GRIFDIR: GRAPH RESOLUTION-INVARIANT DIFFUSION MODELS OVER IRREGULAR DOMAINS

Anonymous authors
Paper under double-blind review

ABSTRACT

Score-based diffusion models in infinite-dimensional function spaces provide a mathematically principled framework for modelling function-valued data, offering key advantages such as resolution invariance and the ability to handle irregular discretisations. However, practical implementations have struggled to fully realise these benefits. Existing backbones like Fourier neural operators are often biased towards regular grids and fail to generalise to complex domain topologies. We introduce an architecture, GRIFDIR, for function-space diffusion models that represents generalised graph convolutional kernels as finite element functions, allowing the score network to operate directly on unstructured meshes over domains of arbitrary shape. We demonstrate the efficacy of our network architecture through a series of unconditional and conditional sampling experiments across diverse geometries, including non-convex and multiply-connected domains. Our results show that the proposed method maintains resolution invariance and achieves high fidelity in capturing functional distributions on non-trivial geometries.

1 INTRODUCTION

Score-based diffusion models have recently been extended to infinite-dimensional Hilbert spaces, providing a principled framework for modelling function-valued data (Lim et al., 2025; Pidstrigach et al., 2024; Hagemann et al., 2025). This is particularly relevant for scientific machine learning applications, where data arises naturally as solutions to partial differential equations (PDEs), spatial fields, or discretisations of continuous operators (Yao et al., 2025; Baker et al., 2026; Baldassari et al., 2023). Instead of modelling finite-dimensional vectors, infinite-dimensional diffusion models learn distributions directly over functions.

There are two main approaches for applying diffusion models to function-valued data. The first is the *discretise-first* approach, where functions are discretised onto a fixed grid and standard finite-dimensional diffusion models are applied. The second is the *discretise-last* approach, where diffusion models are formulated directly in function space and discretisation is introduced only at training or sampling time. This provides several desirable properties:

1. **Resolution invariance**: Models trained at one resolution can generalise to different resolutions.
2. **Handling irregular data**: Models naturally support unstructured meshes and complex geometries.
3. **Theoretical guarantees:** Convergence results hold independently of the discretisation resolution.

Furthermore, modelling directly in function spaces allows the incorporation of prior knowledge about the structure of the data, such as smoothness or Hilbert space constraints, which can improve both modelling flexibility and theoretical convergence guarantees (Pidstrigach et al., 2024). Despite these advantages, existing numerical implementations do not fully realise these theoretical benefits. Convolutional U-Net architectures perform well on fixed, regular grids but lack resolution invariance and struggle with irregular geometries (Ronneberger et al., 2015; Hagemann et al., 2025). Fourier Neural Operators (FNOs) (Li et al., 2021), on the other hand, provide resolution invariance but do not naturally handle irregular geometries or even non-regular grids. There is still a gap between the theoretical promises of infinite-dimensional diffusion models and practical architectures that fully exploit these promises.

Contribution We identify key properties that a neural network architecture should satisfy in order to fully leverage the advantages of the infinite-dimensional diffusion model paradigm over the finite-

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

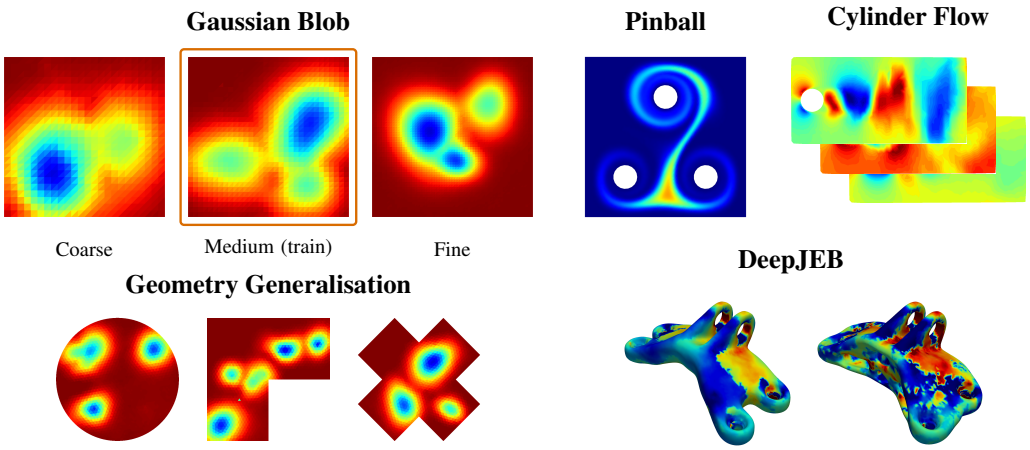


Figure 1: GRIFDIR allows sampling from different resolutions (see `GaussianBlobs`) without retraining, as well as works on non standard domains, e.g., including holes and non-convex geometries. Further, GRIFDIR scales to 3D generation. Details for these experiments in Section 4.

