Light-Weight Diffusion Multiplier and Uncertainty Quantification for Fourier Neural Operators

Albert Matveev **Sanmitra Ghosh** **Aamal Hussain**
James-Michael Leahy **Michalis Michaelides**
PhysicsX
London, UK
albert.matveev@physicsx.ai

Abstract

Operator learning is a powerful paradigm for solving partial differential equations, with Fourier Neural Operators serving as a widely adopted foundation. However, FNOs face significant scalability challenges due to overparameterization and offer no native uncertainty quantification – a key requirement for reliable scientific and engineering applications. Instead, neural operators rely on post hoc UQ methods that ignore geometric inductive biases. In this work, we introduce DINOZAUR: a **di**ffusion-based **n**eural **o**perator parametri**za**tion with **u**nce**r**tainty quantification. Inspired by the structure of the heat kernel, DINOZAUR replaces the dense tensor multiplier in FNOs with a dimensionality-independent diffusion multiplier that has a single learnable time parameter per channel, drastically reducing parameter count and memory footprint without compromising predictive performance. By defining priors over those time parameters, we cast DINOZAUR as a Bayesian neural operator to yield spatially correlated outputs and calibrated uncertainty estimates. Our method achieves competitive or superior performance across several PDE benchmarks while providing efficient uncertainty quantification. The code is available at `https://github.com/PhysicsXLtd/DINOZAUR`.

1 Introduction

Partial differential equations (PDEs) are a fundamental mathematical tool that describe a wide range of physical phenomena observed in the real world. Numerous problems in science and engineering require formulating and solving PDEs, often presenting significant computational and methodological challenges to researchers and practitioners. A common strategy for solving PDEs is via conventional numerical methods, such as finite difference, finite element, or finite volume method [1]. While accurate, they become prohibitively expensive in complex engineering tasks, especially where rapid iteration or real-time response is critical, such as design optimization or control [2].

In recent years, there has been a growing trend toward replacing computationally intensive numerical solvers with neural network-based surrogate models trained to approximate PDE solutions. Since discretized PDE domains are commonly represented using point clouds, meshes, or graphs, *geometric deep learning* [3] has emerged as a promising paradigm for this class of problems. In particular, message passing layers, initially developed for geometric and relational data, have proven effective due to their structural resemblance to physical convolution operators. These layers are especially suitable for capturing the spatial dependencies inherent in PDE systems, making them well-suited for learning dynamics over irregular or non-Euclidean domains [4, 5].

However, geometric deep learning methods often lack key properties required for solving PDEs, motivating alternative approaches. A PDE – its parameters, equation structure, and boundary conditions – defines an operator that maps input functions to solution functions. *Neural operators*

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

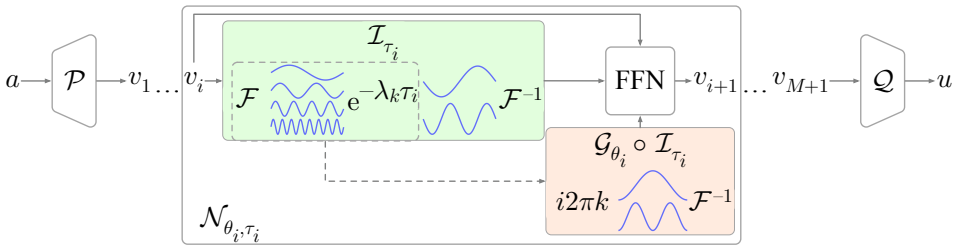

Figure 1: Overview of DINOZAUR. NO block $\mathcal{N}_{\theta_i,\tau_i}$ is revised by updating the integral transform $\mathcal{I}_{\tau_i}$ with diffusion multiplier $\exp(-\lambda_k \tau_i)$ and including gradient features $\mathcal{G}_{\theta_i}$ to add anisotropy. Feed-forward network (FFN) is applied to mix two sets of features.

(NOs) [6–8] have recently emerged as a leading framework for approximating such solution operators and, more generally, for tackling the *operator learning* problem implicitly present in PDEs. These models incorporate inductive biases that improve performance and offer discretization invariance and universal approximation [7], making them a strong alternative to geometric deep learning. A prominent example is the Fourier Neural Operator (FNO) [9], which learns transformations in the spectral domain via the Fast Fourier Transform. While FNOs are restrictive in requiring input data on a regular grid, they remain a widely used backbone in neural operator architectures [10]. Despite their success on problems such as Darcy flow and Burgers' equation, FNOs are known to be overparameterized [11, 12]. Their design leads to exponential growth with domain dimensionality, causing overfitting and significant memory overhead.

Moreover, PDE solvers often inform downstream decision-making, so it is imperative for any substitute method to quantify the uncertainty on its output. *Uncertainty quantification* (UQ) is a common strategy to improve model robustness by estimating confidence in its predictions. It enhances model outputs by accounting for different sources of error, helping guide more informed and reliable downstream use. Uncertainty modeling has long relied on Bayesian inference [13], ensembles [14], and conformal prediction [15] to produce confidence estimates. While some of these methods have been adapted to deep learning with varying success [16–18], UQ for GNNs and neural operators [19] remains underdeveloped, often relying on post hoc methods ill-suited to their structure.

To address the scalability issues and limited UQ methods in FNOs, we present DINOZAUR – a **di**ffusion-based **n**eural **o**perator parametri**za**tion with **u**ncer**t**ainty quantification. Our *diffusion multiplier* is inspired by the heat kernel in the Fourier basis and requires only one learnable parameter per channel, resulting in superior scalability without sacrificing predictive performance. This physically motivated choice of multiplier allows us to place meaningful probabilistic priors on the learnable time parameters, recover the posterior via inexpensive variational inference (VI) [20] of these parameters, and efficiently sample spatially correlated functions. We summarize our contributions as follows:

- We introduce **dimensionality-independent** spectral multiplier parametrization, reducing the parameter count by **several orders of magnitude** compared to equivalent FNO-based architectures.
- We define a Bayesian formulation of the heat kernel and build an **interpretable** uncertainty quantification method to sample **spatially correlated** outputs.
- We demonstrate that our method is scalable, maintains strong predictive performance across multiple PDE benchmarks, and achieves superior UQ estimates compared to conventional probabilistic methods.

2 Related work

The area of designing learnable approximation for PDE solutions is vast and rapidly growing [21, 22]. Given the increasing availability of data, the need to incorporate inductive biases, and the inherently geometric structure of many physical problems, two classes of deep learning models have come to dominate the field: geometric deep learning models and neural operators.

Geometric deep learning Graph neural networks (GNNs) [23] offer a flexible framework for learning on non-Euclidean domains. Many GNNs implement spatial convolution by performing learnable aggregation of local neighborhoods [24, 25]. Neural PDE solvers increasingly rely on spatial GNN layers to model and predict physical phenomena across various domains, including control [26], deformable material simulation [27], and spatiotemporal mesh evolution [4]. Specific architectures target building robust autoregressive solvers [5] and encoding the geometric boundary conditions [28]. However, their performance often suffers from sensitivity to discretization, limiting generalization across mesh resolutions. Spectral GNNs define convolutions using the graph Laplacian's eigendecomposition [29, 30], often with polynomial approximations of spectral filters [31, 32]. These have seen success in 3D shape analysis, particularly with functional maps [33]. DiffusionNet [34] exemplifies this approach by using the spectral form of the heat kernel for a simple and interpretable kernel parametrization. It exhibits strong performance and generalization; we note that such a kernel is applicable in arbitrary domains.

Neural operators Neural operators are a powerful class of models for learning mappings between infinite-dimensional spaces in a resolution-invariant manner [7], offering an alternative to GNNs. Lu et al. [6] introduced DeepONet, inspired by operator learning and universal approximation theory, using a branch-trunk architecture to encode inputs and evaluate the solution at arbitrary locations. A parallel line of work introduced the Graph Neural Operator (GNO) architecture [35, 36], transferring the message passing intuition from GNNs to NOs. A family of neural operators decomposes the input function into a basis defined by one of the popular transforms: Fourier [9], Laplace [37], Wavelet [38]. In particular, FNOs are popular for their performance and interpretability despite needing a regular grid sampling, and are commonly used in other networks, such as Geo-FNO [39], U-FNO [40], and Geometry-Informed Neural Operator (GINO) [10], to name a few. However, FNOs face scalability and generalization issues due to exponential parameter growth with domain dimensionality [11]. Efforts to improve this, such as FFT factorization [11], low-rank tensor decomposition [12], and MLPs in the spectral domain [41], have reduced growth but still rely on spatial dimensions.

Functional uncertainty quantification UQ in deep learning has been widely studied, with popular methods including MC Dropout [16], Laplace approximation [42], ensembles [43], and non-parametric conformal prediction [17], which typically require extra computations or careful calibration. In the context of *functional* UQ – crucial for neural operators – the landscape is further limited. Recent approaches include applying a post hoc Laplace approximation to the final layer of the NO [44], using conformal methods after training [45], or learning an uncertainty measure as an auxiliary output [46]. Another family of approaches leverages diffusion-based generative models for uncertainty quantification in physics surrogates [47–50]. While these methods use diffusion models to capture statistics of high-dimensional dynamical systems and generate realistic samples, they require solving a differential equation for reverse diffusion numerically through multiple denoising steps, which is computationally intensive compared to deterministic forward passes in neural operators. Despite these various approaches, none of them model uncertainty within the NO itself; instead, they adopt generic methods for neural networks and neglect the inductive biases of neural operators.

3 Diffusion multiplier parametrization for neural operators

3.1 Problem setting

We consider the task of learning a parametric surrogate to map the initial and boundary conditions of a PDE to its solution. Let $\Omega \subset \mathbb{R}^d$ be a bounded open set representing a PDE domain in d-dimensional Euclidean space. Let also $\mathcal{A} = \mathcal{A}(\Omega; \mathbb{R}^{d_a})$ be a Banach space of real-valued functions defined on Ω and $\mathcal{U} = \mathcal{U}(\Omega; \mathbb{R}^{d_u})$ be a space of solution functions. We associate $a \in \mathcal{A}$ with a function that parametrizes the partial differential operator $\mathcal{L}_a$ and trace operator $\mathcal{B}_a$. Then, we form the PDE:

$$\begin{aligned} \mathcal{L}_a[u](x) &= f(x) \quad \forall x \in \Omega, \\ \mathcal{B}_a[u](x) &= g(x) \quad \forall x \in \partial\Omega\,, \end{aligned} \tag{1}$$

where $f(x)$ is a fixed forcing function and $g(x)$ is a boundary condition. This formulation is general enough to include varying geometric boundaries by including the signed distance function of a geometry in functions from $\mathcal{A}$ and defining a solution as an extension to the whole domain [51, 10].

