seen in Figure 8, the representation tears off at the shock wave regions, bringing some interpretability to the metric space used to quantify uncertainty.

D Ablation Studies

We use the *Airfoil* benchmark for the ablation studies. This dataset is sufficiently challenging to surface meaningful differences between design choices, while remaining feasible to retrain across many configurations. The training details and hardware used are described in Appendix B.4.

D.1 Layer Selection

We compare four strategies for constructing the augmented state $\mathbf{h}$: (i) using only the input block L_0, (ii) only the output block L_{L-1}, (iii) concatenating the full stack of all layers, and (iv) our default CKA-based selection of three layers $\{L_0, L_{l^*}, L_{L-1}\}$. Results are reported in Table 3 and visualized in Figure 9. Single-layer variants (L_0 or L_{L-1} alone) underperform across all probabilistic metrics, with L_0 alone being the weakest. Our CKA-selected three-layer combination is statistically indistinguishable from concatenating the full stack (overlapping error bars on every metric), at lower memory and compute cost.

D.2 Number of Training Samples

We jointly vary the number of training samples used to train the base Transolver and learn the GP discrepancy model. The test set is held fixed across all configurations. As the operator is trained on more data, its predictions become more accurate, leaving smaller residuals for the GP to model.

Table 3: **Layer selection ablation.** Effect of the layer set used to construct the augmented state $\mathbf{h}$. Lower is better (↓). L_0 uses only the input block; L_{L-1} only the output; All layers concatenates the entire stack; **REEF-GP** selects $\{L_0, L_{l^*}, L_{L-1}\}$ via CKA. Best per metric in **bold**, second best underlined.

Method	**rL2** ↓	**NLL** ↓	**CRPS** ↓	**NIS** ↓	**ES** ↓
Airfoil (2D)	(%)		($\times 10^{-4}$)	($\times 10^{-3}$)	
REEF-GP	$\mathbf{1.24}$ ± 0.12	$\underline{-3.51}$ ± 0.06	$\underline{39.80}$ ± 2.35	$\underline{56.89}$ ± 4.66	$\mathbf{0.45}$ ± 0.05
L_{L-1} only	$\mathbf{1.24}$ ± 0.12	-3.36 ± 0.12	40.25 ± 2.41	60.70 ± 5.10	$\underline{0.46}$ ± 0.05
L_0 only	$\mathbf{1.24}$ ± 0.12	-2.75 ± 0.14	40.89 ± 2.46	67.72 ± 5.50	$\underline{0.46}$ ± 0.05
All layers ($L_0 \dots L_{L-1}$)	$\mathbf{1.24}$ ± 0.12	$\mathbf{-3.54}$ ± 0.06	$\mathbf{39.71}$ ± 2.40	$\mathbf{55.99}$ ± 4.74	$\mathbf{0.45}$ ± 0.05

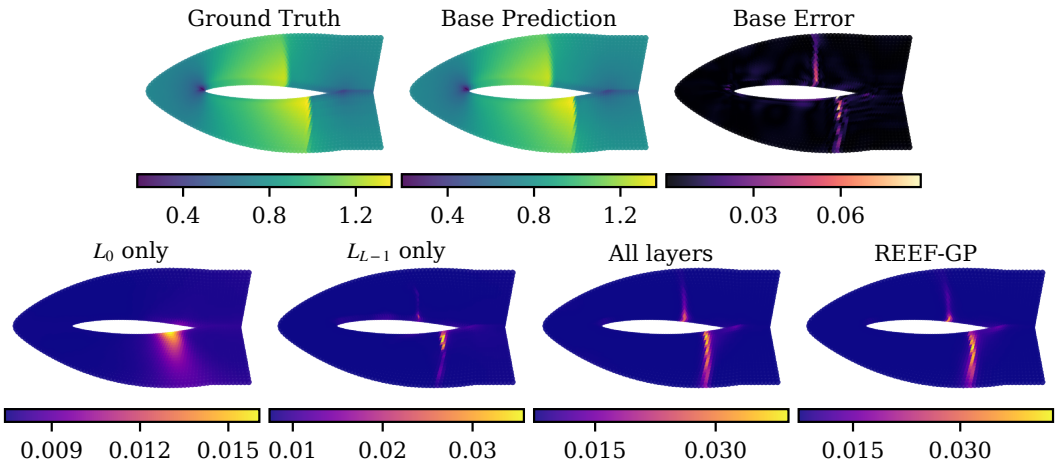

Figure 9: **Layer-selection ablation on *Airfoil*.** Top: ground truth field, base prediction, absolute error. Bottom: predicted standard deviation under four layer-selection variants. All variants concentrate uncertainty around the shock front; the CKA-selected three-layer combination (REEF-GP) closely matches the full-stack baseline at reduced time and memory costs.

We observe that REEF-GP adapts gracefully to base operators of varying quality, recovering well-calibrated uncertainty in both data-rich and data-poor regimes. Results are reported in Table 4 and visualized in Figure 10 and Figure 11.

Table 4: **Training samples ablation on *Airfoil*.** Effect of the number of training samples used to train both the base Transolver and our GP discrepancy model. Lower is better (↓). Best per metric in **bold**, second best underlined.

Method	**rL2** ↓	**NLL** ↓	**CRPS** ↓	**NIS** ↓	**ES** ↓
Airfoil (2D)	(%)		($\times 10^{-3}$)	($\times 10^{-2}$)	
$M_{\text{train}} = 50$	9.68 ±0.80	0.34 ±1.87	32.55 ±3.04	62.16 ±11.72	3.70 ±0.38
$M_{\text{train}} = 150$	3.92 ±1.14	−2.17 ±0.63	11.22 ±2.75	18.45 ±5.47	1.49 ±0.42
$M_{\text{train}} = 250$	2.18 ±0.68	−3.07 ±0.06	6.35 ±1.46	9.11 ±0.87	0.80 ±0.22
$M_{\textbf{train}} = 500$	1.24 ±0.12	−3.51 ±0.06	3.98 ±0.24	5.69 ±0.47	0.45 ±0.05
$M_{\text{train}} = 750$	**1.03** ±0.03	**−3.60** ±0.02	**3.46** ±0.04	**5.06** ±0.08	**0.38** ±0.01

D.3 Stochastic Training Subset Size

We ablate the per-expert support-set size N_s used both during stochastic training (each gradient step conditions the GP on a random size-N_s subset of the training residuals) and during inference (each gPoE expert is conditioned on an independent size-N_s subset). Larger N_s provides each GP with more conditioning data per step, at the cost of $\mathcal{O}(N_s^3)$ per Cholesky factorization. Results are reported in Table 5 and visualized in Figure 12. We observe that REEF-GP recovers the same calibration quality across two orders of magnitude in N_s (from 500 to 25,000). Figure 12 confirms this visually for a random test sample: the predicted uncertainty fields are nearly identical across all five subset sizes, concentrating around the shock front in every variant. Following these observations, we adopt $N_s = 5000$ as a middle-of-the-range default that balances conditioning quality against the cubic cost of Cholesky factorization.

D.4 Number of Experts

We compare three values of K, the number of GP experts used in the gPoE inference scheme: $K = 1$, our default $K = 5$, and $K = 10$. Results on *Airfoil* are reported in Table 6 and visualized in Figure 13. Calibration is essentially insensitive to K on this benchmark: all three settings produce statistically indistinguishable metrics.

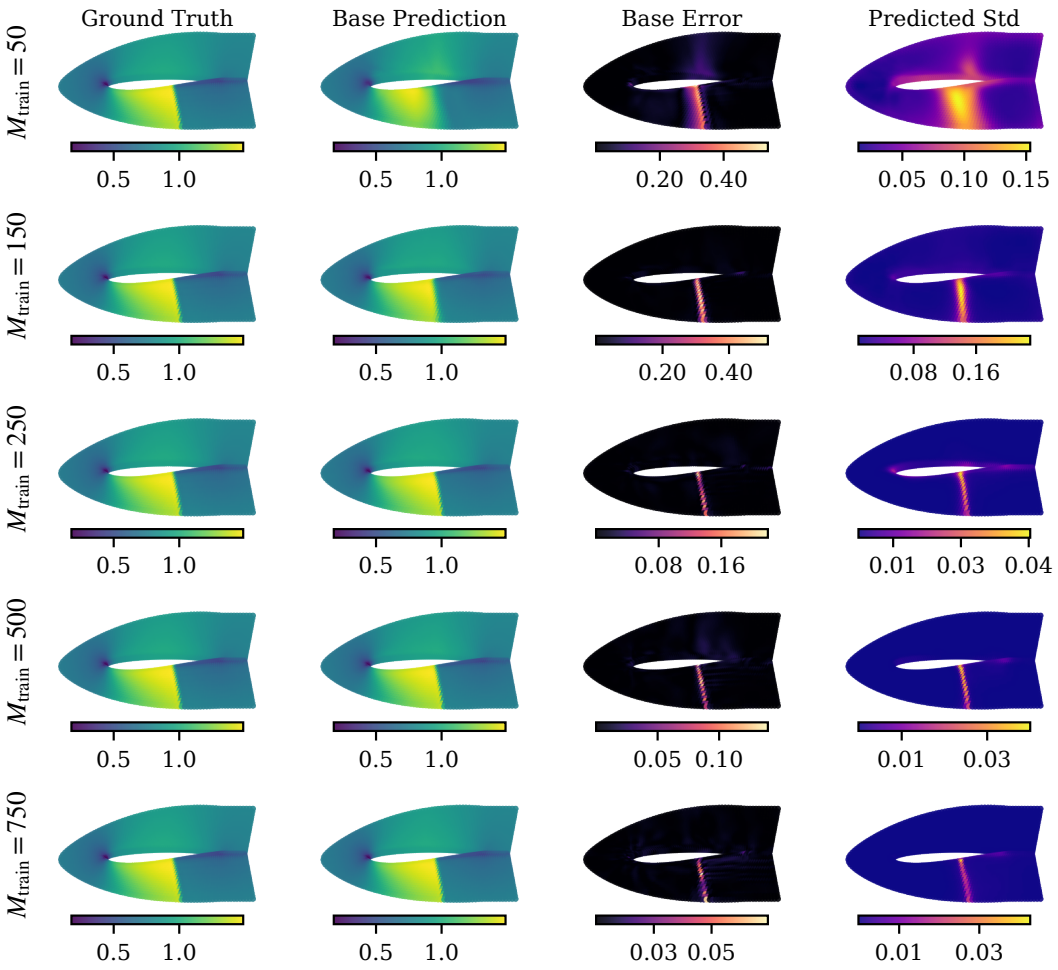

Figure 10: **Training samples ablation on *Airfoil*.** Each row corresponds to a different training-set size M_{train}, applied jointly to the base Transolver and the GP discrepancy model. Columns: ground truth field, base prediction, absolute error of the base prediction, and predicted standard deviation. As M_{train} decreases, the base operator's error grows; REEF-GP correctly inflates its predicted uncertainty to track this growth, with the spatial structure of the predicted standard deviation closely matching the spatial structure of the error.