dimension paradigm, including resolution invariance and support for irregular domain geometries. We explore the abilities of existing models to fulfil these key properties. We further propose **GRIFDIR** (Graph Resolution-Invariant FEM Diffusion over Irregular domains), a graph neural operator for infinite-dimensional diffusion models. Our approach is designed to satisfy the identified desirable properties, while remaining computationally efficient and easy to train. We demonstrate that the proposed architecture enables practical function-space diffusion modelling across varying domains and discretisations (see Figure 1). We further present conditional sampling results.

The paper is structured as follows: In Section 2 we provide background on infinite-dimensional diffusion models and recent conditional sampling techniques. In Section 3 we provide desiderata on suitable network architectures and explain the GNN-based network architecture. In Section 4.1 we explain the experiment setups and in Sections 4.2 to 4.4, we show results on varying domains and datasets.

2 Infinite-dimensional Diffusion Models

Similarly to finite-dimensions, we define score-based diffusion models in infinite-dimensions with a forward and reverse stochastic differential equation (SDE). However, the SDEs are defined directly in a Hilbert space $\mathcal{H}$ and some care has to be taken in defining the time-reversal since Lebesgue measures (and therefore the standard densities) do not exist in Hilbert space. We will refer to elements in the Hilbert space by $a \in \mathcal{H}$ and with A (and A_t) to random variables with realisations in $\mathcal{H}$.

Following Pidstrigach et al. (2024) we consider the forward SDE starting from the prior data distribution π, with support in $\mathcal{H}$, with C a trace-class covariance operator and $W^{\mathcal{H}}$ the cylindrical Wiener process associated to $\mathcal{H}$

$$dA_t = -\frac{1}{2}A_t dt + \sqrt{C}dW_t^{\mathcal{H}}, \quad A_0 \sim \pi. \tag{1}$$

The time-reversal $Z_t \coloneqq A_{T-t}$ is given by

$$\mathrm{d}Z_t = \left[\frac{1}{2}Z_t + s(T-t, Z_t)\right]\mathrm{d}t + \sqrt{C}\mathrm{d}W_t^{\mathcal{H}}, \tag{2}$$

$$s(t,a) \coloneqq -(1-e^{-t})^{-1}\left(a - e^{-\frac{t}{2}}\mathbb{E}[A_0|A_t = a]\right), \tag{3}$$

where $Z_0 \sim \mathrm{Law}(A_T) \approx \mathcal{N}(0, C)$ and $Z_T \sim \pi$. Alternatively, we can formulate the Variance Exploding (VE) SDE (Song et al., 2021) in this infinite-dimensional setting. Let $\sigma_t > 0$ be a monotonically increasing noise schedule. Then the forward VE-SDE is given by

$$\mathrm{d}A_t = g_t\sqrt{C}\mathrm{d}W_t^{\mathcal{H}}, \quad A_0 \sim \pi, \text{ where } g_t \coloneqq \sqrt{\frac{\mathrm{d}}{\mathrm{d}t}\sigma_t^2}. \tag{4}$$

The transition measure is given by $\text{Law}(A_t \mid A_0) = \mathcal{N}(A_0, \sigma_t^2 C)$. Similarly to the previous case, we define a score operator $s_{\text{VE}}(t, a)$ which absorbs C:

$$s_{\text{VE}}(t, a) \coloneqq -\frac{1}{\sigma_t^2}\left(a - \mathbb{E}[A_0 \mid A_t = a]\right), \tag{5}$$

$$\mathrm{d}Z_t = g_{T-t}^2 s_{\text{VE}}(T-t, Z_t)\mathrm{d}t + g_{T-t}\sqrt{C}\mathrm{d}W_t^{\mathcal{H}}, \tag{6}$$

where $Z_0 \sim \text{Law}(A_T) \approx \mathcal{N}(0, \sigma_T^2 C)$ and $Z_T \sim \pi$. One can then use the infinite-dimensional version of score matching to learn $s(t, a)$ or $s_{\text{VE}}(t, a)$ from data.