We assume the PDE induces a continuous solution map $\mathcal{N} : \mathcal{A} \to \mathcal{U}$ that maps inputs a to solutions u, and we are seeking to find its parametrized approximation $\mathcal{N}_\theta$ with p parameters. Given the discretized observations, $\{a_n, u_n\}_{n=1}^N$, we pose a problem of the empirical risk minimization :

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{N} \sum_{n=1}^{N} \|u_n - \mathcal{N}_\theta[a_n]\|_{\mathcal{U}}^2. \tag{2}$$

We construct the neural operator in the usual form as a sequence of lifting layer $\mathcal{P}$, M NO blocks $\mathcal{N}_{\theta_i}$, and projection layer $\mathcal{Q}$, where the NO block $\mathcal{N}_{\theta_i}$ is given by:

$$v_{i+1}(x) = \mathcal{N}_{\theta_i}[v_i](x) = \sigma\left(W_i^{\text{skip}} v_i(x) + b_i + \int_\Omega \kappa_{\gamma_i}(x, y) v_i(y) \mathrm{d}y\right), \; i = 1, \ldots, M, \tag{3}$$

where κ_{γ_i} is a kernel, matrix $W^{\text{skip}} \in \mathbb{R}^{d_c \times d_c}$, $b_i \in \mathbb{R}^{d_c}$ is a bias and d_c is the number of channels. FNO restricts domain to be periodic torus $\Omega = \mathbb{T}^d$, kernels to be stationary and represents the integral transform $\mathcal{I}$ in the following way:

$$\mathcal{I}_\gamma[v](x) = \int_\Omega \kappa_\gamma(x - y) v(y) \mathrm{d}y = \mathcal{F}^{-1}\big[R_\gamma(k) \mathcal{F}[v](k)\big](x), \tag{4}$$

where $\mathcal{F}$ is Fourier transform, and $R_\gamma(k)$ acts as a unique complex-valued matrix if $k < k_{\max} \in \mathbb{Z}^d$ and as zero otherwise.

Given the block's width d_c and maximal truncation modes $k_{\max} = (k_{\max}^1, \ldots, k_{\max}^d) \in \mathbb{Z}^d$, tensor $R_\gamma = [R_\gamma(k)]_{k \le k_{\max}}$ contains $O(d_c^2 \cdot \Pi_{j=1}^d k_{\max}^j)$ learnable parameters, yielding quadratic dependency on the number of channels and exponential growth of parameters in the dimensionality d of the domain Ω. In this paper, we target the form of kernel κ_γ and propose an alternative parametrization.

3.2 DINOZAUR architecture

Diffusion multiplier Motivated by the success of DiffusionNet [34] and its resemblance to FNO, we choose to restrict the form of κ_γ to that of the heat kernel for our architecture. Consider the heat equation on the torus $\Omega = \mathbb{T}^d$:

$$\begin{aligned} \partial_\tau v(\tau, x) &= \Delta v(\tau, x) \\ v(0, x) &= v_0(x), \end{aligned} \tag{5}$$

with a solution [52] given by:

$$v(\tau, x) = \mathcal{I}_\tau[v_0](x) = \mathcal{F}^{-1}\big[R_\tau(k) \mathcal{F}[v_0](k)\big](x) = \mathcal{F}^{-1}\big[\exp(-\lambda_k \tau) \odot \mathcal{F}[v_0](k)\big](x), \tag{6}$$

where $\odot$ is the Hadamard product and $\lambda_k = -4\pi^2 \|k\|^2$. We refer to $R_\tau = [\exp(-\lambda_k \tau)]_{k \le k_{\max}}$ as a *diffusion multiplier*. Figure 1 illustrates the updated block architecture.

When integrated into the FNO block, $R_\tau \in \mathbb{R}^{k_{\max}^1 \times \cdots \times k_{\max}^d \times d_c}$ is a tensor with the only learned parameters being $\tau = (\tau^1, \ldots, \tau^{d_c}) \in \mathbb{R}^{d_c}$. Each τ^c is an independent *scalar* defining diffusion time within channel c of the NO block. Times τ control the adjustable receptive field of the convolution kernel and are not related to the time variable that may be present in the original PDE (1). We note that our multiplier parametrization *does not depend on dimensionality* d of the domain Ω or on the truncation modes $k_{\max}$ since λ_k is precomputed and fixed.

Gradient features The diffusion multiplier propagates information across the domain, enabling efficient function convolution. However, the heat kernel is isotropic by construction, which may limit its applicability. We found that including gradient features mitigates that issue and yields better experimental results.

Since we already map to Fourier basis, it's natural to calculate gradients in the spectral domain. Raw gradients can be ill-defined due to non-periodic boundaries or uneven sampling. Following Sharp et al. [34], to improve robustness and add some flexibility, we apply a linear transformation $W^{\text{grad}} \in \mathbb{R}^{d_c \times d_c}$ with a hyperbolic tangent non-linearity to normalize the output in the physical domain. We denote the gradient features operator as $\mathcal{G}_\theta$ and define its output at channel c as follows:

$$\begin{aligned} \nabla v(x) &= \mathcal{F}^{-1}\big[i 2\pi k \mathcal{F}[v](k)\big](x) \\ \mathcal{G}_\theta[v]^c(x) &= \tanh\left(\sum_{r=1}^{d_c} \Big\langle \nabla v^c(x), W_{cr}^{\text{grad}} \nabla v^r(x) \Big\rangle\right), \; c = 1, \ldots, d_c. \end{aligned} \tag{7}$$

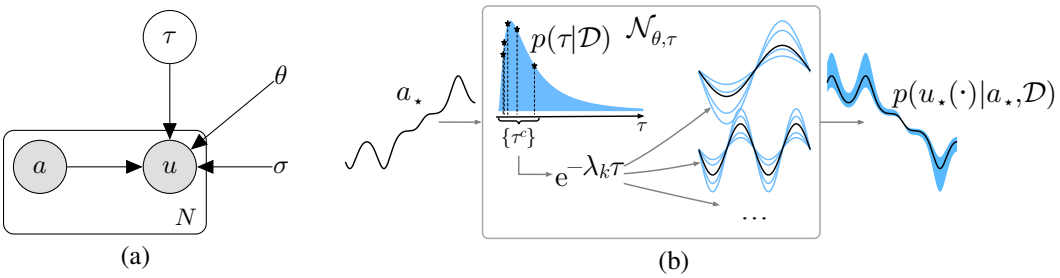

Figure 2: (a) Probabilistic graph for DINOZAUR, showing how the latent diffusion-times, τ, relate to the observations and non-mechanistic model parameters, θ, σ. (b) We define priors for τ and sample multiple time parameters from posterior in each block at inference time, which translates into spatially correlated uncertainty at the output.

Since $v \in \mathbb{R}^{d_c}$, its Jacobi matrix is $\nabla v \in \mathbb{R}^{d_c \times d}$. By taking the inner product, we reduce the spatial dimensionality d and end up with a compatible shape. We report the results with and without the gradient features in Section 4.1.

Gradient features are applied to the output of the integral transform $\mathcal{I}_\tau$ and stacked with it inside the DINOZAUR block, doubling the number of channels. We add a matrix $W^{\text{mix}} \in \mathbb{R}^{d_c \times 2d_c}$ to map all features back to the width of the block. The final block architecture is:

$$v_{i+1}(x) = \sigma\left(W_i^{\text{skip}} v_i(x) + b_i + W_i^{\text{mix}} \begin{bmatrix} \mathcal{I}_{\tau_i}[v_i](x) \\ (\mathcal{G}_{\theta_i} \circ \mathcal{I}_{\tau_i})[v_i](x) \end{bmatrix}\right). \tag{8}$$

Substituting the original blocks in (3) in the neural operator $\mathcal{N}_\theta$ with modified blocks from (8) results in DINOZAUR architecture that we denote as $\mathcal{N}_{\theta,\tau}$, with all non-diffusion parameters lumped together in θ and separated from τ.

Universal approximation Despite the seeming simplicity of the diffusion multiplier, we claim that our formulation of a neural operator exhibits universal approximation properties: see Proposition 1.

Proposition 1. *Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with Lipschitz boundary and such that the closure $\overline{\Omega} \subset (0, 2\pi)^d$. For given integers $s, s' > 0$, let $\mathcal{N} : C^s(\overline{\Omega}; \mathbb{R}^{d_a}) \to C^{s'}(\overline{\Omega}; \mathbb{R}^{d_u})$ be a continuous operator, and fix a compact set $\mathbf{A} \subset C^s(\overline{\Omega}; \mathbb{R}^{d_a})$. Then there exists a continuous, linear operator $\mathcal{E} : C^s(\Omega; \mathbb{R}^{d_a}) \to C^s(\mathbb{T}^d; \mathbb{R}^{d_a})$ such that $\mathcal{E}[a]|_\Omega = a$ for all $a \in C^s(\Omega; \mathbb{R}^{d_a})$. Furthermore, for any $\varepsilon > 0$, there exists a* DINOZAUR such that

$$\sup_{a \in \mathbf{A}} \|\mathcal{N}[a] - (\mathcal{N}_{\theta,\tau} \circ \mathcal{E}[a])|_\Omega\|_{C^{s'}(\overline{\Omega}; \mathbb{R}^{d_u})} \leq \varepsilon,$$

where $\mathcal{N}_{\theta,\tau} = \mathcal{Q} \circ \mathcal{N}_{\theta_1,\tau} \circ \mathcal{P} : C^s(\mathbb{T}^d; \mathbb{R}^{d_a}) \to C^{s'}(\mathbb{T}^d, \mathbb{R}^{d_u})$.

We provide the proof in the Appendix E.