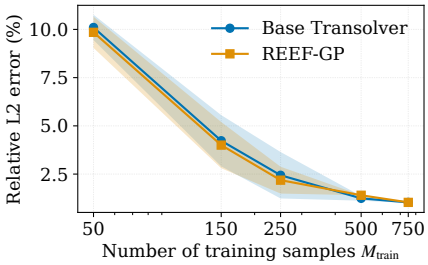

Figure 11: **Relative L2 error vs. training-set size on *Airfoil*.** Both curves show mean across 5 seeds with shaded ± 1 standard deviation. REEF-GP closely follows the base Transolver's accuracy across two orders of magnitude in M_{train}, confirming that adding GP-based uncertainty quantification does not degrade predictive performance.

Table 5: **Subset-size ablation on *Airfoil*.** Effect of the per-expert support-set size N_s used at inference. For the results in text we use $N_s = 5000$. Lower is better (↓).

Method	**rL2** ↓	**NLL** ↓	**CRPS** ↓	**NIS** ↓	**ES** ↓
Airfoil (2D)	(%)		($\times 10^{-4}$)	($\times 10^{-3}$)	
$N_s = 500$	1.24 ±0.12	−3.50 ±0.08	39.91 ±2.39	57.00 ±5.53	0.45 ±0.05
$N_s = 1000$	1.24 ±0.12	−3.50 ±0.08	39.87 ±2.41	56.81 ±5.92	0.45 ±0.05
$N_s = 5000$	1.24 ±0.12	−3.51 ±0.06	39.80 ±2.35	56.89 ±4.66	0.45 ±0.05
$N_s = 10000$	1.24 ±0.12	−3.51 ±0.06	39.80 ±2.36	56.94 ±4.64	0.45 ±0.05
$N_s = 25000$	1.24 ±0.12	−3.51 ±0.06	39.80 ±2.38	57.30 ±4.68	0.45 ±0.05

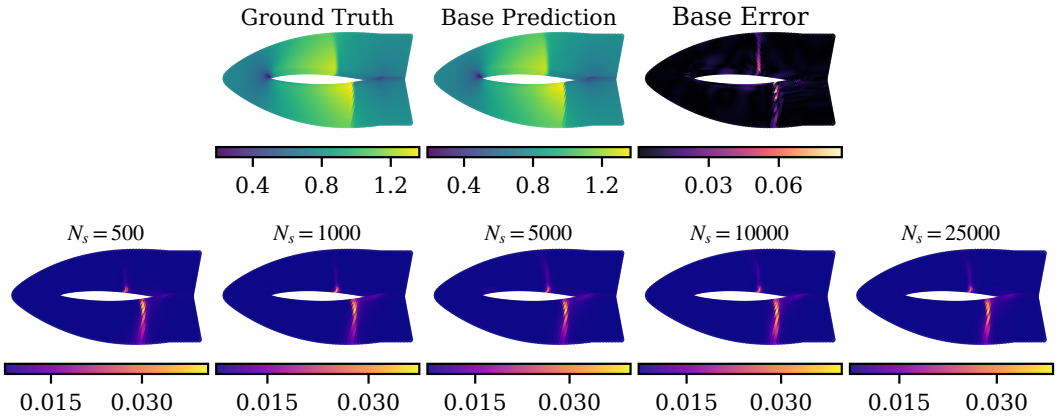

Figure 12: **Subset-size ablation on *Airfoil*.** Top: ground truth field, base prediction, absolute error. Bottom: predicted standard deviation under five subset sizes N_s. All variants concentrate uncertainty around the shock front.

D.5 Heteroscedastic vs. Homoscedastic Noise

We compare our heteroscedastic noise model $\sigma^2_\psi(\bar{\mathbf{z}})$, which is input-dependent and parameterized by an MLP, against a homoscedastic baseline that fits a single scalar noise variance. Results are reported in Table 7 and visualized in Figure 14. The two noise models yield comparable aggregate metrics, with overlapping error bars on every probabilistic score. The qualitative difference is spatial: both models concentrate uncertainty around the shock front, but the heteroscedastic model produces a sharper, higher-magnitude uncertainty signal localized to the error region, while the homoscedastic model assigns non-negligible uncertainty across regions where the operator's error is small, with a compressed numerical range overall. We adopt the heteroscedastic model as our default as it provides more structurally meaningful uncertainties.

Table 6: **Number of gPoE experts ablation on *Airfoil*.** Effect of the number of GP experts K used in the generalized Product of Experts inference scheme. Lower is better (↓).

Method	**rL2** ↓	**NLL** ↓	**CRPS** ↓	**NIS** ↓	**ES** ↓
Airfoil (2D)	(%)		($\times 10^{-4}$)	($\times 10^{-3}$)	
$K = 1$	1.24 ±0.12	−3.51 ±0.06	39.82 ±2.36	56.90 ±4.65	0.45 ±0.05
$K = 5$	1.24 ±0.12	−3.51 ±0.06	39.80 ±2.35	56.89 ±4.66	0.45 ±0.05
$K = 10$	1.24 ±0.12	−3.51 ±0.06	39.80 ±2.35	56.89 ±4.66	0.45 ±0.05

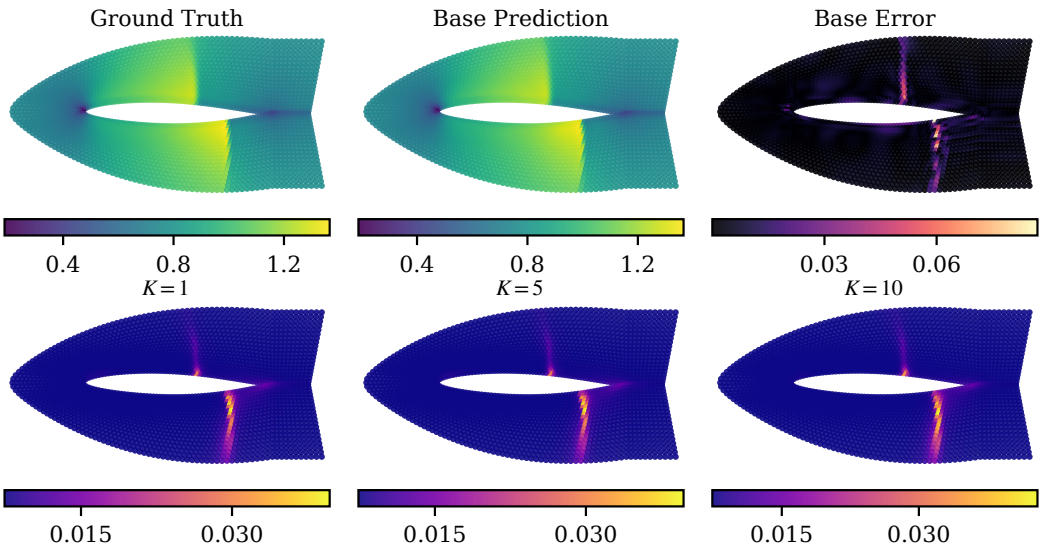

Figure 13: **Number of gPoE experts ablation on *Airfoil*.** Top: ground truth field, base prediction, absolute error. Bottom: predicted standard deviation under three values of K, the number of GP experts in the gPoE inference scheme.

Table 7: **Heteroscedastic vs. homoscedastic noise ablation on *Airfoil*.** Effect of the noise model used in the GP likelihood. Heteroscedastic uses an input-dependent variance $\sigma^2_{\psi}(\bar{\mathbf{z}})$; homoscedastic uses a single scalar variance.

Method	**rL2** ↓	**NLL** ↓	**CRPS** ↓	**NIS** ↓	**ES** ↓
Airfoil (2D)	(%)		($\times 10^{-4}$)	($\times 10^{-3}$)	
Heteroscedastic	1.24 $\pm$0.12	$-$3.51 $\pm$0.06	39.80 $\pm$2.35	56.89 $\pm$4.66	0.45 $\pm$0.05
Homoscedastic	1.24 $\pm$0.12	$-$3.41 $\pm$0.14	38.82 $\pm$2.66	58.84 $\pm$5.84	0.45 $\pm$0.05

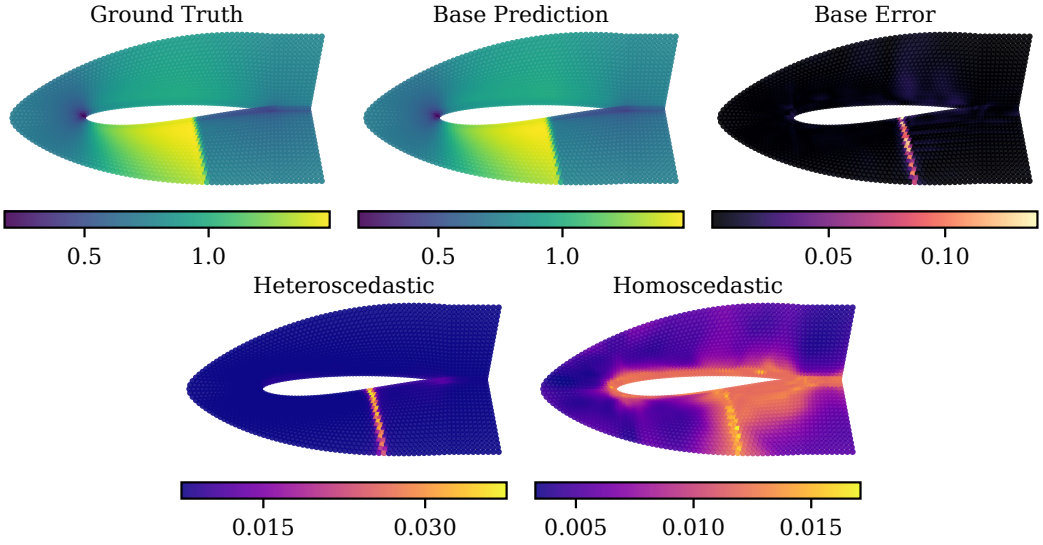

Figure 14: **Heteroscedastic vs. homoscedastic noise ablation on *Airfoil*.** Top: ground truth, base prediction, absolute error. Bottom: predicted standard deviation under each noise model. The heteroscedastic model concentrates uncertainty around the shock front where the operator's error is largest.

E Base Neural Operator Configuration

To ensure a rigorous evaluation of our post-hoc UQ framework, we train a deterministic Transolver [10] model as required for each of the benchmark tasks and baseline configurations. The architecture is configured based on the original setup to achieve near state-of-the-art predictive accuracy.