2.1 Conditional Sampling

One class of function-valued sampling problems arises from (Bayesian) inverse problems. Let $L : \mathcal{H} \to \mathcal{Y}$ be a forward operator between Hilbert spaces, and the $Y = \mathcal{L}(a_0) + \eta$, with $\eta \in \mathcal{Y}$ being observation noise and $a_0 \sim \pi$. The inverse problem is to infer a_0 given some observation $Y = y$, which we model as sampling from the posterior distribution $\pi^y \coloneqq \pi_{|Y=y}$. Following Dashti & Stuart (2017), we assume that $\pi_y \ll \pi$ for y π-almost surely. Then, for some potential $\Phi : \mathcal{H} \times \mathcal{Y} \to \mathbb{R}$, we write the likelihood as

$$\frac{\mathrm{d}\pi^y}{\mathrm{d}\pi}(a_0) = \frac{1}{\xi}\exp(-\Phi(a_0, y)), \text{ where } \xi = \mathbb{E}_{a_0 \sim \pi}[\exp(-\Phi(a_0, y))]. \tag{7}$$

To sample from the conditional distribution π^y using diffusion models, one replaces the unconditional score in the time-reversed SDEs with a conditional score $s^y(t, a)$. While one can directly train a conditional model if paired data is available (Baldassari et al., 2023), a significant advantage of diffusion models is the ability to leverage a pre-trained, unconditional prior to solve various inverse problems without complete retraining.

In finite-dimensional settings, conditional sampling fundamentally relies on Bayes' theorem (see Appendix B.1). In function spaces, this relationship can be formalised via the h-transform (Rogers & Williams, 2000). The conditional score splits into the unconditional score and a guidance term:

$$s^y(t, a) = s(t, a) + C\nabla \log h^y(t, a), \tag{8}$$

$$h^y(t, a) \coloneqq \xi^{-1}\mathbb{E}_{\mathbb{P}}[\exp(-\Phi(A_0, y)) \mid A_t = a], \tag{9}$$

where C denotes the covariance operator, $\mathbb{P}$ is the path measure of the forward SDE, and Φ is the potential defined in (7); see Theorem 3.2 in Baker et al. (2026) for a proof. Substituting this conditional score into the time-reversed SDEs yields samples from the posterior π^y.

Because the expectation in the guidance term (9) is generally intractable, practical implementations require approximations. Broadly, methods either circumvent the guidance term by learning a joint embedding (Yao et al., 2025), utilise training-free approximations like the Tweedie estimate (Lin et al., 2026), or perform supervised fine-tuning to explicitly learn the guidance (Baker et al., 2026). We defer a detailed discussion of these conditional sampling paradigms to Appendix B.2.

3 Architecture for Infinite-Dimensional Diffusion Models

3.1 Desirable Properties for the Diffusion Network

A central challenge in the infinite-dimensional framework is to parametrise the score function as a *neural operator* $s_\theta : [0, T] \times \mathcal{H} \to \mathcal{H}$ that maps between function spaces (Kovachki et al., 2023). While neural operators are most prominently used to learn solution operators of PDEs (Azizzadenesheli et al., 2024), they provide a natural backbone for parametrising the infinite-dimensional score. To fully leverage the infinite-dimensional framework on irregular data, the score network should exhibit four key properties:

1. **Multiscale processing:** Local fine-scale features and global structures must be captured, analogously to U-Nets.
2. **Resolution and discretisation invariance:** The operator must define a mapping between function spaces. A model trained at resolution N must generalise to $N' \neq N$ without retraining.
3. **Domain flexibility:** The architecture must be able to encode and decode functions on arbitrary and complex domain geometries without structural modification.

4. **Global structure:** The model should capture and maintain global information about the target distribution when evaluating the score.

The last point is essential, since the score function depends on the full input configuration, not only on local features. Architectures based solely on message passing propagate information through local interactions and therefore, lack an explicit global state (unless augmented with pooling or hierarchical mechanisms). As a result, they may struggle to capture long-range dependencies and global structure.

3.2 Existing Neural Operator Architectures

Grid-based neural operators Standard U-Nets satisfy Property 1 but fail Property 2 and Property 3, as convolutions are restricted to fixed regular grids. A widespread class of neural operators similarly relies on regular grids, including CNO (Raonic et al., 2023) and FNO (Li et al., 2021), which satisfy Property 1 and Property 2 up to spectral truncation but fail Property 3 on irregular geometries (Bartolucci et al., 2023). While adaptations exist, e.g., domain masking, non-uniform Fourier transforms (Lingsch et al., 2023), or spatial deformation maps (Li et al., 2023), they introduce computational overhead and additional training complexity.