3.3 Bayesian inference for diffusion multiplier

Now that we established the overall architecture, we will describe the Bayesian inference scheme to obtain UQ. An advantage of the proposed diffusion multiplier as a message passing scheme is the ability to impose meaningful priors on the diffusion-time parameters τ. Doing so recasts DINOZAUR as a *Bayesian* neural operator, with the source of uncertainty coming from the *integral transform*. In our method, other model parameters (θ, σ) are treated deterministically. The conciseness of our multiplier's parametrization allows for tractable inference and alleviates the need to grapple with extremely high-dimensional distributions. We define our conditional generative model for any output signal u_n given an input signal a_n as

$$\begin{aligned} \ln \tau_i^c &\sim \mathbf{N}(\mu_{\text{prior}}, \sigma_{\text{prior}}^2),\ i = 1, \ldots, M;\ c = 1, \ldots, d_c \\ u_n(\cdot)|a_n(\cdot), \tau &\sim \mathbf{N}(\mathcal{N}_{\theta,\tau}[a_n](\cdot), \sigma^2), \end{aligned} \tag{9}$$

where $f(\cdot)$ denotes functions evaluated at any finite number of points in the domain Ω, $\mathbf{N}$ denotes a normal distribution, and $\mu_{\text{prior}}, \sigma_{\text{prior}}$ are hyperparameters of the prior distribution. We collect all

time variables from each channel c and block i for notational convenience, $\tau = \left\{(\tau_i^c)_{c=1}^{d_c}\right\}_{i=1}^{M}$. See Figure 2(a) for the corresponding graphical model. We also collect training data in a set of discrete observations, $\mathcal{D} = \left\{\{a_{nk}\}_{k=1}^{K_n}, \{u_{nj}\}_{j=1}^{J_n}\right\}_{n=1}^{N}$, with possibly different sets of function evaluation points at the input and output.

We seek to recover the approximate posterior for the diffusion time, $p_{\theta,\sigma}(\tau|\mathcal{D})$, according to (9), which yields a posterior for the diffusion multipliers, R_τ, and the entire neural operator $\mathcal{N}_{\theta,\tau}$. This can be used to produce a posterior predictive distribution over the output for a new input function $a_\star$ (see Figure 2(b)):

$$p_{\theta,\sigma}(u_\star(\cdot)|a_\star, \mathcal{D}) = \int p_{\theta,\sigma}(\tau|\mathcal{D})\mathbf{N}(u_\star(\cdot); \mathcal{N}_{\theta,\tau}[a_\star](\cdot), \sigma^2)\mathrm{d}\tau, \tag{10}$$

where we marginalize over τ to reflect epistemic uncertainty over the possible operators that fit the data, as well as aleatoric uncertainty from the assumed normal noise with fitted σ. Note that while the noise is point-wise independent for a fixed τ, the τ dependence *spatially correlates* predictions.

Variational inference As mentioned, the posterior $p_{\theta,\sigma}(\tau|\mathcal{D})$ is intractable, so we resort to VI to recover an approximate posterior. We assume that it can be adequately approximated by a distribution $q(\tau;\phi)$ parametrized by ϕ. We choose $q(\tau;\phi)$ such that it factorizes over the $i = 1, \ldots, M$ blocks of the NO but keeps correlations between diffusion times of channels in a block:

$$q(\tau;\phi) \doteq \prod_{i=1}^{M} q(\tau_i;\phi_i) \doteq \prod_{i=1}^{M} \mathbf{N}(\ln \tau_i; \mu_i, \Sigma_i), \tag{11}$$

where $\tau_i, \mu_i \in \mathbb{R}^{d_c}$, and $\Sigma_i \in \mathbb{R}^{d_c \times d_c}$. We collect the parameters of the approximating factors, $\phi = \{\phi_i\}_{i=1}^{M} = \{(\mu_i, \Sigma_i)\}_{i=1}^{M}$, for notational convenience. Due to their parsimonious and mechanistic role in controlling message propagation, diffusion time parameters avoid the redundancy and symmetries that challenge VI for typical neural network weights, allowing for better identifiability and justifying a full-rank approximation within each block.

We then seek to find ϕ that minimizes the Kullback-Leibler divergence (KL) from $q(\tau;\phi)$ to the true posterior. As this objective is still intractable [53], we instead maximize a lower bound (ELBO):

$$\mathcal{L}_{\mathrm{ELBO}}(\theta, \sigma; \phi) = \mathbb{E}_{q(\tau;\phi)}[\ln p_{\theta,\sigma}(\mathcal{D}|\tau)] - D_{\mathrm{KL}}[q(\tau;\phi)\|p(\tau)], \tag{12}$$

which allows to jointly learn the network parameters (θ, σ) and variational parameters ϕ. To predict for a new instance, we replace the true posterior $p_{\theta,\sigma}$ with its approximation $q(\tau;\phi)$ in (10).

4 Experiments

We evaluate the performance of the DINOZAUR architecture on several standard benchmarks. Section 4.1 presents results for the deterministic variant of DINOZAUR. In Section 4.2, we enable the Bayesian version and assess its UQ capabilities, comparing it to established UQ methods for neural operators. Ablation studies and computational performance comparisons are detailed in Section 4.3. All models are implemented in PyTorch [54] and trained on a single NVIDIA A100 GPU for 150 epochs with AdamW [55] and One-Cycle scheduler [56] or AdamW Schedule-Free [57] with learning rate (in most cases) set to 10^{-3}. See the Appendix B for details. All experiments were repeated three times with random initialization of trainable parameters. Reported results include the mean and standard deviation over all runs.

Benchmarks We evaluate our method on two regularly sampled datasets from [9] and two datasets with irregular domains used in [10]. For the 2D Darcy flow, we learn a mapping from a scalar permeability field of resolution 241^2 to a scalar pressure field, applying standard scaling to both. The 2D Navier-Stokes (NS) dataset contains 64^2 spatial points and 20 time steps of simulations with viscosity $\nu = 10^{-5}$ on a uniform grid. We learn to map the first 10 time steps of vorticity to the last 10, without applying data scaling. For further dataset and PDE details, we refer to [58]. The irregular-domain benchmarks include two datasets for predicting surface pressure on 3D meshes: ShapeNet car [59] and Ahmed bodies [60]. Following the protocol of Li et al. [10], we evaluate the

signed distance function (SDF) of each mesh on a 64^3 uniform grid and apply standard scaling to the outputs during training. We learn a map from SDF values to scalar pressure on mesh vertices, with inlet velocity added to input in the Ahmed dataset. A summary of the benchmark datasets is provided in the Appendix A.

Baselines In all baselines, we used a "double skip" implementation of the FNO block introduced in [12]. On uniform benchmarks, we compare DINOZAUR to the FNO architecture from Li et al. [9] with 4 blocks. We set the width $d_c = 32$ and modes $k_{\max} = [12, 12]$ for Darcy and $d_c = 64$, $k_{\max} = [16, 16, 4]$ for NS. For non-uniform domains, we follow the GINO-decoder setup from [10] with 4 FNO blocks, 1 GNO decoder block, and parameters $d_c = 64$, $k_{\max} = [24, 24, 24]$. The original GINO applied low-rank Tucker factorization to the multipliers R_γ with a rank $r = 0.4$. Given that FNO/GINO, even with moderate factorization, produce significantly more parameters than DINOZAUR, we also include Tensorized FNO/GINO (TFNO/TGINO) [12] in our baselines. We choose a factorization rank $r = 10^{-4}$ for TFNO/TGINO as it yields a comparable parameter count to that of DINOZAUR in all cases. Additionally, for all models, we assess the influence of spatial gradient features by including them in the (FNO/GINO)$_{\text{g}}$ and (TFNO/TGINO)$_{\text{g}}$ implementations, as well as removing them from our model (DINOZAUR$_{\text{no-g}}$). As the main metric, we report the Relative L_2:

$$\text{RL}_2 = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \frac{\|u_n(x) - \widehat{u}_n(x)\|}{\|u_n(x)\|}.$$

We emphasize that the only structural change our model makes is the multiplier in FNO blocks.

4.1 Results: deterministic performance

Table 1 reports evaluation results. DINOZAUR significantly outperforms all baselines on Darcy and remains competitive on other benchmarks. The parameter counts in the table highlight a key trend: as problem dimensionality increases, FNO's parameter count grows by two orders of magnitude, whereas DINOZAUR scales only with network width. Although TFNO shares DINOZAUR's favorable scaling, it suffers notable performance drops on uniform datasets. Interestingly, reducing the parameter count in the non-uniform setting improves GINO's performance, further underscoring the overparameterization issue. In contrast, DINOZAUR maintains strong results across domains, demonstrating the effectiveness of diffusion-based message passing as a parameter-efficient alternative.

Table 1: Comparison of DINOZAUR against the baselines in deterministic setting, reporting the RL_2 values on test split. Mean and standard deviation values are computed from three trained models. The best result is **bold**, second best is underlined

Model	# params ($\times 10^3$)	$\text{RL}_2 \downarrow$ ($\times 10^{-2}$)	# params ($\times 10^3$)	$\text{RL}_2 \downarrow$ ($\times 10^{-2}$)
	Darcy flow		Navier-Stokes	
FNO	361.4	$\underline{0.90 \pm 0.03}$	12,650.8	$\underline{15.90 \pm 0.20}$
FNO$_{\text{g}}$	371.9	0.91 ± 0.07	12,692.2	17.08 ± 0.12
TFNO	17.7	8.04 ± 0.86	69.1	21.48 ± 0.27
TFNO$_{\text{g}}$	28.2	4.16 ± 0.61	110.6	19.47 ± 0.58
DINOZAUR$_{\text{no-g}}$ (Ours)	17.5	1.08 ± 0.06	68.2	20.10 ± 0.31
DINOZAUR (Ours)	28.0	$\mathbf{0.75 \pm 0.10}$	109.7	$\mathbf{15.68 \pm 0.21}$
	ShapeNet car		Ahmed bodies	
GINO	49,608.6	8.03 ± 0.39	49,612.9	8.57 ± 0.13
GINO$_{\text{g}}$	49,650.1	7.76 ± 0.64	49,645.3	8.25 ± 0.16
TGINO	149.8	$\underline{7.27 \pm 0.16}$	154.0	$\underline{8.19 \pm 0.49}$
TGINO$_{\text{g}}$	191.3	8.30 ± 0.38	195.5	8.23 ± 0.64
DINOZAUR$_{\text{no-g}}$ (Ours)	138.9	8.34 ± 0.80	143.2	9.59 ± 0.62
DINOZAUR (Ours)	180.4	$\mathbf{7.04 \pm 0.06}$	184.6	$\mathbf{7.99 \pm 0.07}$