Transolver. The model architecture consists of a sequence of transformer blocks that rely on a Physics-Attention mechanism that adaptively groups the discretized spatial domain into learnable slices. Table 8 details the specific configurations used for the evaluated benchmarks. Across all tasks, we use the GELU [60] activation function and optimize the network using AdamW [61]. The learning rate is decayed using a OneCycle learning rate scheduler to ensure stable convergence. We use the relative L^2 error as the loss function. All models are trained for a fixed number of epochs, with results averaged over five random seeds to ensure statistical robustness.

Table 8: **Architecture and hyperparameter configurations.** The base Transolver slightly varies across the benchmarks.

Hyperparameter	**Elasticity**	**Airfoil**	**Pipe**	**ShapeNet Car**	**Ahmed Body**
Hidden channels (D_h)	128	128	128	256	256
Transformer layers (L)	8	8	8	8	8
Physics slices (S)	64	64	64	32	32
Attention heads (H)	8	8	8	8	8
Epochs	500	500	500	500	500
Batch size	1	4	4	1	1
Initial learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Weight decay	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}

Dataset Splits for UQ Evaluation. While standard operator learning benchmarks often assume abundant data, the value of UQ is most pronounced in data-constrained environments where the surrogate model is forced to generalize. To rigorously test the calibration of REEF-GP and the baselines under these conditions, we evaluate all main experiments using a reduced data split. Table 9 shows the data splits for each dataset.

Table 9: **Dataset splits.** The number of samples slightly varies across the five benchmarks.

	Elasticity	**Airfoil**	**Pipe**	**ShapeNet Car**	**Ahmed Body**
Train (M_{train})	500	500	500	500	400
Validation (M_{val})	100	100	100	100	50
Test (M_{test})	250	250	250	250	100

F Benchmarks

We use three 2D datasets and two 3D datasets to evaluate REEF-GP against competing methods detailed in Appendix G.

Elasticity. This benchmark provides the stress field of an elastic material subject to a tension traction $\mathbf{t}$ applied on the top edge of a unit cell with an irregular shape void in the interior [9]. There are a total of $M = 2000$ samples. For each sample the input domain is discretized as a point cloud comprising $N = 972$ spatial nodes.

Airfoil. This benchmark provides the resulting Mach number field from a transonic flow over 2D airfoil geometries, governed by the Euler equation [9]. The spatial domain is discretized into a structured mesh around the airfoil which we fix to $N = 2820$ points per sample. There are a total of $M = 2490$ samples.

Pipe. This benchmark provides the fluid velocity through a pipe of changing shape [9]. The input domain is discretized via $N = 129 \times 27$ spatial points and there are a total of $M = 2310$ samples.

ShapeNet Car. This benchmark provides the surface pressure field resulting from turbulent aerodynamic flow over diverse 3D vehicle geometries extracted from the "car" category of ShapeNet [62], governed by the Navier-Stokes equations [63]. The input domain for each vehicle surface is discretized as an unstructured point cloud, which we fix to $N = 3500$ spatial points per sample. There are a total of $M = 889$ samples.

Ahmed Body. This benchmark provides vehicle aerodynamics simulations based on the Ahmed-body shapes [28]. There are a total of $M = 551$ shapes and we fix the spatial points to $N = 5000$ per sample. As additional inputs we consider the inlet velocity and Reynolds number as they change across samples.

G UQ Baselines Configuration

To assess the performance of our post-hoc UQ framework, we compare it against six established UQ methodologies. To ensure a fair and rigorous comparison, all baselines are implemented using the exact same base Transolver architecture and evaluated over the same dataset splits.

Deep Ensembles. We ensemble 5 independent Transolver models, each of them trained from scratch using the same hyperparameters but initialized with a different random seed. The final predictive mean and variance are computed empirically from the ensemble's 5 distinct forward passes. While highly accurate, this method incurs a $5\times$ multiplier in total training cost.

Monte Carlo (MC) Dropout. Following standard practices, dropout is injected directly into the transformer blocks. Specifically, within the Physics-Attention mechanism, dropout is applied in two distinct locations: first, directly to the attention matrix after the softmax operation (randomly dropping query-key connections prior to multiplication with the value matrix), and second, to the output of the linear projection immediately following the deslicing step, before the tensor is passed to subsequent network components. We use a standard dropout rate of $p = 0.1$. At inference time, dropout is kept active, and we draw 50 stochastic forward passes to compute the predictive mean and variance.

Probabilistic Neural Operator (PNO). PNO [37] extends the MC Dropout approach by optimizing the network directly for probabilistic evaluation during training. Using Transolver as the base architecture with the identical dropout locations described above, we implement PNO by modifying the training objective. Instead of the deterministic relative L^2 error, PNO minimizes the empirical Energy Score (ES) defined in Equation 30 on the training set. To balance computational feasibility with stable training, we use $K = 5$ forward passes during training, and $K = 50$ forward passes during evaluation.

Function-Valued Laplace Approximation (LUNO-LA). We implement a last-layer Laplace approximation derived from the LUNO framework [20] using the `laplace-torch` library [19]. LUNO leverages the concept of *currying* to translate weight-space uncertainty into a continuous, function-valued Gaussian Process over the spatial domain. Because Transolver relies on standard pointwise linear projections in its final layers (rather than complex spectral convolutions), the network mapping is inherently linear with respect to the last layer's weights. Thus, applying a Laplace approximation to the final layer natively yields the exact LUNO covariance kernel $k(x, x') = \mathbf{h}(x)^\top \Sigma \mathbf{h}(x')$, where $\mathbf{h}(x)$ are the spatially-aligned Transolver features and Σ is the weight posterior covariance computed using a full Hessian structure. In our experiments we find that utilizing all spatial points makes covariance construction and inversion memory-prohibitive and unstable, preventing convergence to a robust solution. To overcome this bottleneck and ensure well-conditioned covariance inversion, we uniformly subsample a maximum of $50k$ spatial points across the dataset during optimization. The prior precision of the Laplace approximation is optimized via grid search on a validation set.

Input Perturbation. Drawing on test-time data augmentation [64] and initial-condition perturbation [1], we derive predictive uncertainty from a fully trained, deterministic Transolver model's sensitivity to input variations. At inference time, we inject independent and identically distributed (i.i.d.) Gaussian noise ($\sigma_n = 10^{-2}$) directly into the input tensors. For each test sample, we execute 50 independent forward passes, sampling a new noise mask for the inputs prior to each pass. The final predictive mean and variance are computed empirically across these 50 stochastic forward passes.

Deep Vecchia Ensemble (DVE-spatial). We adapt the Deep Vecchia Ensemble (DVE) [43], utilizing the authors' original implementation, which constructs an ensemble of GPs over the internal hidden-layer representations of a trained neural network. The original DVE computes the nearest-neighbor conditioning sets for its Vecchia approximation using distances strictly in the intermediate feature space. However, in dense spatial and operator learning tasks, geometrically neighboring points frequently exhibit nearly identical embeddings, which leads to severely ill-conditioned covariance matrices. To evaluate DVE as a viable baseline in our setting, we adapt it into *DVE-spatial* by concatenating the physical spatial coordinates to the intermediate embeddings prior to computing the distance metrics. Although DVE was designed for standard regression rather than operator learning and it might require further specialized modifications for this domain, it serves as a highly relevant benchmark for UQ applied directly within a network's feature space. Because the layer-wise nearest-neighbor searches and GP inference are computationally expensive, we configure DVE-spatial to balance its cost based on its default settings. Specifically, we restrict the Vecchia nearest-neighbor conditioning set size to $m = 8$ and utilize an approximate nearest neighbor search by partitioning the feature space into 500 cells and searching across the 15 closest cells for each query.

H Metrics

We evaluate predictive accuracy with the relative L^2 error, and probabilistic calibration with four standard scoring rules: NLL, CRPS, NIS, and ES. Every method is evaluated in the same physical (denormalized) units and on identical splits. Throughout, M_{test} denotes the number of test instances; for instance $\mathbf{v}_i \in \mathbb{R}^N$ is the ground-truth nodal solution with entries $v_{i,j}$, and the predictive distribution at node j is Gaussian with mean $\mu_{i,j}$ and variance $\sigma^2_{i,j}$.

Relative L^2 Error. Quantifies the predictive accuracy of the operator:

$$\mathrm{rL2} = \frac{1}{M_{\text{test}}} \sum_{i=1}^{M_{\text{test}}} \frac{\|\mathcal{G}^\dagger(u_i, a_i) - \mathcal{G}_\theta(\mathbf{u}_i, \mathbf{a}_i)\|_{\mathcal{V}}}{\|\mathcal{G}^\dagger(u_i, a_i)\|_{\mathcal{V}}}, \tag{26}$$

where $\|\cdot\|_{\mathcal{V}}$ is the L^2 norm over the physical domain.

Negative Log-Likelihood (NLL). A strictly proper scoring rule that penalizes both miscalibration and overconfidence:

$$\mathrm{NLL} = \frac{1}{M_{\text{test}} N} \sum_{i=1}^{M_{\text{test}}} \sum_{j=1}^{N} \left[\frac{1}{2} \log(2\pi\sigma^2_{i,j}) + \frac{(v_{i,j} - \mu_{i,j})^2}{2\sigma^2_{i,j}} \right]. \tag{27}$$

Continuous Ranked Probability Score (CRPS). A probabilistic generalization of the mean absolute error. For Gaussian predictives it admits the closed form:

$$\mathrm{CRPS} = \frac{1}{M_{\text{test}} N} \sum_{i=1}^{M_{\text{test}}} \sum_{j=1}^{N} \sigma_{i,j} \left[z_{i,j} \big(2\Phi(z_{i,j}) - 1\big) + 2\phi(z_{i,j}) - \tfrac{1}{\sqrt{\pi}} \right], \tag{28}$$

with standardized error $z_{i,j} = (v_{i,j} - \mu_{i,j})/\sigma_{i,j}$, and Φ, ϕ the standard-normal CDF and PDF.

Negatively Oriented Interval Score (NIS). Evaluates the quality of central prediction intervals. For confidence level $1 - \alpha$, let $L_{i,j}$ and $U_{i,j}$ be the predicted lower and upper bounds at node j:

$$\mathrm{NIS} = \frac{1}{M_{\text{test}} N} \sum_{i=1}^{M_{\text{test}}} \sum_{j=1}^{N} \left[(U_{i,j} - L_{i,j}) + \frac{2}{\alpha}(L_{i,j} - v_{i,j})_+ + \frac{2}{\alpha}(v_{i,j} - U_{i,j})_+ \right], \tag{29}$$

where $(x)_+ = \max(x, 0)$. The first term rewards sharpness; the latter two penalize miscoverage. We report NIS at $\alpha = 0.05$.

Energy Score (ES). A multivariate proper scoring rule, estimated by Monte Carlo with $K = 50$ samples $\mathbf{v}_i^{(k)}$ drawn from the predictive:

$$\mathrm{ES} = \frac{1}{M_{\text{test}}} \sum_{i=1}^{M_{\text{test}}} \left[\frac{1}{K} \sum_{k=1}^{K} \|\mathbf{v}_i^{(k)} - \mathbf{v}_i\|_2 - \frac{1}{2K(K-1)} \sum_{k \neq k'} \|\mathbf{v}_i^{(k)} - \mathbf{v}_i^{(k')}\|_2 \right]. \tag{30}$$

The first term rewards accuracy; the second penalizes overconfidence by rewarding sample diversity.