Message-Passing GNN Message-Passing (MP) GNNs offer a natural alternative to handle complex geometries. Methods like MeshgraphNets (Pfaff et al., 2021) and MP-PDEs (Brandstetter et al., 2022) operate on unstructured meshes and thus satisfy domain flexibility. Given node features and physical coordinates, standard message passing iteratively updates each node by aggregating permutation-invariant messages from its neighbouring nodes and applying an update function. Standard MP-GNNs fail to achieve resolution invariance (Property 2). A K-layer MP-GNN aggregates information over a physical radius proportional to $K\bar{d}$, where $\bar{d}$ is the average edge length of the mesh. As the mesh is refined ($\bar{d} \to 0$), the physical receptive field shrinks.

Graph Neural Operators Graph Neural Operators (GNOs) (Anandkumar et al., 2019; Wen et al., 2025) attempt to resolve the shrinking receptive field of standard MP-GNNs by decoupling the aggregation neighbourhood from the topological mesh. Instead of aggregating over immediate graph edges, GNOs aggregate over a fixed physical radius r. For a signal $a : \Omega \to \mathbb{R}^c$, the operator $\mathcal{K}$ is defined as

$$(\mathcal{K}a)(x_i) = \int_{B_r(x_i)} \kappa_\theta(x_i, y)\, a(y)\, \mathrm{d}y \approx \frac{1}{|N_r(i)|} \sum_{j \in N_r(i)} \kappa_\theta(x_i, x_j)\, a(x_j), \tag{10}$$

where κ_θ is a learnable continuous kernel parametrised by a neural network, $B_r(x_i)$ is a ball of radius r centred at x_i, and $N_r(i)$ contains all nodes within $B_r(x_i)$. This formulation maintains a constant physical receptive field.

Continuous Kernels The choice of how to parametrise the kernel κ_θ is central to these operators. GNOs parametrise κ_θ as a continuous function of relative coordinates $x_j - x_i$. Common approaches include using B-spline bases as in SplineCNN (Fey et al., 2018), Gaussian mixture models in MoNet (Monti et al., 2017), or MLPs as seen in QuadConv (Doherty et al., 2024), NNConv (Simonovsky & Komodakis, 2017), PointNet++ (Qi et al., 2017), or PointTransformer (Zhao et al., 2021).

3.3 GRIFDIR: A Multiscale Diffusion GNN

We propose **GRIFDIR**: a multiscale graph neural operator defined over a hierarchy of meshes, combined with a finite-element-based convolution operator and a latent transformer. The architecture operates on functions defined over general meshes, achieving resolution-invariance (Property 2), support for complex geometries (Property 3), hierarchical multi-scale processing (Property 1), and a global compressed latent representation via the transformer (Property 4).

We first fix the notation of the mesh. We will define the architecture first for domains $\Omega \subset \mathbb{R}^2$ and later generalise this to higher dimensions. A discretisation of Ω is a finite set of nodes $\mathcal{T} = \{x_1, \dots, x_N\} \subset \Omega$ together with a finite element basis $\{\psi_j\}_{j=1}^N$. The basis functions ψ_j satisfy $\psi_j(x_k) = \delta_{jk}$ and are local, i.e., supp(ψ_j) has a diameter h_j around x_j (for piecewise linear elements h_j is the size of the largest triangle incident to x_j). We further assume that the basis functions are normalised, i.e., $\sum_{j=1}^N \psi_j(x) = 1$ for all $x \in \Omega$.

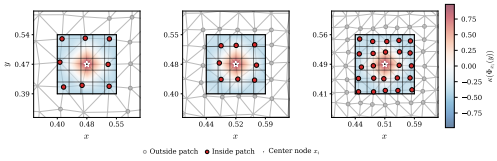


Figure 2: FEM convolution across increasing mesh resolutions. The reference patch and the learned filter (parameterised on a 5×5 grid) remain fixed in physical space, ensuring resolution-invariant aggregation over nodes (red dots).

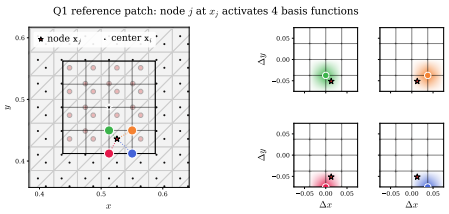


Figure 3: Q1 basis function for the FEM convolution for a reference patch. Every node only activates four neighbouring Q1 basis functions (red, blue, orange, green).