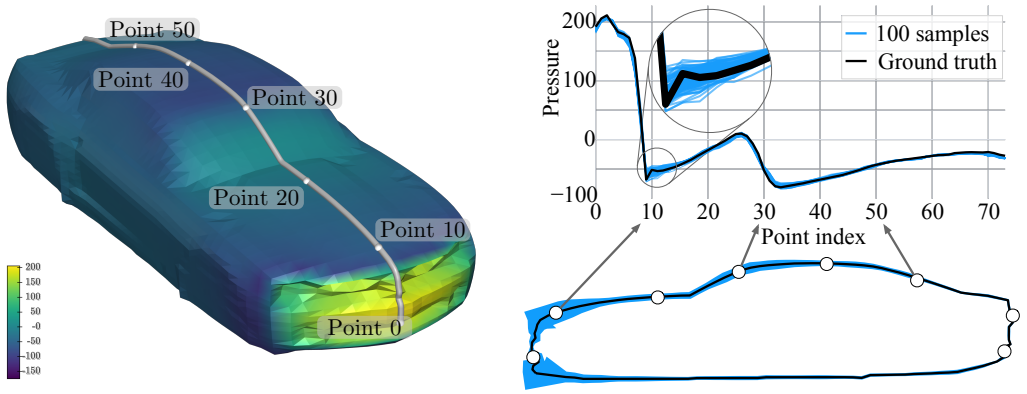

Figure 3: Uncertainty predictions. Left: mean of pressure samples on a test mesh with a geodesic path traced along the top. Right, top: ground truth and 100 sampled predictions along the same path. Right, bottom: car profile with a qualitative visualization of the standard deviation of pressure samples along the outline. High uncertainty is near sharp field changes, e.g., between Points 0 and 10.

We find that gradient features do not consistently improve the performance of baseline models. While general tensors in FNO are capable of implicitly learning gradient-like behavior, our experiments suggest that the addition of explicit gradient features may introduce redundancy or interfere with learned representations in models that already have sufficient capacity. In contrast, without gradients, DINOZAUR$_{\text{no-g}}$ performs poorly on each benchmark. As noted earlier, this is due to the radial symmetry of the diffusion operation, which limits the NO's training capabilities. These observations are consistent with the findings of Sharp et al. [34]. By incorporating gradient features into the diffusion block, DINOZAUR matches the performance of dense FNO/GINO models.

4.2 Results: uncertainty quantification

In this section, we assess the UQ performance of the Bayesian version of our model, DINOZAUR$_{\text{B}}$, introduced in Section 3.3, on the more challenging non-uniform data. As baselines, we consider MC Dropout [16] (NO$_{\text{D}}$) and Laplace approximation [44] (NO$_{\text{L}}$). MC Dropout is a widely used, model-agnostic UQ method, which we apply by enabling 10% dropout in the Fourier blocks' MLPs to emulate stochasticity in the message passing. The Laplace approximation follows [44], modeling uncertainty via a fitted distribution over the parameters of the final MLP layer $\mathcal{Q}$. The discussion of hyperparameter choice, including $\mu_{\text{prior}}, \sigma_{\text{prior}}$ required for setting $\ln \tau$ distributions is in the Appendix B.

UQ metrics in Table 2 are calculated by drawing 100 posterior predictive distribution samples per test data instance. The reported mean and standard deviation are obtained from three independently

Table 2: Evaluation of the uncertainty output quality of DINOZAUR and baseline models. Mean and standard deviation values are computed from three trained models. The best result is **bold**, second best is underlined

Model	$\text{RL}_2 \downarrow$ $(\times 10^{-2})$	NLL↓	MA↓	IS↓
		ShapeNet car		
GINO$_{\text{D}}$	7.94 ± 0.54	3.090 ± 0.044	0.288 ± 0.010	14.731 ± 0.625
GINO$_{\text{L}}$	7.90 ± 0.29	3.106 ± 0.023	0.283 ± 0.007	14.709 ± 0.343
TGINO$_{\text{D}}$	7.18 ± 0.17	2.996 ± 0.010	0.278 ± 0.005	13.704 ± 0.189
TGINO$_{\text{L}}$	7.51 ± 0.73	3.071 ± 0.059	$\underline{0.266 \pm 0.018}$	14.240 ± 0.811
DINOZAUR$_{\text{D}}$ (Ours)	$\mathbf{6.96 \pm 0.04}$	$\underline{2.987 \pm 0.003}$	0.272 ± 0.002	$\underline{13.479 \pm 0.044}$
DINOZAUR$_{\text{L}}$ (Ours)	$\underline{7.11 \pm 0.05}$	3.041 ± 0.008	$\underline{0.266 \pm 0.001}$	13.749 ± 0.003
DINOZAUR$_{\text{B}}$ (Ours)	7.49 ± 0.07	$\mathbf{2.767 \pm 0.011}$	$\mathbf{0.168 \pm 0.001}$	$\mathbf{11.667 \pm 0.136}$

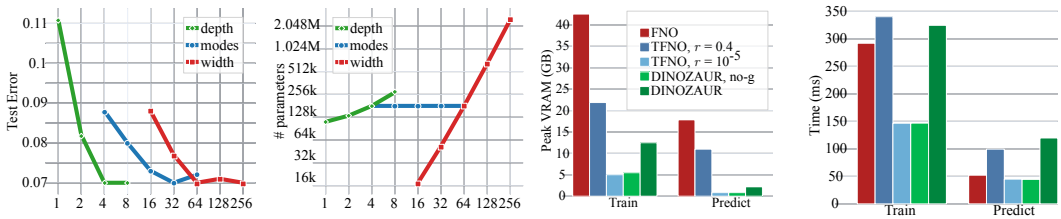

Figure 4: Scalability and efficiency of DINOZAUR. (a) Test RL_2 error on ShapeNet car as width, depth, and modes are varied. (b) Model size under the same settings. (c) Average peak GPU memory usage across five train-predict cycles on synthetic data. (d) Average time under the same settings.

trained models with random weight initialization. Metrics are computed using the Uncertainty Toolbox [61]; full metric definitions are provided in the Appendix D. Aleatoric noise was added at prediction time to all model outputs, using a fixed $\sigma^2 = e^{-4}$ for dropout and Laplace models, and a learned variance term for Bayesian DINOZAUR that converged to $\sigma^2 \approx e^{-4.8}$. $\text{DINOZAUR}_{\text{B}}$ matches or surpasses all baselines, achieving lower negative log-likelihood (NLL) than dropout models or Laplace approximation. It also outperforms all baselines in terms of miscalibration area (MA) and interval score (IS). An important consideration is that the Laplace approximation requires a secondary computationally expensive step to obtain the Hessian of the loss function with respect to the network parameters after training. In contrast, $\text{DINOZAUR}_{\text{B}}$ natively supports UQ, thereby providing accurate uncertainty estimates out of the box. See results in Figure 3.

4.3 Ablation studies

To evaluate the scalability of DINOZAUR, we conduct experiments varying model width, depth, and the number of truncation modes using the ShapeNet car dataset. As shown in Figure 4(a), test error (measured by RL_2 loss) decreases with increasing architectural hyperparameters, saturating near the values used in our main experiments. Further improvements may be limited by the input data resolution of 64^3. Figure 4(b) confirms our scaling claims: depth increases parameter count almost linearly, width affects it quadratically, while modes do not affect the model size.

We further assess runtime and memory efficiency using synthetic uniform data (64^3 resolution, 5 input channels, 7 output channels), fixing architecture settings to 128 channels, [32, 32, 32] modes, and 4 FNO blocks. We compare our model with and without gradients, FNO, TFNO ($r = 0.4$), and TFNO ($r = 10^{-5}$, matching $\text{DINOZAUR}_{\text{no-g}}$'s parameter count). We record VRAM and training time across five runs. Memory usage is highest for FNO and drops with rank r in TFNO. $\text{DINOZAUR}_{\text{no-g}}$ and TFNO ($r = 10^{-5}$) yield nearly identical memory profiles, while our full model uses roughly twice the memory due to gradients. Runtime varies: TFNO ($r = 0.4$) is slower than standard FNO due to additional tensor multiplications in Tucker decomposition. Our model with gradients is similarly slow, limited by FFT complexity, as we concatenate gradient features before inverse FFT. Corresponding charts are in Figures 4(c-d).

5 Discussion

The recent emergence of neural operators has introduced a powerful alternative to traditional, computationally expensive numerical methods for solving PDEs in downstream applications. Architectures such as the Fourier Neural Operator and the Geometry-Informed Neural Operator have demonstrated notable success in learning solution operators across various physical systems. However, these models rely on highly parameterized transformations in spectral space, which can hinder scalability and interpretability. Furthermore, FNOs and other common architectures fail to provide native support for uncertainty quantification, a critical requirement for reliable deployment in scientific and engineering applications. As a result, practitioners are often forced to apply generic UQ techniques post hoc, which fail to exploit the spatio-temporal inductive biases intrinsic to the underlying physical systems.

In this work, we introduced DINOZAUR, an FNO-based model featuring a parsimonious and physically motivated parameterization of the message passing mechanism. This design stems

from the analytic solution to the heat equation and is augmented by spatial gradient features. By presenting learned diffusion time parameters, DINOZAUR defines a new diffusion multiplier that is dimensionality-independent and offers more favorable scalability than the original FNO. Through extensive evaluation on benchmarks, we showed that DINOZAUR consistently matches or exceeds the predictive performance of dense FNO and GINO architectures while using orders of magnitude fewer parameters. Leveraging this efficient parameterization, we introduce meaningful priors over diffusion time parameters, obtaining a Bayesian neural operator that delivers competitive UQ performance against classical deep learning UQ methods such as MC Dropout and Laplace approximation. To the best of our knowledge, DINOZAUR is the first Bayesian NO to explicitly define distributions in the integral transform.

Limitations and future work In our investigations, we found that the primary limitation of the diffusion mechanism is its radial symmetry. We observed that diffusion alone struggles to train well. Our solution is to include gradient features, though at the cost of increased computational time. Investigating other strategies, like introducing anisotropy intrinsically, using the modifications proposed in [62], or mixing gradients with other features before applying inverse FFT overhead, are promising directions for future work. We found that our Bayesian formulation is sensitive to hyperparameter settings of the prior distributions. Customizing priors across network layers to capture different physical properties could improve the robustness and interpretability of the model. More broadly, exploring other parsimonious kernel parameterizations may offer both interpretability and improved generalization, especially in settings with limited data or strong physical priors. Finally, since our parameterization is independent of spatial dimensionality, it opens up opportunities for pretraining on lower-dimensional or simpler physical systems, with the potential to accelerate learning in more complex, higher-dimensional domains through transfer learning [63].