I Additional Results

This section presents the results of REEF-GP and the UQ baselines on the remaining three benchmarks: *Elasticity* (2D), *Pipe* (2D), and *Ahmed Body* (3D). The quantitative results are reported in Table 10, while spatial uncertainty visualizations and compute cost trade-offs are provided in Figures 15–17 and Figure 18, respectively.

Consistent preservation of predictive accuracy. REEF-GP successfully preserves the predictive accuracy of the base Transolver across all three additional datasets. As shown in Table 10, the rL2 of REEF-GP matches that of the deterministic Base model perfectly. In contrast, train-time modifications such as MC Dropout, PNO, and DVE-spatial consistently suffer from degraded rL2 performance.

Competitive probabilistic performance. REEF-GP remains highly competitive across all probabilistic metrics. Most notably, on the complex 3D *Ahmed Body* dataset, REEF-GP achieves an NLL of 4.21, effectively tying with the computationally expensive Deep Ensembles (4.20) while outperforming all other post-hoc and train-time baselines. On the 2D benchmarks (*Elasticity* and *Pipe*), REEF-GP yields well-calibrated posteriors with NLL and CRPS scores closely tracking the top-performing methods.

Spatial alignment of uncertainty. Figures 15, 16, and 17 further confirm the spatial coherence of REEF-GP. The predicted standard deviation tracks the localized prediction errors of the base operator, although in *Elasticity* benchmark our REEF-GP uncertainty is quite regular across the entire domain.

Cost-quality tradeoff. Figure 18 compares the compute cost across all UQ methods on all available benchmarks. It confirms that REEF-GP consistently achieves a training overhead an order of magnitude lower than Deep Ensembles while remaining competitive on evaluation time and train memory. The only tradeoff is in evaluation memory, although it remains well within commodity GPU memory.

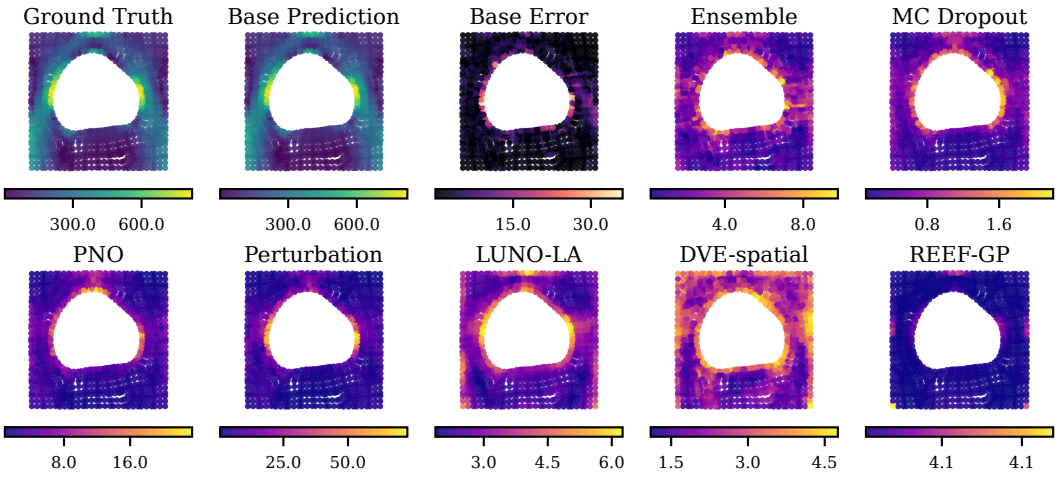

Figure 15: **Predictive standard deviation fields on *Elasticity*.** The rows show in order: ground truth, base prediction, base error and the standard deviation of a sample for each of the UQ baselines.

Table 10: **Evaluation metrics for the three benchmarks not in the main text.** Lower is better (↓). Best in **bold**, second best underlined. Rankings exclude Base. See Appendix H for the definition of the metrics.

Method	**rL2** ↓	**NLL** ↓	**CRPS** ↓	**NIS** ↓	**ES** ↓
Elasticity (2D)	(%)				
Base (no UQ)	1.68 ±0.16	–	–	–	–
Ensemble	**1.24**	**2.45**	**1.30**	**13.88**	**53.73**
MC Dropout	1.58 ±0.12	19.87 ±3.22	2.06 ±0.14	59.76 ±4.89	97.29 ±7.60
PNO	2.39 ±0.14	3.08 ±0.07	2.65 ±0.15	30.58 ±2.04	118.18 ±7.12
Perturbation	2.17 ±0.14	2.85 ±0.05	2.43 ±0.10	29.13 ±0.71	125.12 ±4.99
LUNO-LA	1.68 ±0.16	2.80 ±0.36	1.85 ±0.18	25.32 ±5.82	87.82 ±10.50
DVE-spatial	5.25 ±0.23	10.48 ±0.73	6.36 ±0.30	162.11 ±10.59	317.43 ±15.28
REEF-GP	1.68 ±0.16	2.83 ±0.10	1.97 ±0.15	25.94 ±2.70	86.37 ±7.96
Pipe (2D)	(%)		($\times 10^{-4}$)	($\times 10^{-3}$)	($\times 10^{-2}$)
Base (no UQ)	0.52 ±0.02	–	–	–	–
Ensemble	**0.42**	**−6.32**	**2.61**	**3.68**	**3.38**
MC Dropout	0.64 ±0.03	−5.58 ±0.08	5.12 ±0.20	6.96 ±0.28	5.78 ±0.29
PNO	0.77 ±0.05	−5.12 ±0.05	7.77 ±0.36	11.07 ±0.51	7.68 ±0.46
Perturbation	2.49 ±0.25	−4.22 ±0.10	21.18 ±1.59	26.35 ±0.81	22.28 ±1.35
LUNO-LA	0.52 ±0.02	−4.63 ±0.15	10.11 ±1.30	16.61 ±2.16	8.63 ±0.90
DVE-spatial	3.69 ±0.89	0.60 ±1.60	33.98 ±8.33	69.72 ±12.12	35.29 ±7.82
REEF-GP	0.52 ±0.02	−4.73 ±0.02	8.19 ±0.09	13.85 ±0.29	7.21 ±0.16
Ahmed Body (3D)	(%)				($\times 10^{3}$)
Base (no UQ)	6.62 ±0.05	–	–	–	–
Ensemble	**5.98**	**4.20**	**6.27**	**104.01**	**1.03**
MC Dropout	6.94 ±0.42	15.10 ±5.25	8.96 ±1.42	227.43 ±37.79	1.34 ±0.07
PNO	11.59 ±0.94	8.38 ±1.68	15.43 ±1.63	244.62 ±30.07	1.81 ±0.15
Perturbation	7.51 ±0.14	6.88 ±0.80	8.43 ±0.11	144.79 ±3.94	1.24 ±0.02
LUNO-LA	6.62 ±0.05	16.95 ±10.78	8.07 ±0.32	206.69 ±50.88	1.34 ±0.09
DVE-spatial	13.54 ±1.59	7.90 ±1.72	17.66 ±1.52	400.41 ±24.53	2.65 ±0.24
REEF-GP	6.63 ±0.05	4.21 ±0.08	8.88 ±0.79	136.68 ±3.12	1.19 ±0.01

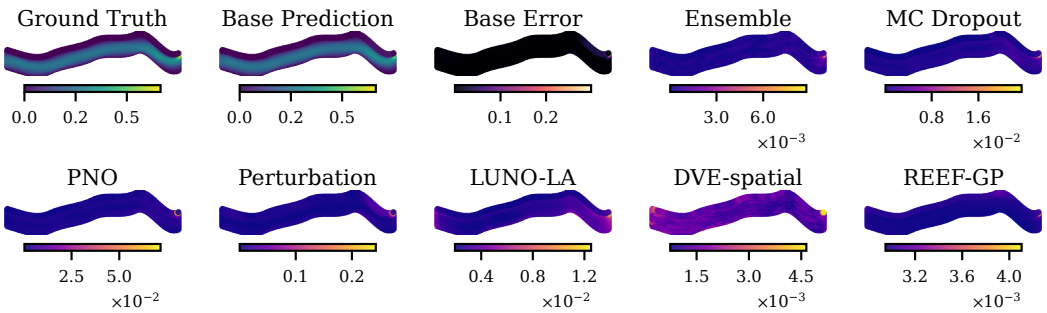

Figure 16: **Predictive standard deviation fields on *Pipe*.** The rows show in order: ground truth, base prediction, base error and the standard deviation of a sample for each of the UQ baselines.

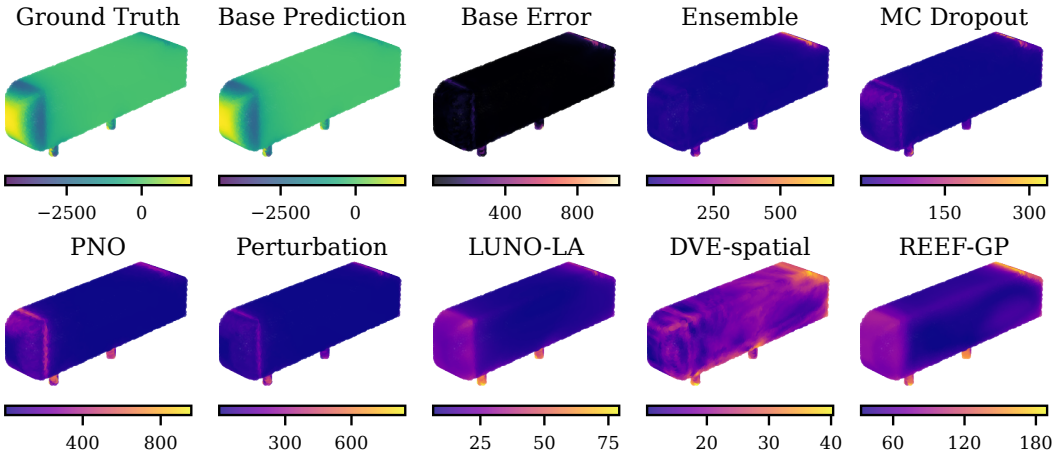

Figure 17: **Predictive standard deviation fields on *Ahmed*.** The rows show in order: ground truth, base prediction, base error and the standard deviation of a sample for each of the UQ baselines.