Resolution-invariant FEM convolution. We generalise CNN convolutions to unstructured meshes by parametrising the convolutional filter as a finite-element function on a fixed *reference patch* in physical space, and projecting neighbouring nodes into the filter's coordinate frame, rather than evaluating a learned kernel in the coordinate frame of the domain. In particular, we define the *reference patch* $\hat{\Omega} = [-1, 1]^2 \subset \mathbb{R}^2$ with a regular $P \times P$ Cartesian grid $\{\xi_p\}_{p=1}^{P^2} \subset \hat{\Omega}$ and a bilinear (Q1) basis $\{\phi_p\}_{p=1}^{P^2}$ with $\phi_p(\xi_q) = \delta_{pq}$. The learnable convolutional kernel is then defined as

$$\kappa : \hat{\Omega} \to \mathbb{R}, \quad \kappa(\xi) = \sum_{p=1}^{P^2} w_p \phi_p(\xi)$$

with $w = (w_1, \ldots, w_{P^2}) \in \mathbb{R}^{P^2}$ as trainable parameters. For every node x_i, let $B_r(x_i) := \{\, y \in \Omega : \|y - x_i\|_\infty \le r \,\}$ be the physical patch around x_i (a square of side $2r$, centred at x_i). We now use κ to define our FEM convolution. We project the nodes to the references patch via a rescaling map

$$\Phi_{x_i} : B_r(x_i) \to \hat{\Omega}, \qquad \Phi_{x_i}(y) = \frac{y - x_i}{r}.$$

We have $\Phi_{x_i}(x_i) = 0$. Composing the kernel with Φ_{x_i}, i.e. $\kappa(\Phi_{x_i}(\cdot))$, evaluates the fixed reference kernel in physical coordinates centred and scaled at x_i. For a feature field $a : \Omega \to \mathbb{R}$, the FEM convolution of a with κ at node x_i is

$$(\mathcal{K}a)(x_i) = \int_{B_r(x_i)} \kappa\big(\Phi_{x_i}(y)\big)\, a(y)\, \mathrm{d}y. \tag{11}$$

We evaluate (11) with a nodal (mass-lumped) quadrature rule on the mesh, replacing a by its interpolant $a_h = \sum_j a(x_j)\psi_j$ and using $\int_{B_r(x_i)} \psi_j(y)\, f(y)\, \mathrm{d}y \approx V_{i,j}\, f(x_j)$, where $V_{i,j} := \int_{B_r(x_i)} \psi_j(y)\, \mathrm{d}y$ is the *quadrature weight* of neighbour j at node i:

$$(\mathcal{K}^{\text{FEM-Conv}}a)(x_i) := \sum_{j \in N(i)} V_{i,j}\, \kappa\big(\Phi_{x_i}(x_j)\big)\, a(x_j) = \sum_{p=1}^{P^2} w_p \sum_{j \in N(i)} V_{i,j}\, \phi_p\big(\Phi_{x_i}(x_j)\big)\, a(x_j), \tag{12}$$

where $N(i) = \{j : x_j \in B_r(x_i)\}$ are the neighbours of node i: A multi-channel extension mirrors a standard CNN and is described in Appendix D.1. We refer to the weighted aggregation using the quadrature weights $V_{i,j}$ as *lumped aggregation*. The lumped aggregation is of particular importance for non-uniform node densities and ensures that the sum in (12) converges to the true integral in (11).

Remark 3.1. The construction extends to d-dimensional domains by replacing the Q1 basis with a d-fold multilinear basis on a $P^{\times d}$ reference grid, giving P^d filter weights and at most 2^d nonzero basis overlaps per neighbour. While this is tractable for the 2 or 3-dimensional domains, which are considered in recent applications (Yao et al., 2025; Huang et al., 2024; Ju et al., 2026), for $d \gg 3$ alternative methods have to be considered, e.g., based on sparse-grids (Bungartz & Griebel, 2004).

This parametrisation offers three advantages over MLP or spline kernels. First, it is a strict generalisation of standard convolutions, see Proposition C.1 for a formal statement. The architecture therefore inherits the inductive biases of CNNs that have proven effective in image-based diffusion

models. Second, since each basis function ϕ_p is compactly supported, any projected neighbour x_j has non-zero overlap with at most four basis functions via its bilinear stencil, see Figure 3. Each node in $B_r(x_i)$ thus contributes to exactly four filter weights, giving $\mathcal{O}(4\,|N(i)|)$ operations per node regardless of P^2, in contrast to MLP kernels which require a full forward pass per edge. Third, as the mesh is refined, the FEM-Conv (12) converges to the true integral, see Proposition C.2 for a formal statement with proof.