We believe our work is a step forward to a deeper understanding of neural operators, advancing them toward becoming reliable tools for real-world scientific and engineering applications.

Acknowledgments and Disclosure of Funding

We want to thank Greg Bellchambers, Bachir Djermani, Axen Georget, Pavel Shmakov, and Phoenix Tse for insightful discussions and collaboration on the implementation.

CRediT author statement: **Albert Matveev**: Conceptualization, Methodology, Software, Investigation, Writing - Original Draft, Visualization. **Sanmitra Ghosh**: Conceptualization, Methodology, Software, Investigation, Writing - Review and Editing, Visualization. **Aamal Hussain**: Software, Investigation, Writing - Original Draft. **James-Michael Leahy**: Conceptualization, Methodology, Software, Writing - Review and Editing. **Michalis Michaelides**: Conceptualization, Resources, Writing - Original Draft, Supervision, Project administration.

References

[1] Stig Larsson and Vidar Thomée. *Partial differential equations with numerical methods*, volume 45. Springer, 2003.

[2] Joaquim RRA Martins and Andrew Ning. *Engineering design optimization*. Cambridge University Press, 2021.

[3] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[4] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International conference on learning representations*, 2020.

[5] Johannes Brandstetter, Daniel E Worrall, and Max Welling. Message passing neural pde solvers. In *International Conference on Learning Representations*, 2022.

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

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

[8] Nikola B Kovachki, Samuel Lanthaler, and Andrew M Stuart. Operator learning: Algorithms and analysis. *Handbook of Numerical Analysis*, 25:419–467, 2024.

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

[10] Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[12] Jean Kossaifi, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-grid tensorized fourier neural operator for high- resolution PDEs. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

[13] David John Cameron Mackay. *Bayesian methods for adaptive models*. California Institute of Technology, 1992.

[14] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14:241–258, 2020.

[15] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[17] Jin Zhang, Ulf Norinder, and Fredrik Svensson. Deep learning-based conformal prediction of toxicity. *Journal of chemical information and modeling*, 61(6):2648–2657, 2021.

[18] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[19] 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.

[20] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[21] Haixin Wang, Yadi Cao, Zijie Huang, Yuxuan Liu, Peiyan Hu, Xiao Luo, Zezheng Song, Wanjia Zhao, Jilin Liu, Jinan Sun, et al. Recent advances on machine learning for computational fluid dynamics: A survey. *arXiv preprint arXiv:2408.12171*, 2024.

[22] Shudong Huang, Wentao Feng, Chenwei Tang, Zhenan He, Caiyang Yu, and Jiancheng Lv. Partial differential equations meet deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[23] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[24] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023. PMLR, 2016.

[25] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International conference on machine learning*, pages 1407–1418. PMLR, 2021.

[26] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019.

[27] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[28] Andreas Mayr, Sebastian Lehner, Arno Mayrhofer, Christoph Kloss, Sepp Hochreiter, and Johannes Brandstetter. Boundary graph neural networks for 3d simulations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37(8), pages 9099–9107, 2023.

[29] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[30] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[31] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[32] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1): 97–109, 2018.

[33] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (ToG)*, 31(4): 1–11, 2012.

[34] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022.

[35] Anima Anandkumar, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Nikola Kovachki, Zongyi Li, Burigede Liu, and Andrew Stuart. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 workshop on integration of deep neural models and differential equations*, 2020.

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

[37] Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6(6):631–640, 2024.

[38] Tapas Tripura and Souvik Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115783, 2023.

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

[40] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

[41] Zipeng Xiao, Siqi Kou, Hao Zhongkai, Bokai Lin, and Zhijie Deng. Amortized fourier neural operators. *Advances in Neural Information Processing Systems*, 37:115001–115020, 2024.

[42] David JC MacKay. The evidence framework applied to classification networks. *Neural computation*, 4(5): 720–736, 1992.

[43] Rahul Rahaman et al. Uncertainty quantification and deep ensembles. *Advances in neural information processing systems*, 34:20063–20075, 2021.

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

[45] Ziqi Ma, David Pitt, Kamyar Azizzadenesheli, and Anima Anandkumar. Calibrated uncertainty quantification for operator learning via conformal prediction. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

[46] Christopher Bülte, Philipp Scholl, and Gitta Kutyniok. Probabilistic neural operators for functional uncertainty quantification. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

[47] Han Gao, Sebastian Kaltenbach, and Petros Koumoutsakos. Generative learning for forecasting the dynamics of high-dimensional complex systems. *Nature Communications*, 15(1):8904, 2024.

[48] Roberto Molinaro, Samuel Lanthaler, Bogdan Raonić, Tobias Rohner, Victor Armegioiu, Stephan Simonis, Dana Grund, Yannick Ramic, Zhong Yi Wan, Fei Sha, et al. Generative ai for fast and accurate statistical computation of fluids. *arXiv preprint arXiv:2409.18359*, 2024.

[49] Naichen Shi, Hao Yan, Shenghan Guo, and Raed Al Kontar. Diffusion-based surrogate modeling and multi-fidelity calibration. *IEEE Transactions on Automation Science and Engineering*, 2025.

[50] Vivek Oommen, Aniruddha Bora, Zhen Zhang, and George Em Karniadakis. Integrating neural operators with diffusion models improves spectral representation in turbulence modelling. *Proceedings of the Royal Society A*, 481(2309):20240819, 2025.

[51] Elias M Stein. *Singular integrals and differentiability properties of functions*. Princeton university press, 1970.

[52] Walter A Strauss. *Partial differential equations: An introduction*. John Wiley & Sons, 2007.

[53] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[54] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[55] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[56] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[57] Aaron Defazio, Xingyu Yang, Ahmed Khaled, Konstantin Mishchenko, Harsh Mehta, and Ashok Cutkosky. The road less scheduled. *Advances in Neural Information Processing Systems*, 37:9974–10007, 2024.

[58] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[59] Nobuyuki Umetani and Bernd Bickel. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics (TOG)*, 37(4):1–10, 2018.

[60] Syed R Ahmed, G Ramm, and Gunter Faltin. Some salient features of the time-averaged ground vehicle wake. *SAE transactions*, pages 473–503, 1984.

[61] Youngseog Chung, Ian Char, Han Guo, Jeff Schneider, and Willie Neiswanger. Uncertainty toolbox: an open-source library for assessing, visualizing, and improving uncertainty quantification. *arXiv preprint arXiv:2109.10254*, 2021.

[62] Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32576–32594, 2024.

[63] Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. In *International Conference on Machine Learning*, pages 17616–17635. PMLR, 2024.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[66] Samuel Lanthaler, Zongyi Li, and Andrew M Stuart. Nonlocality and nonlinearity implies universality in operator learning. *arXiv preprint arXiv:2304.13221*, 2023.

[67] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22(290):1–76, 2021.

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: Derivation of the method and results in Section 4 support the claims.

   Guidelines:

   - The answer NA 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 NA 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: Paragraph is presented in Section 5.

   Guidelines:

   - The answer NA 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: [Yes]

Justification: The main and only theoretical statement Proposition 1 has a proof presented in the Appendix E.

Guidelines:

- The answer NA 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: Architecture is described in Section 3. Description of the benchmarks, data processing, optimization settings, and model specifications are included in Section 4 and detailed in the Appendix B and A.

   Guidelines:

   - The answer NA 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 paper used publicly available datasets. Full source code is available at `https://github.com/PhysicsXLtd/DINOZAUR`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Tables including this information are provided in the Appendix B and A.

   Guidelines:

   - The answer NA 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: Numerical experiments had multiple runs, with mean and standard deviation reported in Tables 1 and 2.

   Guidelines:

   - The answer NA 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: Provided in Section 4.

   Guidelines:

   - The answer NA 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: None of the concerns apply to our paper.

   Guidelines:

   - The answer NA 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: [Yes]

    Justification: Impact Statement is provided in the Appendix F.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No risks identified.

    Guidelines:

    - The answer NA 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:

    Guidelines:

    - The answer NA 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 source code contains documentation.

    Guidelines:

    - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification:

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Light-Weight Diffusion Multiplier and Uncertainty Quantification for Fourier Neural Operators: Supplementary Material

A Benchmarks

Table 3: Summary of the benchmark datasets

Dataset	d	Input type	Output type	$N_{\text{train}}, N_{\text{test}}$
Darcy flow	2	Regular $[241; 241]$	Regular $[241; 241]$	1024, 1024
NS 2D+T	2+1	Regular $[64; 64; 10]$	Regular $[64; 64; 10]$	1000, 200
ShapeNet car	3	Regular $[64; 64; 64]$	Irregular $[3{,}586]$	561, 100
Ahmed bodies	3	Regular $[64; 64; 64]$	Irregular $\sim [150{,}000]$	500, 51

Table 4: Data transformations during training

Dataset	Inputs	Input scaler	Padded input	Target	Target scaler
Darcy flow	Permeability	Standard	$[271; 271]$	Pressure	Standard
NS 2D+T	Vort.$\vert_{0 \le t < 10}$, $[x, y, t]$	None	$[64; 64; 15]$	Vort.$\vert_{10 \le t < 20}$	None
ShapeNet car	SDF	None	$[72; 72; 72]$	Pressure	Standard
Ahmed bodies	SDF, Inlet velocity	None	$[72; 72; 72]$	Pressure	Standard

To evaluate our model and compare it against the baselines, we selected two datasets with regularly sampled domains and two datasets with non-uniform outputs.

Darcy flow This dataset contains steady-state 2D solutions of the Darcy flow equation on the unit box, modeling the flow of fluid through a porous medium [9]. It includes samples with spatial resolution of 241×241 points. The network received a scalar permeability field as input and was tasked to predict scalar pressure values. To ensure non-periodicity, we applied padding during training, increasing the resolution to 271×271. A standard scaler was applied to both input and output fields. Train and test splits had 1024 samples each.

Navier-Stokes The Navier–Stokes dataset includes simulations of incompressible, viscous fluid flow with constant density and viscosity $\nu = 10^{-5}$ [9]. It features a periodic 2D domain of resolution 64×64, with 20 time steps of the vorticity field evolution. We trained models to predict the final 10 time steps from the initial 10, using 1000 training and 200 test samples. Inputs were composed as 3D tensors containing 10 time steps for every spatial point. Padding was introduced only to the time dimension, making the input to be of $64 \times 64 \times 15$. We appended spatial coordinates uniformly sampled in $[0, 2\pi)$ and a time coordinate with an integer index $t = [0, 1, \dots 9]$, which resulted in 4 input channels that were mapped to one channel.