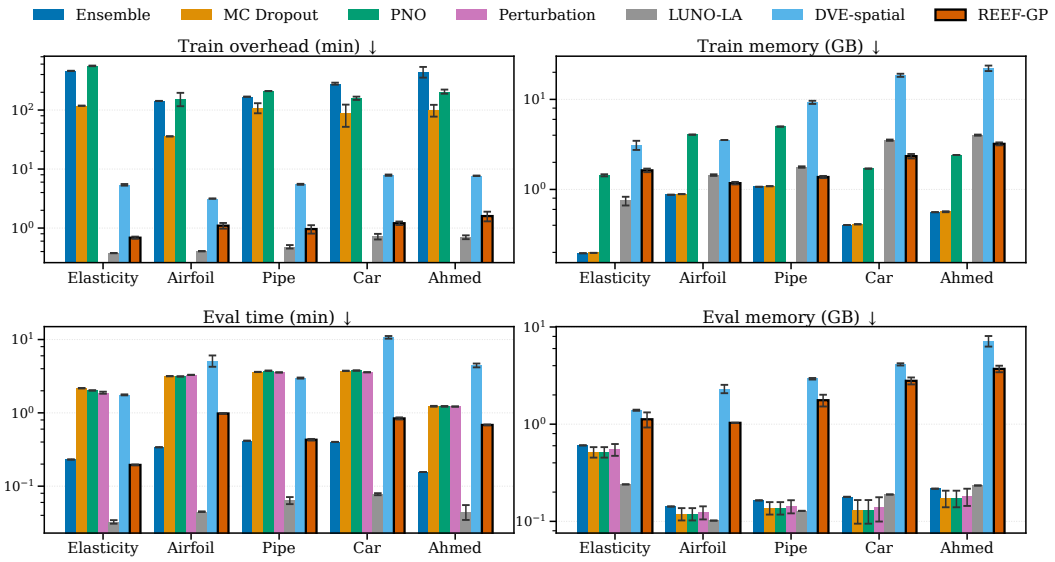

Figure 18: **Compute costs on all benchmarks:** Train overhead (the additional time needed on top of a trained base operator to enable UQ; for REEF-GP it includes feature extraction, CKA layer selection, and GP fitting), evaluation time, peak train memory, and peak evaluation memory on all benchmarks. REEF-GP maintains an excellent balance between training overhead and inference speed, while peak evaluation memory is higher.

J Geometric OOD

Maximum Mean Discrepancy (MMD). Given two point clouds $\mathcal{X}$ and $\mathcal{Y}$, each comprising P points, the squared MMD is defined as:

$$MMD^2(\mathcal{X}, \mathcal{Y}) = \frac{1}{P^2}\sum_{i=1}^{P}\sum_{j=1}^{P} k(x_i, x_j) + \frac{1}{P^2}\sum_{i=1}^{P}\sum_{j=1}^{P} k(y_i, y_j) - \frac{2}{P^2}\sum_{i=1}^{P}\sum_{j=1}^{P} k(x_i, y_j) \quad (31)$$

where $k(\cdot, \cdot)$ denotes the Gaussian (Radial Basis Function) kernel, given by $k(x, y) = \exp\left(-\frac{||x-y||^2}{2l^2}\right)$. To appropriately scale the kernel to the data distribution, the lengthscale parameter l is determined via a median heuristic computed as the median of the pairwise L_2 distances sampled from the training point clouds.

Table 11: **OOD robustness across MMD-based geometric quantiles.** Test samples are partitioned into four groups by MMD distance from the training distribution: Q1 (closest, in-distribution) through Q4 (furthest, most geometrically novel). For each method we report rL2 (↓) and NLL (↓); rL2 in percent. Best in **bold**, second best underlined.

Method	Q1		Q2		Q3		Q4	
	rL2 (%)	NLL	rL2 (%)	NLL	rL2 (%)	NLL	rL2 (%)	NLL
Elasticity (2D)								
Ensemble	**1.08**	**2.59**	**1.09**	**2.23**	**1.35**	**2.43**	**1.42**	**2.56**
MC Dropout	1.39 ±0.11	19.04 ±3.48	1.42 ±0.12	19.66 ±3.32	1.72 ±0.13	20.05 ±2.93	1.81 ±0.12	20.76 ±3.22
PNO	2.20 ±0.15	2.93 ±0.09	2.23 ±0.13	3.00 ±0.07	2.51 ±0.15	3.14 ±0.08	2.62 ±0.13	3.25 ±0.06
Perturbation	1.99 ±0.13	2.71 ±0.04	2.01 ±0.14	2.75 ±0.04	2.30 ±0.14	2.91 ±0.07	2.40 ±0.16	3.02 ±0.07
LUNO-LA	1.49 ±0.15	**2.59** ±0.28	1.52 ±0.16	2.60 ±0.30	1.82 ±0.16	2.93 ±0.41	1.92 ±0.18	3.09 ±0.46
DVE-spatial	4.78 ±0.19	9.69 ±0.59	4.89 ±0.20	9.68 ±0.69	5.44 ±0.27	10.71 ±0.82	5.88 ±0.27	11.88 ±0.84
REEF-GP	1.49 ±0.15	2.73 ±0.09	1.52 ±0.16	2.69 ±0.08	1.82 ±0.16	2.92 ±0.11	1.92 ±0.18	2.97 ±0.13
Airfoil (2D)								
Ensemble	**0.85**	−3.60	**0.95**	**−3.56**	**1.13**	**−3.49**	**1.31**	−3.19
MC Dropout	1.38 ±0.22	−3.49 ±0.13	1.40 ±0.19	−3.45 ±0.13	1.60 ±0.18	−3.36 ±0.14	1.94 ±0.27	−3.22 ±0.20
PNO	2.06 ±1.40	−3.36 ±0.65	2.12 ±1.31	−3.36 ±0.63	2.39 ±1.47	−3.26 ±0.68	2.73 ±1.41	−3.13 ±0.63
Perturbation	3.20 ±0.19	−3.35 ±0.05	3.13 ±0.16	−3.36 ±0.05	3.41 ±0.19	−3.31 ±0.05	3.60 ±0.22	−3.23 ±0.06
LUNO-LA	1.03 ±0.10	−3.37 ±0.52	1.13 ±0.13	−3.28 ±0.57	1.30 ±0.11	−2.92 ±1.04	1.51 ±0.17	−2.68 ±1.16
DVE-spatial	1.88 ±0.27	17.36 ±6.55	1.97 ±0.26	18.01 ±6.45	2.14 ±0.25	20.48 ±7.39	2.38 ±0.20	24.14 ±9.44
REEF-GP	1.03 ±0.10	**−3.61** ±0.05	1.13 ±0.13	**−3.56** ±0.07	1.30 ±0.11	−3.47 ±0.07	1.51 ±0.17	**−3.41** ±0.07
Pipe (2D)								
Ensemble	**0.38**	**−6.27**	**0.31**	**−6.63**	**0.39**	**−6.58**	**0.60**	**−5.80**
MC Dropout	0.54 ±0.05	−5.59 ±0.06	0.52 ±0.04	−5.80 ±0.04	0.68 ±0.07	−5.68 ±0.11	0.84 ±0.02	−5.25 ±0.16
PNO	0.65 ±0.06	−5.20 ±0.04	0.63 ±0.05	−5.20 ±0.04	0.77 ±0.06	−5.15 ±0.04	1.04 ±0.07	−4.92 ±0.07
Perturbation	2.06 ±0.19	−4.33 ±0.08	2.48 ±0.24	−4.29 ±0.09	2.48 ±0.26	−4.22 ±0.11	2.97 ±0.30	−4.05 ±0.11
LUNO-LA	0.47 ±0.05	−4.69 ±0.15	0.40 ±0.03	−4.67 ±0.16	0.49 ±0.06	−4.64 ±0.16	0.72 ±0.03	−4.54 ±0.14
DVE-spatial	3.34 ±0.95	0.12 ±1.04	3.73 ±0.84	0.72 ±1.77	3.55 ±0.90	0.26 ±1.54	4.13 ±0.88	1.31 ±2.16
REEF-GP	0.47 ±0.05	−4.72 ±0.05	0.40 ±0.03	−4.85 ±0.03	0.49 ±0.06	−4.79 ±0.05	0.72 ±0.03	−4.57 ±0.03
ShapeNet Car (3D)								
Ensemble	**5.81**	3.13	**6.84**	4.26	**8.55**	4.48	**9.77**	4.81
MC Dropout	7.19 ±0.74	14.77 ±4.73	8.02 ±0.53	21.25 ±5.54	9.84 ±0.54	17.03 ±3.67	11.15 ±0.57	15.86 ±3.09
PNO	12.50 ±1.71	4.35 ±1.24	12.08 ±1.45	4.54 ±1.21	14.61 ±1.83	5.01 ±1.47	15.89 ±2.03	6.04 ±2.19
Perturbation	6.65 ±0.13	6.61 ±0.65	7.62 ±0.13	10.79 ±1.14	9.48 ±0.15	15.17 ±1.70	10.79 ±0.13	21.41 ±1.77
LUNO-LA	6.59 ±0.11	19.79 ±13.47	7.64 ±0.13	30.44 ±21.51	9.51 ±0.17	37.41 ±25.97	10.79 ±0.12	46.99 ±33.15
DVE-spatial	10.67 ±0.29	8.27 ±1.90	11.12 ±0.27	9.66 ±2.65	14.17 ±0.23	12.30 ±3.51	16.46 ±0.29	14.75 ±4.15
REEF-GP	6.47 ±0.12	**2.29** ±0.02	7.54 ±0.14	**2.68** ±0.09	9.42 ±0.18	**3.00** ±0.11	10.71 ±0.14	**3.42** ±0.17
Ahmed Body (3D)								
Ensemble	**5.16**	**3.90**	**6.35**	4.77	**5.83**	4.03	**6.60**	**4.09**
MC Dropout	6.21 ±0.46	10.75 ±3.22	7.21 ±0.46	18.75 ±5.92	6.71 ±0.39	13.13 ±3.39	7.61 ±0.43	17.75 ±8.58
PNO	11.05 ±1.05	5.86 ±0.80	11.63 ±0.95	12.70 ±3.55	11.61 ±0.78	5.13 ±0.52	12.07 ±1.10	9.84 ±2.26
Perturbation	6.74 ±0.26	5.42 ±0.45	7.98 ±0.13	7.97 ±1.15	7.15 ±0.18	6.47 ±0.68	8.15 ±0.17	7.66 ±1.39
LUNO-LA	5.74 ±0.12	10.12 ±5.77	6.94 ±0.08	26.00 ±17.84	6.48 ±0.16	9.45 ±5.13	7.33 ±0.23	22.22 ±14.74
DVE-spatial	13.17 ±1.52	8.02 ±1.49	13.93 ±1.68	8.93 ±2.19	13.00 ±1.48	6.14 ±1.12	14.05 ±1.69	8.50 ±2.10
REEF-GP	5.82 ±0.15	4.06 ±0.15	6.92 ±0.09	**4.34** ±0.08	6.49 ±0.17	**4.01** ±0.16	7.29 ±0.23	4.42 ±0.06

OOD Evaluation Results. Table 11 reports the quantitative performance of all baselines partitioned across the four geometric MMD quartiles. A robust UQ method should seamlessly scale its predictive uncertainty as the test geometry deviates from the training manifold, resulting in a stable NLL even as the deterministic error (rL2) naturally increases.