Multiscale mesh hierarchy. To enable multiscale processing (Property 1) while preserving domain flexibility (Property 3), we embed the FEM convolution within a hierarchical mesh architecture. This multiscale processing has been widely used in GNNs (Gao & Ji, 2019; Li et al., 2020; Würth et al., 2025; Bouziani & Boullé, 2024). For any domain we construct a hierarchy of L meshes $\{\mathcal{T}^{(\ell)}\}_{\ell=0}^{L-1}$, where $\mathcal{T}^{(0)}$ is the finest level, yielding progressively smaller node sets $\mathcal{Z}^{(\ell)}$ with $N_0 > \cdots > N_{L-1}$. Each mesh $\mathcal{T}^{(\ell)}$ induces a graph $\mathcal{G}^{(\ell)} = (\mathcal{Z}^{(\ell)}, \mathcal{E}^{(\ell)})$ with edges connecting adjacent nodes.

Coupling FEM convolutions to the hierarchy. Both the physical radius $r^{(\ell)}$ and the patch width $w^{(\ell)}$ at each level ℓ are hyperperameters. We set $r^{(\ell)} = \mu\,\tilde{d}^{(\ell)}$ and patch width $w^{(\ell)} = 2r^{(\ell)}$, where $\tilde{d}^{(\ell)}$ is the median edge length (distance between nodes) of $\mathcal{T}^{(\ell)}$ with $\mu > 0$. The same physical radius and patch width are then used for different meshes. The physical extent of the filter therefore grows with the coarseness of the mesh, mirroring the expansion of the receptive field of a U-Net. Transfers between mesh resolution (up- and downsampling) are implemented as learned restriction and prolongation maps. Noise (i.e., diffusion time) and positional context are reinjected at each level via FiLM modulation and learned per-level projections of the mesh coordinates. At the coarsest level, domain-conditioned cross attention bridges the N_{L-1} coarse-mesh tokens to a latent Diffusion Transformer (Peebles & Xie, 2023) with AdaLN-Zero time conditioning, which serves as the global latent memory of the model (Property 4). Full architectural details are provided in Appendix D.

Conditioning. We concatenate both the diffusion time (noise level) t, embedded via a Fourier embedding, and an optional condition vector $c \in \mathbb{R}^m$ (such as physical parameters for a family of PDEs) alongside the nodal coordinates x_i to each node.The diffusion time is additionally re-injected at every hierarchy level via FiLM (Perez et al., 2018), and at the coarsest level via AdaLN-Zero (Peebles & Xie, 2023) in the latent transformer.

4 Results

We evaluate the different properties: invariance to resolution and mesh discretisation (Section 4.2), generalisation across domain geometries (Section 4.3), and scaling to realistic and higher-dimensional data (Section 4.4). Beyond unconditional sampling, we evaluate conditional sampling for inverse problems. In the Appendix, Section F.2 isolates the contribution of each architectural component, whilst Section F.1 compares approximations of the guidance term via Fun-DPS (Yao et al., 2025) and Fun-DAPS (Lin et al., 2026) (see Section B.2 for further details).

4.1 Experimental Setup

Datasets We make use of the following datasets, which are described in more detail in Section E.1. The first is `GaussianBlobs`, see Figure 6 for some ground truth samples. The second is the `Pinball` dataset (Tomasetto et al., 2025). Samples of the `Pinball` dataset are shown in Figure 8. We also use the `CylinderFlow` dataset (Han et al., 2022) as used in Shi et al. (2025). Finally, we use `DeepJEB` (Hong et al., 2025), a 3D dataset, to show how our method scales.

Baselines We compare our GRIFDIR with diffusion models based on MP-GNN (Brandstetter et al., 2022), GAOT (Wen et al., 2025), FNO (Li et al., 2021), convolutional neural networks and MINO (Shi et al., 2025). These approaches are described in Appendix E.2.