ShapeNet car This benchmark contains 661 car meshes paired with Reynolds-Averaged Navier–Stokes (RANS) simulations [59]. We focus on predicting the scalar pressure field at the mesh vertices, which are fixed in size at 3,586 points per mesh. The dataset is split into 561 training and 100 test samples. Target pressures were standardized during training. Following Li et al. [10], inputs are constructed by sampling a fixed-resolution 64^3 grid over the global bounding box encompassing all meshes. A signed distance function (SDF) is computed per mesh. Grids are padded to 72^3 before being passed to the network.

Ahmed bodies The Ahmed bodies dataset [60] follows a similar setup to the ShapeNet car but introduces varying inlet velocities as an additional signal. This constant is appended to the SDF input, resulting in two input channels. Meshes vary in size, containing between 90,000 and 200,000 vertices. The grid resolution, padding, and target scaling match those used in the ShapeNet car.

The GINO-decoder architecture [10] used for both ShapeNet and Ahmed datasets requires precomputed fixed-radius neighborhoods to map grid samples to mesh vertices. We use the radii from the original paper: 0.05 for the ShapeNet car and 0.035 for Ahmed bodies.

For models trained on non-uniform data, we additionally apply Transformer-style positional encoding [64] to the input features, grid locations, and mesh vertices.

Results in the main text are reported on the *unscaled* targets. Raw dataset specifications are provided in Table 3, and data transformations are detailed in Table 4.

B Configurations of trained architectures

Table 5: Baseline configurations. GF indicates Gradient Features

Dataset	Model	d_c	Modes	Rank r	GF	Extra parameters
Darcy flow	FNO	32	[12, 12]	1	False	
	FNO_{g}	32	[12, 12]	1	True	
	TFNO	32	[12, 12]	10^{-4}	False	
	TFNO_{g}	32	[12, 12]	10^{-4}	True	
	$\text{DINOZAUR}_{\text{no-g}}$	32	[12, 12]	No	False	
	DINOZAUR	32	[12, 12]	No	True	
NS 2D+T	FNO	64	[16, 16, 4]	1	False	
	FNO_{g}	64	[16, 16, 4]	1	True	
	TFNO	64	[16, 16, 4]	10^{-4}	False	
	TFNO_{g}	64	[16, 16, 4]	10^{-4}	True	
	$\text{DINOZAUR}_{\text{no-g}}$	64	[16, 16, 4]	No	False	
	DINOZAUR	64	[16, 16, 4]	No	True	
ShapeNet car	GINO	64	[24, 24, 24]	0.4	False	
	GINO_{g}	64	[24, 24, 24]	0.4	True	
	TGINO	64	[24, 24, 24]	10^{-4}	False	
	TGINO_{g}	64	[24, 24, 24]	10^{-4}	True	
	$\text{DINOZAUR}_{\text{no-g}}$	64	[24, 24, 24]	No	False	
	DINOZAUR	64	[24, 24, 24]	No	True	
	GINO_{D}	64	[24, 24, 24]	0.4	False	
	TGINO_{D}	64	[24, 24, 24]	10^{-4}	False	$\text{D} = 0.1$, $\sigma^2 = e^{-4}$
	$\text{DINOZAUR}_{\text{D}}$	64	[24, 24, 24]	No	True	
	GINO_{L}	64	[24, 24, 24]	0.4	False	$\sigma_H^2 = e^{-4}$,
	TGINO_{L}	64	[24, 24, 24]	10^{-4}	False	$C = 500$,
	$\text{DINOZAUR}_{\text{L}}$	64	[24, 24, 24]	No	True	$\alpha = 10^6$
	$\text{DINOZAUR}_{\text{B}}$	64	[24, 24, 24]	No	True	$\mathbf{N}_{(\ln \tau)}(\ln 0.01, 1)$, $\mathbf{N}_{(\ln \tau \mid \mathcal{D})}(-5, 0.5)$
Ahmed bodies	GINO	64	[24, 24, 24]	0.4	False	
	GINO_{g}	64	[24, 24, 24]	0.4	True	
	TGINO	64	[24, 24, 24]	10^{-4}	False	
	TGINO_{g}	64	[24, 24, 24]	10^{-4}	True	
	$\text{DINOZAUR}_{\text{no-g}}$	64	[24, 24, 24]	No	False	
	DINOZAUR	64	[24, 24, 24]	No	True	

All results reported in our paper are from experiments conducted *specifically* for this study to ensure a fair and consistent comparison across models. Architectural settings for the baselines are summarized in Table 5. All models featured 4 FNO blocks with GELU activations. The channel width d_c and the number of modes were selected to align with the original configurations of FNO [9] and GINO [10], with the exception of Navier-Stokes data. We increased modes and channels to reflect the complexity of this benchmark. For models employing Tucker factorization, the rank r was either adopted directly

from the original implementation (GINO) or chosen to match the total number of parameters in DINOZAUR.

For all models incorporating dropout, the probability for a weight element to be zero was set to $\mathrm{D} = 0.1$ during training.

For the Laplace approximation, we fit the Hessian of the Negative log-likelihood loss (assuming homoscedastic noise with $\sigma_H^2 = \exp(-4)$) with respect to the parameters $\theta_{\mathcal{Q}}$ of the final (linear) layer $\mathcal{Q}$ of each model. Following Ritter et al. [65], we scale the Hessian and regularize it by adding a multiple of the identity matrix, yielding

$$H_{C,\alpha} = C\mathbb{E}\left[-\frac{\partial^2 \log p(\mathcal{D} \mid \theta_{\mathcal{Q}})}{\partial \theta^2}\right] + \alpha I$$

where the expectation is approximated via Monte Carlo samples, and C and α are hyperparameters selected on a validation set. In our experiments, we set $C = 500$, $\alpha = 10^6$

$\text{DINOZAUR}_{\mathrm{B}}$ requires us to set the prior distribution parameters for each $\ln \tau_i^c$ and specify the initial values for the approximate posterior $\ln \tau_i^c \mid \mathcal{D}$; we initialize this distribution with a full-rank covariance matrix within each NO block. Distribution parameters are informed by the time distributions in the deterministic experiment on ShapeNet car (Figure 5).

Table 6: Baseline optimization parameters

Dataset	Model	Optimizer	N epochs	Scheduler	LR
Darcy flow	FNO	AdamW	150	One-Cycle	10^{-3}
	FNO_{g}	AdamW	150	One-Cycle	10^{-3}
	TFNO	AdamW Schedule-Free	150	None	10^{-3}
	TFNO_{g}	AdamW	150	One-Cycle	10^{-3}
	$\text{DINOZAUR}_{\text{no-g}}$	AdamW	150	One-Cycle	10^{-3}
	DINOZAUR	AdamW	150	One-Cycle	10^{-3}
NS 2D+T	FNO	AdamW	150	One-Cycle	10^{-3}
	FNO_{g}	AdamW	150	One-Cycle	10^{-3}
	TFNO	AdamW	150	One-Cycle	10^{-3}
	TFNO_{g}	AdamW	150	One-Cycle	10^{-3}
	$\text{DINOZAUR}_{\text{no-g}}$	AdamW	150	One-Cycle	10^{-2}
	DINOZAUR	AdamW	150	One-Cycle	10^{-2}
ShapeNet car	GINO	AdamW Schedule-Free	150	None	10^{-3}
	GINO_{g}	AdamW Schedule-Free	150	None	10^{-3}
	TGINO	AdamW Schedule-Free	150	None	10^{-3}
	$\text{TGINO}_{\mathrm{g}}$	AdamW	150	One-Cycle	10^{-3}
	$\text{DINOZAUR}_{\text{no-g}}$	AdamW	150	One-Cycle	10^{-3}
	DINOZAUR	AdamW	150	One-Cycle	10^{-3}
	GINO_{D}	AdamW Schedule-Free	150	None	10^{-3}
	GINO_{L}	AdamW Schedule-Free	150	None	10^{-3}
	$\text{TGINO}_{\mathrm{D}}$	AdamW Schedule-Free	150	None	10^{-3}
	$\text{TGINO}_{\mathrm{L}}$	AdamW Schedule-Free	150	None	10^{-3}
	$\text{DINOZAUR}_{\mathrm{D}}$	AdamW	150	One-Cycle	10^{-3}
	$\text{DINOZAUR}_{\mathrm{L}}$	AdamW	150	One-Cycle	10^{-3}
	$\text{DINOZAUR}_{\mathrm{B}}$	AdamW	200	One-Cycle	10^{-3}
Ahmed bodies	GINO	AdamW Schedule-Free	150	None	10^{-3}
	GINO_{g}	AdamW Schedule-Free	150	None	10^{-3}
	TGINO	AdamW	150	One-Cycle	10^{-3}
	$\text{TGINO}_{\mathrm{g}}$	AdamW	150	One-Cycle	10^{-3}
	$\text{DINOZAUR}_{\text{no-g}}$	AdamW	150	One-Cycle	10^{-3}
	DINOZAUR	AdamW	150	One-Cycle	5×10^{-3}

Optimization hyperparameters are detailed in Table 6. All models were trained using either AdamW Schedule-Free [57] or AdamW combined with a One-Cycle scheduler [56], weight decay set to 10^{-4}.

Both options allow for faster convergence with fewer epochs and reduce sensitivity to hyperparameter choices. The final setup was selected by comparing both options and reporting the results for the configuration that achieved the best performance. A limited learning rate sweep was conducted in the range $[10^{-4}, 10^{-2}]$, with early termination applied to underperforming runs. For the One-Cycle scheduler, the learning rate in the table refers to the max LR parameter.