Across all benchmarks, REEF-GP demonstrates strong out-of-distribution stability. For instance, in the *ShapeNet Car* dataset, as geometries shift from Q1 (most similar to the training samples) to Q4 (most different from training samples), REEF-GP's NLL increases moderately and it remains the best-calibrated method in every quartile. In contrast, other post-hoc methods degrade much more under identical shifts: Perturbation's NLL more than triples between Q1 and Q4 and LUNO-LA's more than doubles, and similar trends are observed on the Ahmed Body benchmark. Furthermore, REEF-GP consistently delivers competitive metrics with deep ensembles.

Qualitative OOD Uncertainty. Figures 19 through 22 and Figures 23 through 26 visualize the spatial distribution of uncertainty for the 3D benchmarks as the test geometries become increasingly distinct from the training set (progressing towards Q4). REEF-GP's predicted uncertainty remains coherent with the base model error regions over the geometry.

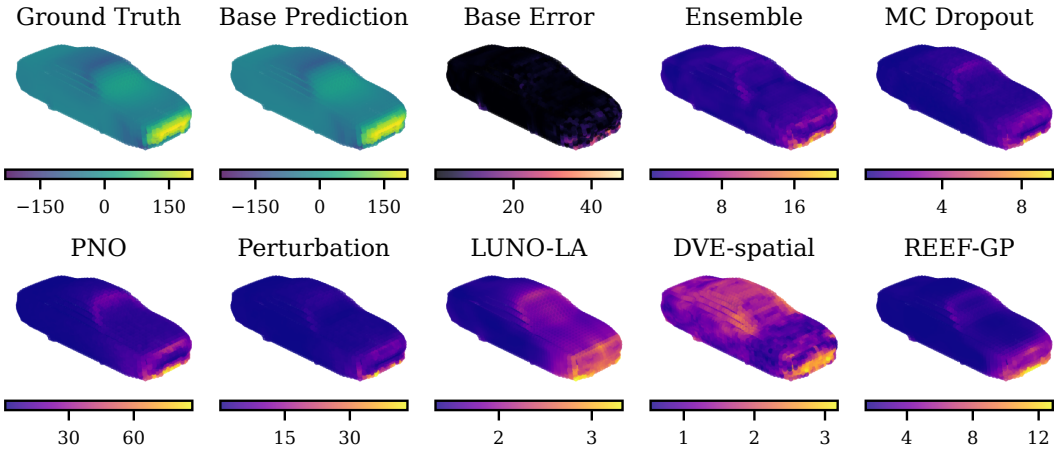

Figure 19: Predictive standard deviation fields on a test *Car* sample from Q1.

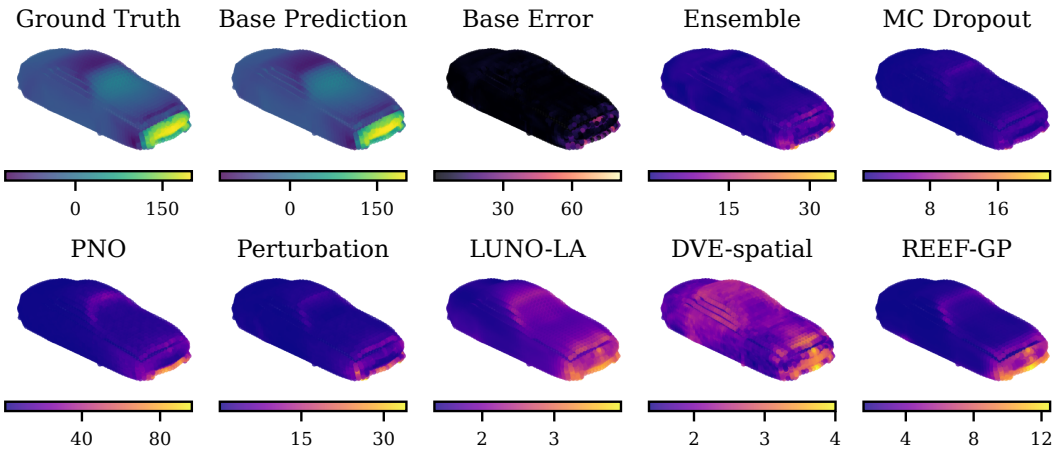

Figure 20: Predictive standard deviation fields on a test *Car* sample from Q2.

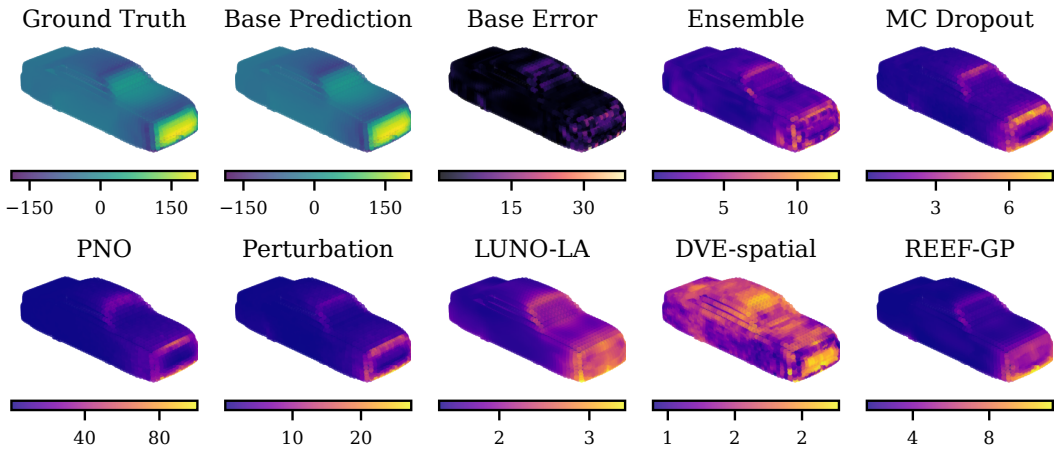

Figure 21: Predictive standard deviation fields on a test *Car* sample from Q3.

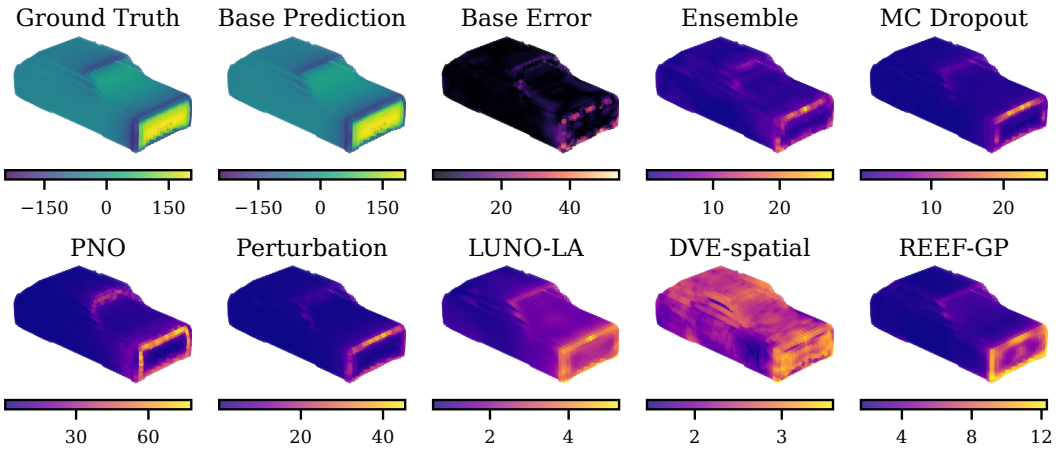

Figure 22: Predictive standard deviation fields on a test *Car* sample from Q4.

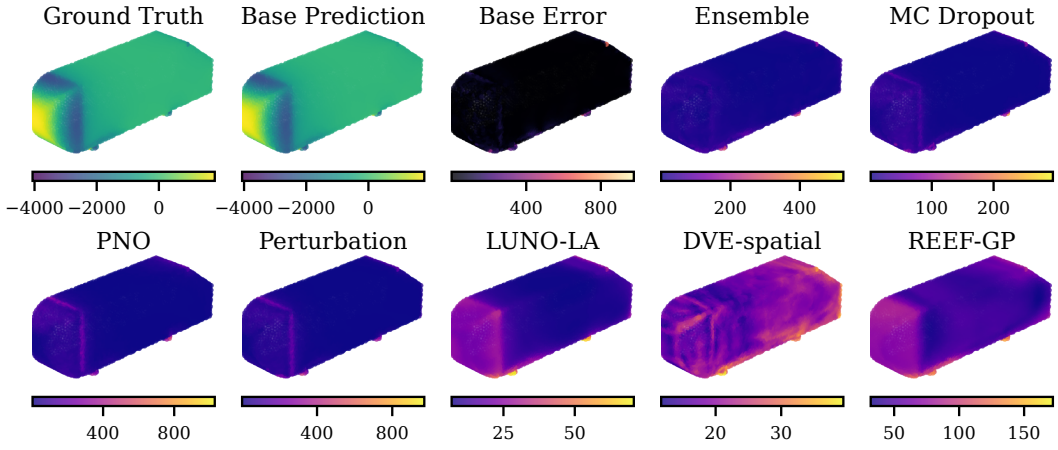

Figure 23: Predictive standard deviation fields on a test *Ahmed* sample from Q1.

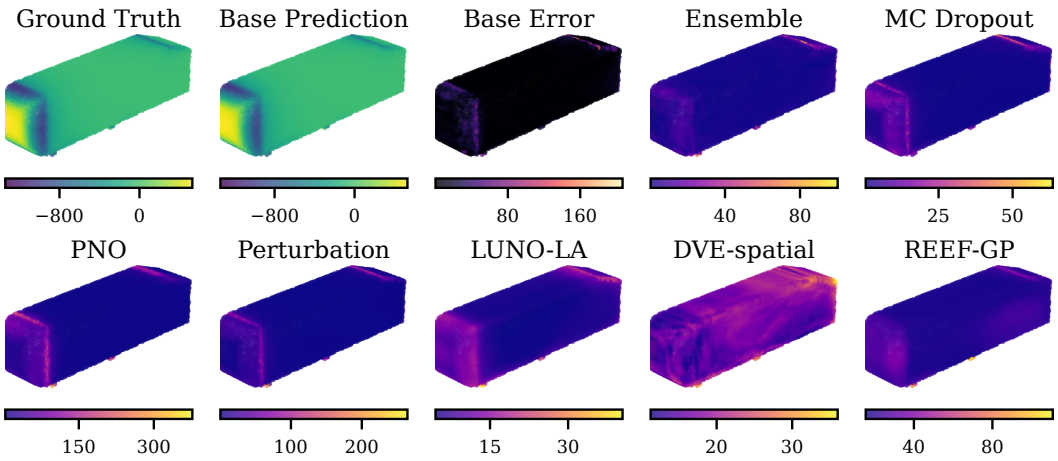

Figure 24: Predictive standard deviation fields on a test *Ahmed* sample from Q2.