Diffusion model setup In our experiments, we use the VE-SDE (4) using the specific parametrisation from Karras et al. (2022). We consider a noise schedule with $\sigma(0) = \sigma_{\text{min}} = 0.001$ and $\sigma_T = \sigma_{\text{max}} = 40.0$. To construct the covariance operator C, we follow Yao et al. (2025) and use radial basis function kernels with length-scale $\ell = 0.1$

$$K_\ell(x, y) \coloneqq \exp\left(-\frac{|x-y|^2}{2\ell^2}\right), \qquad (C_\ell f)(x) \coloneqq \int_\Omega K_\ell(x, y) f(y) dy. \tag{13}$$

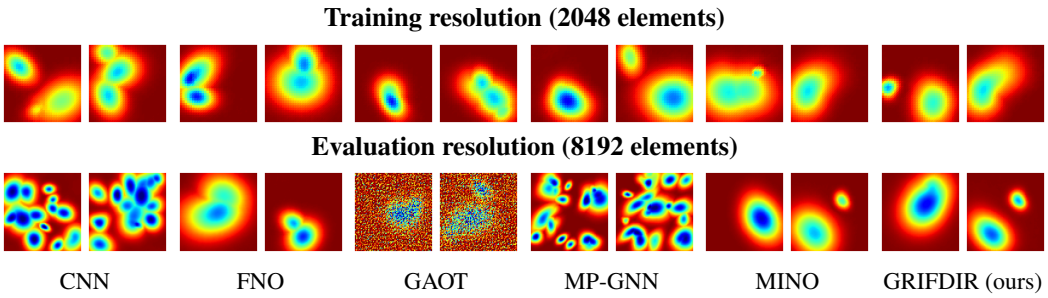


Figure 4: Resolution generalisation for sampling on `GaussianBlobs` sampling. Models are trained on a coarse triangular mesh with 2048 elements (top) and evaluated on a finer mesh with 8192 elements (bottom). Resolution-dependent models degrade or produce artifacts at higher resolution, while resolution-invariant operators maintain spatial structure and sample quality.

For unconditional sampling, we use the Heun sampler (Karras et al., 2022; Yao et al., 2025).

Metrics To evaluate the methods we mainly consider the maximum mean discrepancy (MMD) (Gretton et al., 2012) and the sliced Wasserstein distance (SWD) (Bonneel et al., 2015). We also compute two metrics for posterior sampling: the mean squared error (MSE) and the energy score (ES) between the ground truth and posterior estimates. Here, we consider conditional sampling for two linear inverse problems, a sparse point-wise sensor evaluation and a PDE-constrained operator governed by the Poisson equation, where we approximate the conditional score using Fun-DPS. For the MMD we use a Gaussian kernel with length-scale ℓ. All metrics are described in detail in Section E.3.

4.2 Resolution and Discretisation Invariance

Property 2 requires a single trained model to remain accurate both at unseen resolutions of a uniform mesh and under local changes in node density. We test these two notions separately.

Resolution invariance We train all baselines on a 2048-element mesh and evaluate, without retraining, on a coarser (1152-element) and a finer (8192-element) mesh of the same square domain for `GaussianBlobs`. We use a square domain with a uniform discretisation here specifically so that the CNN and FNO baselines can be included in the comparison. Table 1 reports MMD and SWD at all three resolutions, with samples shown in Figure 4. GRIFDIR achieves the best MMD and SWD at every resolution tested, including at the training resolution. The CNN and MP-GNN baselines fail in an interesting manner: when the number of mesh elements increases by a factor of four, the generated samples exhibit approximately four times as many blobs. While the performance of the FNO is quite stable across resolution, the baseline FNO can only be applied to rectangular domains with a uniform discretisation.

Table 1: MMD and SWD for `GaussianBlobs` at the training resolution (2048 points) and the inference resolution (8192 points or 1152 points) computed with 1024 samples. We report the mean MMD/SWD over the non-uniform meshes; full results are in Table 8. Best values are **bold** and second best underlined.

	1152 pts.		2048 pts. (train res.)		8192 pts.		Non-Uniform Mesh	
	MMD (↓)	SWD (↓)	MMD (↓)	SWD (↓)	MMD (↓)	SWD (↓)	MMD (↓)	SWD (↓)
CNN	<u>0.103</u>	<u>0.044</u>	0.126	0.044	0.501	0.208	-	-
FNO	0.119	0.045	0.117	0.042	0.149	0.052	-	-
GAOT	0.169	0.059	<u>0.094</u>	<u>0.034</u>	0.398	0.163	0.591	3.438
MP-GNN	0.106	0.049	0.132	0.046	0.451	0.190	0.195	0.068
MINO	0.133	0.049	0.109	0.041	<u>0.105</u>	<u>0.040</u>	**0.128**	**0.046**
GRIFDIR	**0.063**	**0.028**	**0.089**	**0.032**	**0.053**	**0.025**	<u>0.160</u>	<u>0.057</u>