C Distributions of diffusion times

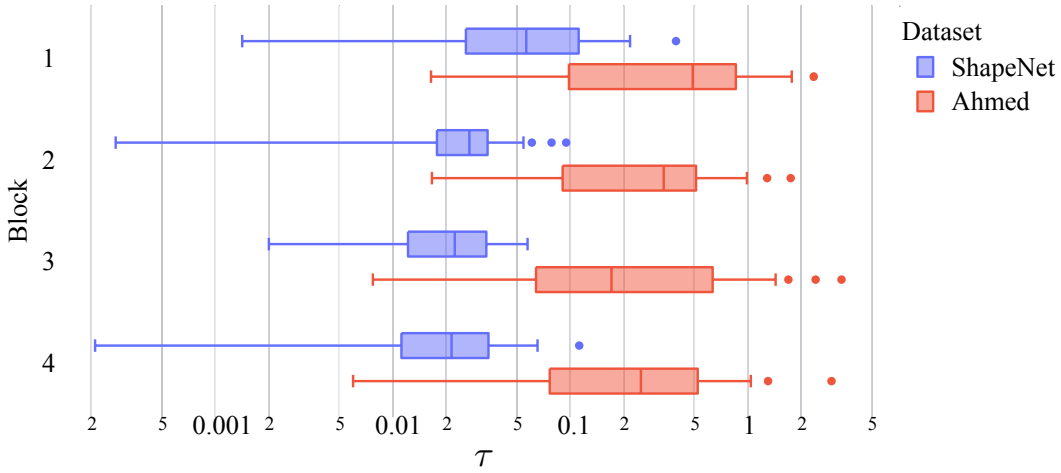

Figure 5: Diffusion times gathered from FNO blocks of deterministic DINOZAUR trained on non-uniform datasets.

Having trained the deterministic DINOZAUR on the non-uniform datasets ShapeNet car and Ahmed bodies, we analyze the behavior of the learned diffusion times across the network. Figure 5 shows the empirical distributions of the 64 diffusion times extracted from each block of the neural operator. We observe two main trends. First, diffusion times generally decrease with network depth. This aligns with the findings of Sharp et al. [34] and suggests that deeper blocks capture finer-scale features, whereas earlier blocks encode broader, more global structures. Second, diffusion times learned on the Ahmed bodies dataset are consistently larger than those learned on the ShapeNet car. We attribute this to the presence of varying inlet velocities in the Ahmed bodies dataset, which increases the variability in the data and may require coarser diffusion scales to capture the broader range of feature patterns effectively.

For the Bayesian version of our model, we initialize all blocks with the same prior distributions for $\ln \tau$ and likewise set identical initial guesses for the corresponding variational posteriors. This initialization allows each block to adapt its uncertainty about diffusion time scales through learning. To visualize this, we draw samples from the approximate posteriors and estimate their empirical location and scale parameters, characterizing the learned diffusion behavior in the probabilistic setting. Figure 6 illustrates these posterior distributions alongside the prior. We find the same consistent pattern: posteriors systematically shift toward shorter diffusion times as depth increases, and their spread becomes tighter. This mirrors the trend observed in the deterministic setting (Figure 5), reinforcing the interpretation that deeper blocks specialize in capturing high-frequency, localized features. The prior has a strong influence on the resulting posteriors, forcing diffusion times to be lower (closer to the prior mean, which was chosen to promote shorter ranges for message passing).

Finally, in Figure 7, we provide the full covariance matrices for each of the posteriors that were learned by the network. Analyzing covariances, we note that they are dominated by the diagonals with few spots of strong correlation off-diagonal. This indicates a good separation of information within features.

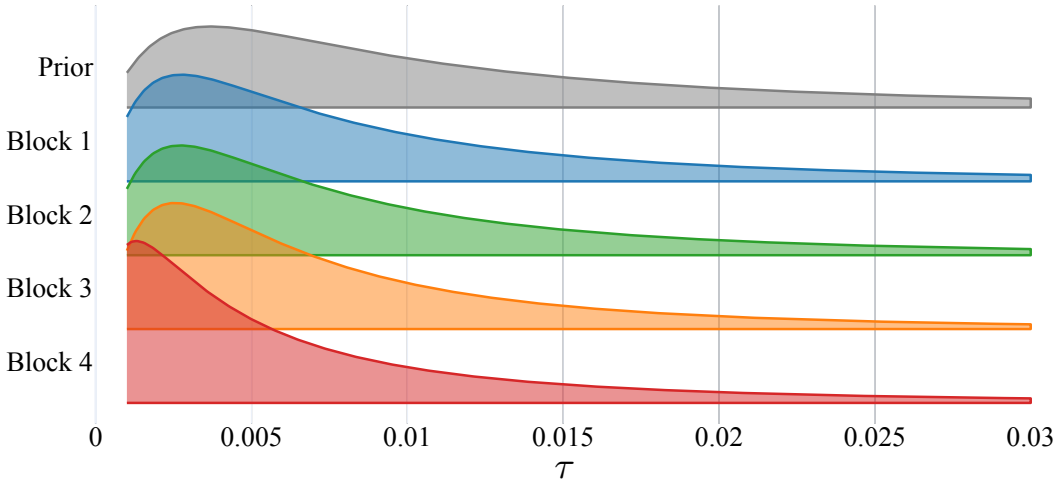

Figure 6: Prior distribution set according to $\ln \tau \sim \mathbf{N}(\ln 0.01, 1)$ and approximate posteriors gathered from the NO blocks of DINOZAUR$_{\mathrm{B}}$ trained on ShapeNet car dataset through $\mathcal{L}_{\mathrm{ELBO}}$ maximization.

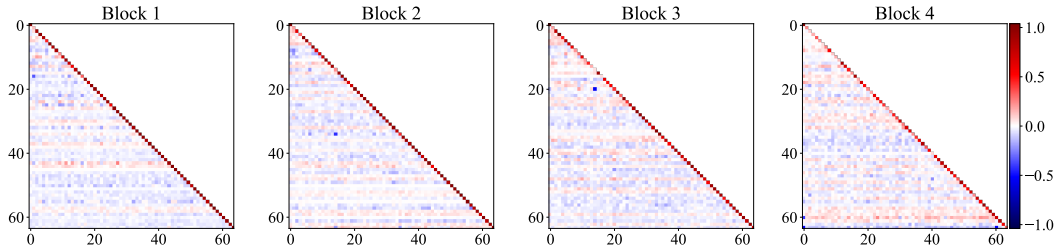

Figure 7: Lower diagonal full-rank learned covariance matrices of the posterior distributions. Each matrix represents a $d_c \times d_c$ (64×64) covariance matrix of the multi-dimensional distribution of diffusion times within each block.

D Metrics definitions

Given a test set element index $n = 1, \ldots, N_{\text{test}}$ and a point evaluation index $j = 1, \ldots, J_n$, let u_{nj} denote the ground truth observation of the target function u for element n at point j, and let $\widehat{u}_{nj}$ denote the corresponding neural operator prediction. For baseline models that support uncertainty quantification via sampling, $\widehat{u}_{nj}$ represents the empirical mean over samples, and $\widehat{\sigma}_{nj}$ denotes the empirical standard deviation. In all probabilistic experiments, we generated 100 samples per prediction.

Relative $\mathbf{L_2}$ The RL_2 loss is a normalized metric that quantifies the discrepancy between predicted outputs and ground truth values, scaled by the squared norm of the ground truth. This normalization allows for fair comparisons across datasets or tasks with varying magnitudes and units.

$$\mathrm{RL}_2 = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \frac{\sqrt{\sum_{j=1}^{J_n} (u_{nj} - \widehat{u}_{nj})^2}}{\sqrt{\sum_{j=1}^{J_n} u_{nj}^2}}.$$

Negative log-likelihood NLL reflects the mean log-likelihood of the true values under the predicted Gaussian distribution. Lower NLL indicates better uncertainty calibration and fit to the data.

$$\mathrm{NLL} = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \frac{1}{J_n} \sum_{j=1}^{J_n} \left[\ln(\widehat{\sigma}_{nj} \sqrt{2\pi}) + \frac{(u_{nj} - \widehat{u}_{nj})^2}{2\widehat{\sigma}_{nj}^2} \right].$$

This averaged formula was used only for reporting metrics for probabilistic models.

Miscalibration area The MA is a scalar metric that quantifies the discrepancy between predicted confidence levels and the actual observed frequencies of events, providing a measure of uncertainty calibration. It captures the area between the model's empirical calibration curve and the ideal diagonal (perfect calibration). Lower values indicate better uncertainty calibration. Let $\pi^{(k)} = \frac{k}{K}$ for $k = 0, \ldots, K$, denote the expected coverage levels. For each test sample n, the observed coverage at level $\pi^{(k)}$ is given by:

$$\widehat{\pi}_n^{(k)} = \frac{1}{J_n} \sum_{j=1}^{J_n} \mathbb{1}\left(\frac{|u_{nj} - \widehat{u}_{nj}|}{\widehat{\sigma}_{nj}} \leq \Phi^{-1}\left(\frac{1 + \pi^{(k)}}{2} \right) \right),$$

where Φ^{-1} is the inverse of the standard normal cumulative distribution function. Then, the miscalibration area across all elements of the test set is given by averaging the area estimates for each of the elements:

$$\mathrm{MA} = \frac{1}{N_{\mathrm{test}}} \sum_{n=1}^{N_{\mathrm{test}}} \sum_{k=0}^{K-1} \left| \frac{(\pi^{(k+1)} - \pi^{(k)})}{2} \left[\widehat{\pi}_n^{(k)} - \pi^{(k)} + \widehat{\pi}_n^{(k+1)} - \pi^{(k+1)} \right] \right|.$$

Interval score IS is another metric for evaluating predictive uncertainty. It balances the width of the prediction interval with a penalty for ground truth values falling outside the interval. For a chosen coverage level $p \in (0, 1)$, define the lower and upper predictive bounds:

$$\underline{b_{nj}^{(p)}} = \widehat{u}_{nj} + \widehat{\sigma}_{nj} \cdot \Phi^{-1}\left(\frac{1-p}{2} \right) \quad \text{and} \quad \overline{b_{nj}^{(p)}} = \widehat{u}_{nj} + \widehat{\sigma}_{nj} \cdot \Phi^{-1}\left(\frac{1+p}{2} \right)$$

where Φ^{-1} is the inverse of the standard normal cumulative distribution function. Then the interval score at level p for prediction at point j in sample n is:

$$\mathrm{IS}_{nj}^{(p)} = \left(\overline{b_{nj}^{(p)}} - \underline{b_{nj}^{(p)}} \right) + \frac{2}{1-p} \cdot \left(\underline{b_i^{(p)}} - u_i \right) \cdot \mathbb{1}\left(u_i < \underline{b_i^{(p)}} \right) + \frac{2}{1-p} \cdot \left(u_i - \overline{b_i^{(p)}} \right) \cdot \mathbb{1}\left(u_i > \overline{b_i^{(p)}} \right).$$

To compute the overall interval score, we average over a predefined set of P equally spaced coverage levels ($p = [0.01, 0.02, \ldots 0.99]$):

$$\mathrm{IS} = \frac{1}{N_{\mathrm{test}}} \sum_{n=1}^{N_{\mathrm{test}}} \frac{1}{P} \sum_{p} \frac{1}{J_n} \sum_{j=1}^{J_n} \mathrm{IS}_{nj}^{(p)}.$$

E Elaboration on the universal approximation

Here, we formulate the statement of Proposition 1, reported in the main text, more strictly.