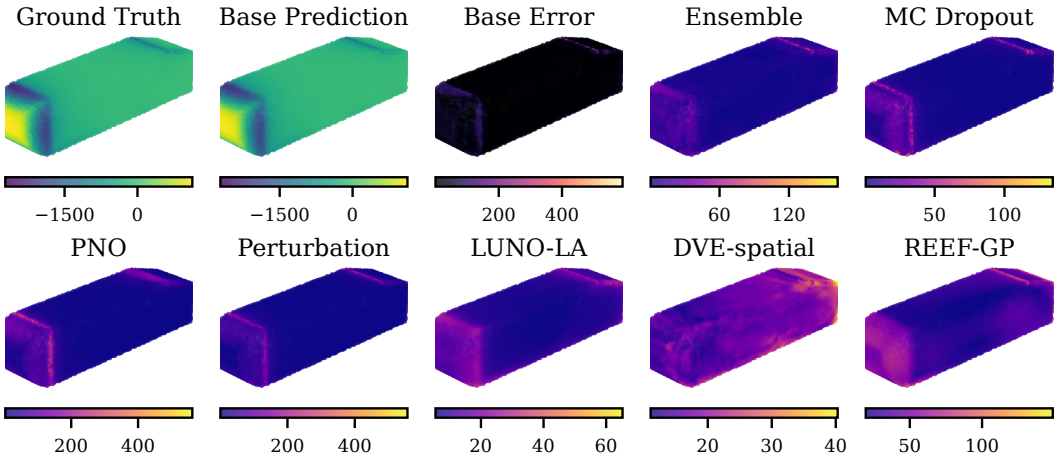

Figure 25: Predictive standard deviation fields on a test *Ahmed* sample from Q3.

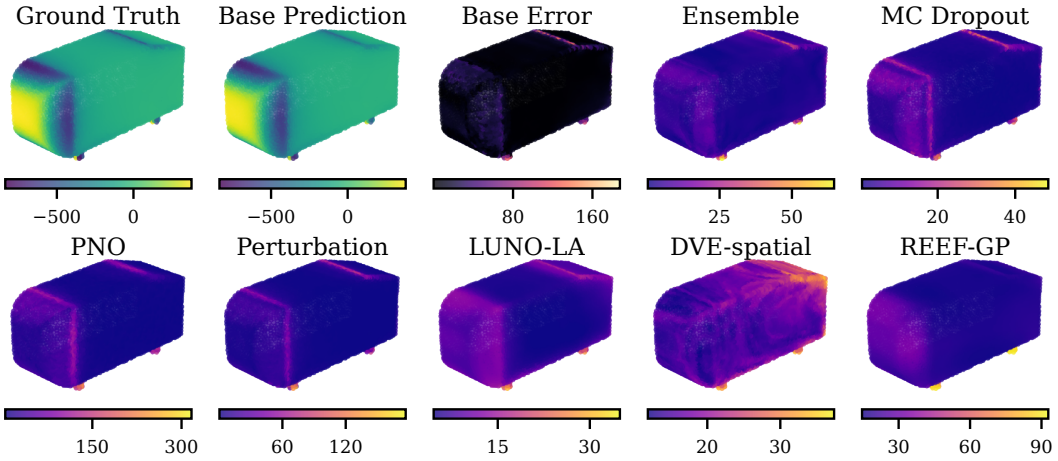

Figure 26: Predictive standard deviation fields on a test *Ahmed* sample from Q4.

J.1 Extrapolation Studies

The MMD quartiles above stratify a test set that is drawn from the same benchmark distribution as the training set. To probe stronger shifts, we construct two splits in which every test sample lies outside the range covered by training.

Reynolds-number extrapolation (*Ahmed Body*). We sort the samples by Reynolds number and hold out the 100 with the highest values as the test set ($\mathrm{Re} \geq 3.79 \times 10^6$). The training (400) and validation (50) sets are drawn at random from the remaining samples ($\mathrm{Re} \leq 3.78 \times 10^6$), so model selection never sees shifted data. The test set also has a higher mean inflow velocity (53.0 vs. 34.8 m/s) and longer bodies. The base Transolver rL2 increases from 6.62% on the original split to 9.22%.

Shape-novelty extrapolation (*ShapeNet Car*). For every geometry we compute a shape-novelty score, which is the mean MMD between its point cloud and a fixed set of 40 reference shapes drawn at random from the dataset. The score does not depend on the training set, which avoids defining the split by distance to training data. The 250 shapes with the highest score form the test set, so all test shapes exceed the score range covered in training, and the training (500) and validation (100) sets are drawn at random from the rest. The base Transolver rL2 increases from 8.62% to 11.39%. This increases the average geometric MMD between test and training sets by about $8.9\times$ relative to the original split.

In both cases the test set is stratified into quartiles Q1–Q4 along the extrapolation factor (Reynolds number or shape-novelty score), from least to most extreme. To summarize sensitivity to extrapolation, we report the drift ratio $r_{\text{metric}} = \text{metric}_{\text{Q4}}/\text{metric}_{\text{Q1}}$ for rL2, CRPS, NIS, and ES, where a ratio close to 1 indicates robustness. For NLL we report the difference $\Delta\text{NLL} = \text{NLL}_{\text{Q4}} - \text{NLL}_{\text{Q1}}$ instead, because NLL shifts by an additive constant when the outputs are rescaled, so its ratio depends on the units. Since the drift ratio can be small for a method that is already poorly calibrated in Q1, we also report the absolute NLL in Q1 and Q4.

Results. Table 12 summarizes the results. All methods degrade under extrapolation. On *Ahmed Body*, REEF-GP has the lowest absolute NLL in both Q1 and Q4 and a smaller NLL increase than Deep Ensembles ($+2.37$ vs. $+4.70$), whereas Perturbation collapses ($+313$). On *ShapeNet Car*, REEF-GP remains far better calibrated than Perturbation in both quartiles, while Deep Ensembles show the smallest degradation and the best Q4 NLL. Accuracy drift (rL2) is nearly identical for all methods, confirming that the differences stem from the quality of the predicted uncertainty rather than from the mean prediction.

Table 12: **Extrapolation studies.** Drift from Q1 to Q4: ratio $\text{metric}_{\text{Q4}}/\text{metric}_{\text{Q1}}$ for rL2, CRPS, NIS, and ES, difference $\text{NLL}_{\text{Q4}} - \text{NLL}_{\text{Q1}}$ for NLL (lower means more robust), and absolute NLL in Q1 and Q4. Values are computed from metrics averaged over five seeds; the Ensemble is a single aggregate. Best per column in **bold**.

	Drift Q1 → Q4 ↓					**NLL** ↓
Method	rL2	ΔNLL	CRPS	NIS	ES	Q1 → Q4
Ahmed Body (3D), Reynolds-number extrapolation						
Ensemble	1.30	+4.70	2.35	3.30	1.89	7.91 → 12.61
Perturbation	**1.28**	+313.1	**2.19**	**2.91**	**1.87**	42.7 → 355.8
REEF-GP	1.33	**+2.37**	2.33	3.43	1.93	**4.85 → 7.22**
ShapeNet Car (3D), shape-novelty extrapolation						
Ensemble	**1.41**	**+0.87**	**1.66**	**1.62**	**1.46**	4.07 → **4.94**
Perturbation	1.42	+26.0	1.73	2.13	1.57	13.3 → 39.3
REEF-GP	**1.41**	+3.35	1.72	2.15	1.53	**4.00** → 7.35

K Application to Other Backbones

REEF-GP only requires hidden features that are aligned with the query points (Section 3), which Transolver provides natively. Here we show how to obtain them for an operator whose hidden states are not point-aligned, using the Factorized Fourier Neural Operator (FFNO [52]) with a learned geometric deformation (Geo-FNO [9]) on *Airfoil*. We denote it as Geo-FFNO.

Point-aligned features via resampling. Geo-FFNO maps the physical domain D_a to a regular latent grid through a coordinate map $\boldsymbol{\xi}_a : D_a \to [0,1]^d$ and applies its factorized spectral layers on that grid. We index the feature positions $l = 0, \dots, L-1$: $\mathbf{h}_0$ is the lifting output, $\mathbf{h}_{L-1}$ is the output projected back to the point cloud, and the positions $l = 1, \dots, L-2$ are the latent-grid states ($L = 7$ for the four spectral layers used here). The hidden state at an interior position l is therefore a grid tensor $\mathbf{H}_l^{\text{grid}} \in \mathbb{R}^{s_1 \times s_2 \times D_h}$ rather than one feature per mesh point, where $s_1 = s_2 = 96$ and $D_h = 64$ on *Airfoil*. We recover a point-aligned feature by evaluating the grid tensor at the latent location of each query point:

$$\mathbf{h}_l(\mathbf{u}, \mathbf{a}, \mathbf{x}) = \mathcal{I}\big[\mathbf{H}_l^{\text{grid}}\big]\big(\boldsymbol{\xi}_a(\mathbf{x})\big), \tag{32}$$

where $\mathcal{I}$ is bilinear interpolation on the $s_1 \times s_2$ latent grid and $\boldsymbol{\xi}_a$ is the coordinate map that Geo-FFNO itself learns, so no new component is introduced. Points that $\boldsymbol{\xi}_a$ maps outside $[0,1]^d$ take the value at the nearest border. The first and last features (the lifting output and the output projected back to the point cloud) already live on the mesh points and need no resampling; only interior layers are resampled. CKA layer selection is run after resampling, so every candidate is compared on the same N points. The resulting features are used exactly as in the Transolver case. We consider two layer sets: $\{\mathbf{h}_0, \mathbf{h}_{L-1}\}$ and the CKA-selected $\{\mathbf{h}_0, \mathbf{h}_{l^*}, \mathbf{h}_{L-1}\}$.

Results. Table 13 compares REEF-GP with a five-member Geo-FFNO Deep Ensemble. As with Transolver, REEF-GP preserves the base accuracy exactly, and the CKA-selected three-layer variant is slightly better than the two-layer one (which we include as it can be applied without resampling to any operator whose first and last layers act pointwise). The gap to the ensemble is larger than with Transolver, which we attribute to Geo-FFNO being a less accurate base operator (2.97% vs. 1.24% rL2) whose residuals are harder to model. We present this experiment as evidence of portability and a comparison against other non-point-aligned backbones is left for future work. We use Transolver in the main text because it is the more accurate base operator in our experiments.