Following the exposition of Lanthaler et al. [66], let $\Omega \subset \mathbb{R}^d$ be a bounded domain with Lipschitz boundary and let $\mathcal{A}(\Omega, \mathbb{R}^{d_a})$, $\mathcal{U}(\Omega, \mathbb{R}^{d_u})$, $\mathcal{V}(\Omega, \mathbb{R}^{d_c})$ denote Banach space of functions on Ω. DINOZAUR architecture defines a mapping

$$\mathcal{N}_{\theta,\tau} : \mathcal{A}(\Omega, \mathbb{R}^{d_a}) \to \mathcal{U}(\Omega, \mathbb{R}^{d_u}),$$

which can be written as a composition of a form:

$$\mathcal{N}_{\theta,\tau} = \mathcal{Q} \circ \mathcal{N}_{\theta_M,\tau_M} \circ \cdots \circ \mathcal{N}_{\theta_1,\tau_1} \circ \mathcal{P},$$

consisting of lifting layer $\mathcal{P}$, hidden blocks $\mathcal{N}_{\theta_i,\tau_i}$, $i = 1, \ldots, M$, and a projection layer $\mathcal{Q}$. Given channel dimension d_c, the lifting layer $\mathcal{P}$ is given by mapping:

$$\mathcal{P} : \mathcal{A}(\Omega, \mathbb{R}^{d_a}) \to \mathcal{V}(\Omega, \mathbb{R}^{d_c}), \quad a(x) \mapsto P(a(x), x),$$

where $P : \mathbb{R}^{d_a} \times \Omega \to \mathbb{R}^{d_c}$ is a learnable neural network acting between finite-dimensional Euclidean spaces. For $i = 1, \ldots, M$, each block $\mathcal{N}_{\theta_i,\tau_i}$ is of the form

$$\mathcal{N}_{\theta_i,\tau_i}[v](x) := \sigma\left(W_i^{\mathrm{skip}} v(x) + b_i + W_i^{\mathrm{mix}} \begin{bmatrix} \mathcal{F}^{-1}\left[\exp(-4\pi^2\|k\|^2\tau_i) \odot \mathcal{F}[v](k)\right](x) \\ \mathcal{G}_{W_i^{\mathrm{grad}}}\left[\mathcal{F}^{-1}\left[\exp(-4\pi^2\|k\|^2\tau_i) \odot \mathcal{F}[v](k)\right]\right](x) \end{bmatrix} \right).$$

Each hidden block defines a mapping $\mathcal{N}_{\theta_i,\tau_i} : \mathcal{V}(\Omega, \mathbb{R}^{d_c}) \to \mathcal{V}(\Omega, \mathbb{R}^{d_c})$. For $i = 1, \ldots, M$, the matrices $W_i^{\text{skip}}, W_i^{\text{grad}} \in \mathbb{R}^{d_c \times d_c}$, $W_i^{\text{mix}} \in \mathbb{R}^{d_c \times 2d_c}$ diffusion times $\tau_i \in \mathbb{R}^{d_c}$ and bias $b_i \in \mathbb{R}^{d_c}$ are learnable parameters. The non-linearity $\sigma : \mathbb{R} \to \mathbb{R}$ acts element-wise and is assumed to be smooth, non-polynomial, and Lipschitz-continuous. Finally, the projection layer $\mathcal{Q}$ is given by a mapping:

$$\mathcal{Q} : \mathcal{V}(\Omega, \mathbb{R}^{d_c}) \to \mathcal{U}(\Omega, \mathbb{R}^{d_u}), \quad v(x) \mapsto Q(v(x), x),$$

where $Q : \mathbb{R}^{d_c} \times \Omega \to \mathbb{R}^{d_u}$ is also a learnable neural network acting between finite-dimensional Euclidean spaces.

From now on, we will focus on the case of a single hidden block:

$$\mathcal{N}_{\theta,\tau}[a](x) = (\mathcal{Q} \circ \mathcal{N}_{\theta_1,\tau} \circ \mathcal{P})[a](x).$$

Now, we re-state the Proposition 1:

Proposition 1. *Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with Lipschitz boundary and such that the closure $\overline{\Omega} \subset (0, 2\pi)^d$. For given integers $s, s' > 0$, let $\mathcal{N} : C^s(\overline{\Omega}; \mathbb{R}^{d_a}) \to C^{s'}(\overline{\Omega}; \mathbb{R}^{d_u})$ be a continuous operator, and fix a compact set $\mathbf{A} \subset C^s(\overline{\Omega}; \mathbb{R}^{d_a})$. Then there exists a continuous, linear operator $\mathcal{E} : C^s(\Omega; \mathbb{R}^{d_a}) \to C^s(\mathbb{T}^d; \mathbb{R}^{d_a})$ such that $\mathcal{E}[a]|_\Omega = a$ for all $a \in C^s(\Omega; \mathbb{R}^{d_a})$. Furthermore, for any $\varepsilon > 0$, there exists a* DINOZAUR such that

$$\sup_{a \in \mathbf{A}} \|\mathcal{N}[a] - (\mathcal{N}_{\theta,\tau} \circ \mathcal{E}[a])|_\Omega\|_{C^{s'}(\overline{\Omega}; \mathbb{R}^{d_u})} \leq \varepsilon,$$

where $\mathcal{N}_{\theta,\tau} = \mathcal{Q} \circ \mathcal{N}_{\theta_1,\tau} \circ \mathcal{P} : C^s(\mathbb{T}^d; \mathbb{R}^{d_a}) \to C^{s'}(\mathbb{T}^d, \mathbb{R}^{d_u})$.

Proof. First, we consider a periodic torus $\mathbb{T}^d$. Without loss of generality, since the weights of the network are free to be updated, we let the matrices W_i^{mix} be of the block structure

$$W_i^{\text{mix}} = \begin{bmatrix} \frac{1}{|\mathbb{T}^d|} I \\ 0 \end{bmatrix},$$

with $I \in \mathbb{R}^{d_c \times d_c}$ being the identity matrix and $0 \in \mathbb{R}^{d_c \times d_c}$ a matrix of all zeros, which renders a hidden block to:

$$\mathcal{N}_{\theta_1,\tau}[v](x) = \sigma\left(W_1^{\text{skip}} v(x) + b_1 + \frac{1}{|\mathbb{T}^d|} \mathcal{F}^{-1}\left[\exp(-4\pi^2 \|k\|^2 \tau) \odot \mathcal{F}[v](k)\right](x)\right).$$

By letting diffusion times $\tau \to \infty$, we get

$$\exp(-4\pi^2 \|k\|^2 \tau) = 1, \ \|k\| = 0,$$
$$\exp(-4\pi^2 \|k\|^2 \tau) \to 0, \ \|k\| > 0,$$

which, in the limit, results in the block:

$$\begin{aligned} \mathcal{N}_{\theta_1,\infty}[v](x) &= \sigma\left(W_1^{\text{skip}} v(x) + b_1 + \frac{1}{|\mathbb{T}^d|} \mathcal{F}^{-1}\left[\mathcal{F}[v](0)\right](x)\right) \\ &= \sigma\left(W_1^{\text{skip}} v(x) + b_1 + \fint_{\mathbb{T}^d} v(y) \mathrm{d}y\right), \end{aligned}$$

making a full network $\mathcal{N}_{\theta,\infty} = \mathcal{Q} \circ \mathcal{N}_{\theta_1,\infty} \circ \mathcal{P}$ an averaging neural operator in the sense of **Equation 2.3** in [66] on the torus $\mathbb{T}^d$. By **Corollary 3.3** [66], $\mathcal{N}_{\theta,\infty}$ is a universal approximator on the torus. Here, we note that the lifting and projection operators defined in [66] have an explicit positional dependency, and thus, one should understand them as periodized MLPs. [1]

Now consider a general domain $\Omega \subset (0, 2\pi)^d$. By **Lemma 41** by Kovachki et al. [67] and **Lemma A1** in [66], there exists a continuous, linear extension operator $\mathcal{E} : C^s(\Omega; \mathbb{R}^{d_a}) \to C^s(\mathbb{T}^d; \mathbb{R}^{d_a})$ such that $\mathcal{E}[a]|_\Omega = a$ and $\mathcal{E}[a]$ is periodic on $[0, 2\pi]^d$ for all $a \in C^s(\Omega; \mathbb{R}^{d_a})$. We note that the conditions of the **Theorem 9** in [67] are satisfied. By following the proof of this theorem, we apply the extension operator $\mathcal{E}$ and use the universal approximation on the torus, establishing universal approximation on Ω. □

[1] Note that **Theorem 2.1** in [66] as it is formulated, applies to bounded Lipschitz domains. Thus, the proof of **Corollary 3.3** [66] is not immediate. One must either follow the proof of **Theorem 2.1** to see that it applies to the torus, or first formulate it on the domain $(0, 2\pi)^d$, invoke the **Theorem 2.1**, and then argue that it holds when one restricts to periodic functions on $(0, 2\pi)^d$.

F Broader impact

This study presents a neural operator tailored for the efficient and scalable solution of partial differential equations, intending to advance computational modeling in scientific and engineering disciplines. By enabling more resource-efficient simulations, our method has the potential to accelerate research in various domains where PDEs play a central role. Furthermore, the integration of uncertainty quantification in our model supports more robust decision-making, which is particularly valuable in high-stakes applications.

While our work is primarily theoretical and presents a low immediate risk, we recognize that advances in machine learning for scientific computing may influence decisions in sensitive contexts. As neural operators become more widely adopted, it will be essential to ensure that they are used responsibly, particularly in applications where predictive reliability and interpretability are critical. Ongoing dialogue among machine learning researchers, domain experts, and the general public is essential to ensure that the development and deployment of such tools remain aligned with societal values and serve the public interest.