Table 13: **REEF-GP with a Geo-FFNO backbone on *Airfoil*.** Lower is better (↓). ± is the standard deviation over five seeds; Ensemble rows aggregate the five seeds and are single values. Transolver results from Table 1 are repeated for reference.

Method	**rL2** ↓	**NLL** ↓	**CRPS** ↓	**NIS** ↓	**ES** ↓
Geo-FFNO backbone	(%)		($\times 10^{-4}$)	($\times 10^{-2}$)	
Base Geo-FFNO (no UQ)	2.97 ±0.20	–	–	–	–
Ensemble (Geo-FFNO)	**2.38**	**−3.30**	**67.80**	**7.89**	**0.75**
REEF-GP $\{\mathbf{h}_0, \mathbf{h}_{L-1}\}$	2.97 ±0.20	−2.24 ±0.29	101.20 ±7.60	16.47 ±2.25	1.11 ±0.09
REEF-GP $\{\mathbf{h}_0, \mathbf{h}_{l^*}, \mathbf{h}_{L-1}\}$	2.97 ±0.20	−2.28 ±0.30	100.87 ±7.55	16.16 ±2.30	1.11 ±0.09
Transolver backbone (Table 1)					
Base Transolver (no UQ)	1.24 ±0.12	–	–	–	–
Ensemble (Transolver)	**1.06**	**−3.46**	**28.63**	**3.89**	**0.35**
REEF-GP (Transolver)	1.24 ±0.12	−3.51 ±0.06	39.80 ±2.35	5.69 ±0.47	0.45 ±0.05

L Multi-Output Extension

Many PDE solutions are vector-valued (e.g., velocity and pressure). We extend REEF-GP to P output channels with an intrinsic coregionalization model (ICM). Let $\delta_p(\bar{\mathbf{z}})$ denote the discrepancy of channel $p \in \{1, \dots, P\}$. The latent residuals share the REEF-GP product kernel $k_{\boldsymbol{\phi}}$ (Section 3.2) and are coupled through a coregionalization matrix $\mathbf{B} \in \mathbb{R}^{P \times P}$:

$$\mathrm{Cov}\left[f_p(\bar{\mathbf{z}}), f_{p'}(\bar{\mathbf{z}}')\right] = B_{pp'}\, k_{\boldsymbol{\phi}}(\bar{\mathbf{z}}, \bar{\mathbf{z}}'), \qquad \mathbf{B} = \mathbf{L}\mathbf{L}^\top + \epsilon \mathbf{I}, \tag{33}$$

where $\mathbf{L} \in \mathbb{R}^{P \times P}$ is a learned lower-triangular matrix and $\epsilon = 10^{-5}$, which guarantees $\mathbf{B}$ is positive definite. $\mathbf{B}$ is therefore full rank. For a subset of N_s points, the joint covariance of all channels is

$$\mathbf{K}^{\mathrm{MO}} = \mathbf{B} \otimes k_{\boldsymbol{\phi}}(\bar{\mathbf{Z}}, \bar{\mathbf{Z}}) + \mathrm{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{\psi}}(\bar{\mathbf{Z}})\right) + \sigma_n^2 \mathbf{I}, \tag{34}$$

where the heteroscedastic noise network now outputs one variance per channel. Training and gPoE inference proceed as in Appendices B.4 and B.5 with the $N_s P \times N_s P$ matrix $\mathbf{K}^{\mathrm{MO}}$. Because the heteroscedastic diagonal breaks the Kronecker structure, the cost per step is $\mathcal{O}((N_s P)^3)$; we therefore use $N_s = 1000$ in this experiment (instead of 5000), so that the joint matrix is 3000×3000.

Results. We consider a three-channel *Airfoil* target (x-velocity u, y-velocity v, and pressure p), with a Transolver trained to predict all three channels, over five seeds. Table 14 shows that REEF-GP-MO preserves the base accuracy on every channel and remains well calibrated, with NLL close to the Deep Ensemble, which again benefits from multiple independently trained operators.

Table 14: **Multi-output REEF-GP (ICM) on a three-channel *Airfoil* target.** Lower is better (↓).

Channel	Method	rL2 ↓ (%)	NLL ↓	CRPS ↓ ($\times 10^{-3}$)	NIS ↓ ($\times 10^{-2}$)
u (x-velocity)	Base (no UQ)	1.01 ±0.13	–	–	–
	Ensemble	**0.85**	**−4.05**	**2.81**	**3.26**
	REEF-GP-MO	1.01 ±0.13	−3.57 ±0.22	3.55 ±0.30	5.41 ±0.83
v (y-velocity)	Base (no UQ)	3.37 ±0.30	–	–	–
	Ensemble	**2.58**	**−4.94**	**1.04**	**1.11**
	REEF-GP-MO	3.36 ±0.30	−4.61 ±0.09	1.41 ±0.10	1.69 ±0.17
p (pressure)	Base (no UQ)	0.90 ±0.13	–	–	–
	Ensemble	**0.74**	**−4.47**	**2.00**	**2.20**
	REEF-GP-MO	0.89 ±0.13	−3.74 ±0.29	2.72 ±0.26	4.48 ±0.77

For time-dependent problems, time can be included as an additional coordinate in k_{space}, which is a natural extension that we leave for future work.

M REEF-GP on Top of Deep Ensembles

A Deep Ensemble captures the variation between independently trained operators, which no post-hoc model of a single operator can, while REEF-GP models the structured residual of one operator. To test whether the two combine, we fit one REEF-GP to each of the $E = 5$ Transolvers of the *ShapeNet Car* ensemble, which gives Gaussian predictives $\mathcal{N}(\mu_e, \sigma_e^2)$, and combine $E' \leq E$ of them with gPoE, adding the disagreement between members to the variance:

$$\frac{1}{\sigma^2_{\text{gPoE}}} = \frac{1}{E'}\sum_{e=1}^{E'}\frac{1}{\sigma_e^2}, \quad \mu_{\text{ens}} = \sigma^2_{\text{gPoE}}\,\frac{1}{E'}\sum_{e=1}^{E'}\frac{\mu_e}{\sigma_e^2}, \quad \sigma^2_{\text{ens}} = \sigma^2_{\text{gPoE}} + \frac{1}{E'}\sum_{e=1}^{E'}\big(\mu_e - \bar{\mu}\big)^2, \tag{35}$$

where $\bar{\mu}$ is the mean of the μ_e.

Results. All metrics improve as members are added (Table 15). With five members, REEF-GP-Ensemble matches the Deep Ensemble in rL2 and improves on it in NLL, CRPS, NIS, and ES, at the cost of five post-hoc fits on top of the trained ensemble. The single-member row ($E' = 1$) is the REEF-GP of Table 1.

Table 15: **REEF-GP on top of Deep Ensembles on *ShapeNet Car*.** E' is the number of members combined. For $E' < 5$: mean $\pm$ std over all member subsets; $E' = 5$ and the Deep Ensemble are single values. Lower is better (↓). Best per metric in **bold**.

Method	**rL2** ↓ (%)	**NLL** ↓	**CRPS** ↓	**NIS** ↓	**ES** ↓
Deep Ensemble ($E = 5$)	7.73	4.16	1.88	30.28	194.6
REEF-GP-Ensemble, $E' = 1$	8.52 $\pm$0.14	2.84 $\pm$0.09	2.05 $\pm$0.03	28.81 $\pm$1.65	224.2 $\pm$4.2
REEF-GP-Ensemble, $E' = 2$	8.04 $\pm$0.10	2.57 $\pm$0.02	1.89 $\pm$0.02	23.18 $\pm$0.61	205.1 $\pm$2.6
REEF-GP-Ensemble, $E' = 3$	7.87 $\pm$0.06	2.50 $\pm$0.01	1.84 $\pm$0.01	21.48 $\pm$0.33	199.2 $\pm$1.8
REEF-GP-Ensemble, $E' = 4$	7.78 $\pm$0.04	2.46 $\pm$0.01	1.81 $\pm$0.01	20.66 $\pm$0.18	196.2 $\pm$1.1
REEF-GP-Ensemble, $E' = 5$	**7.72**	**2.44**	**1.79**	**20.19**	**194.5**

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately state the contributions of the paper: we introduce a geometry-aware deep kernel built on the operator's internal representations, develop scalable training and inference procedures via stochastic subset optimization, and achieve competitive results on 2D and 3D benchmarks including geometric distribution shift. These claims are substantiated by the methodology, the empirical results, and the out of distribution (OOD) analysis in the Appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in Section 5. The assumptions, design choices and experiments are discussed in Sections 3, 4 and Appendix B. Ablation studies are provided in Appendix D to assess the robustness and computational tradeoffs of our proposed approach.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper does not present formal theorems or proofs. The theoretical content of the functional regression formulation for operator learning is provided in Appendix A and the GP kernel derived theory relies on standard, properly cited results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The method is fully described in Section 3 and Appendix B (layer selection, deep kernel design, heteroscedastic noise, training procedure, gPoE inference). Hyperparameter configurations for our method are listed in Table 2, base Transolver configurations in Table 8, dataset splits in Table 9, baseline implementation details in Appendix G, and metric definitions in Appendix H. We also report the random seeds ({0,1,2,3,4}) and hardware used, and we provide the code for reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The code is publicly available at `https://github.com/Bostanabad-Research-Group/GP-GeometryAware-OL`. All datasets used are publicly available and properly cited in Appendix F.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Training and test details are provided throughout the paper: data splits in Table 9, training configuration in Appendix E, base operator hyperparameters in Table 8, our method's hyperparameters in Table 2, and baseline configurations in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: All experiments are run with five random seeds ({0,1,2,3,4}). Tables with metrics report mean $\pm$ one standard deviation across seeds, and figures provide standard deviation bands where necessary.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix B.4 specifies the hardware used to train and test both our model and the baselines we use to compare. Compute costs are reported across all benchmarks in Figure 3 (main text) and Figure 18 (Appendix I).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. The work uses only publicly available PDE benchmark datasets and presents methodological contributions to scientific machine learning with no foreseeable ethical concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This is foundational methodological research on uncertainty quantification for PDE surrogate models. The work is not tied to a specific deployed application and has no direct path to negative societal applications. The intended use, providing calibrated uncertainty estimates for scientific computing surrogates, has potential positive impact by improving the reliability of neural-operator-based predictions in scientific and engineering domains.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release pretrained generative models, scraped datasets, or any assets with high risk for misuse. The released code implements a UQ method on top of standard, publicly available PDE benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All existing assets used are properly cited in the references and we acknowledge and cite the code packages from publicly available sources used to implement the baselines we compare against.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces the REEF-GP code as a new asset. It is publicly available, together with documentation and instructions to reproduce the main experimental results, at `https://github.com/Bostanabad-Research-Group/GP-GeometryAware-OL` (see Appendix B). No new datasets are introduced.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: The paper does not involve human subjects research, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs were not used as an important, original, or non-standard component of the core methods. Any use of LLMs was limited to writing, editing, or formatting assistance and did not impact the methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.