Discretisation invariance Resolution invariance alone does not test robustness to *non-uniform* node density. To test density-dependence directly, we use the `GaussianBlobs` dataset on a graded

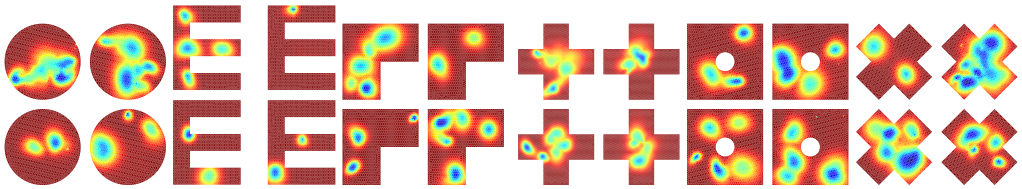


Figure 5: Samples of the multi domain GRIFDIR model for `GaussianBlobs`.

2D square mesh, with element size varying across the domain (fine left, fine centre, fine edge, fine right; see Figure 7 for visualisations of the different meshes). We evaluate the same baselines as in the resolution-invariance experiment above directly on these non-uniform meshes, without retraining. Table 1 reports the mean MMD and SWD across the four configurations. The individual results per mesh are in Table 8.

CNN and FNO cannot be evaluated in this setting at all, since both require inputs on a regular grid. Among the remaining baselines, GAOT degrades most sharply. MP-GNN fares moderately better. MINO and GRIFDIR are the only methods that remain broadly stable under non-uniform discretisation, with MINO achieving the best mean scores (0.128, 0.046) and GRIFDIR the second-best on both metrics (0.160, 0.057).

4.3 Domain and Geometry Generalisation

Next, we test whether a single GRIFDIR model can be trained to generate samples on eight differently shaped domains (Table 6), including non-convex, curved and domains with holes, to verify that the model copes with a range of irregularities in the domain. One set of model weights is trained jointly on a mixed dataset of all eight domains: the shared FEM-convolutional V-cycle acts as a mesh-agnostic encoder and decoder on each domain's own mesh, and at the coarsest level cross-attention heads translate the variable-size coarse mesh into a fixed set of latent tokens shared by every domain, with a one-hot domain embedding added to the attention queries so the same latent space is shared across the geometries (Appendix D). Figure 5 shows that this single model generates plausible samples on every domain.

4.4 Realistic and 3D Geometries

Pinball and Cylinder Flow Next, we consider 2D datasets with non-standard meshes, namely the `CylinderFlow` and `Pinball` dataset, both of which contain holes in the domain. Results for both datasets are in Table 2. We see across both datasets GRIFDIR has competitive results, with the best or second best results across all metrics. Although MP-GNN does well on `CylinderFlow`, we saw in Figure 4 it doesn't generalise well to increases in resolution. MINO performs well on `Pinball`, but falls behind on `CylinderFlow`. We hypothesise that this is to do with the meshes: `Pinball` has very uniform meshes (see Figure 9), whereas `CylinderFlow` does not (see Figure 15), and MINO only includes a mean aggregation over nodes in the neighbourhood.

Table 2: Results for `CylinderFlow` and `Pinball`. MMD and SWD are based on 10 000 samples for `CylinderFlow` and 1024 samples for `Pinball`. For `CylinderFlow`, DPS-MSE and DPS-ES use 42 sensors with farthest-point placement; for `Pinball`, 30 sensors with random (but fixed) placement. Best results are shown in **bold** and second-best results are underlined.

`CylinderFlow`

Method	MMD (↓)	SWD (↓)	DPS-MSE (↓)	DPS-ES (↓)
MP-GNN	**0.0187**	**0.0115**	1.71×10^{-4}	0.4246
MINO	0.0307	0.0165	$\mathbf{1.11 \times 10^{-4}}$	0.4427
GRIFDIR	0.0197	0.0117	1.56×10^{-4}	**0.3788**

`Pinball`

Method	MMD (↓)	SWD (↓)	DPS-MSE (↓)	DPS-ES (↓)
Spline-Conv	0.3627	0.4416	0.1738	33.1317
Geo-FNO	0.1686	0.2256	0.1783	23.3092
MLP-Conv	0.2779	0.3097	0.0778	17.1494
MINO	**0.0839**	**0.0920**	0.0927	16.8855
GRIFDIR	0.0907	0.0954	**0.0721**	**14.2480**

Scaling to 3D Finally, we show that GRIFDIR scales to 3D meshes. For this, we generalise the FEM kernel to 3D via a trilinear interpolation of the kernel and evaluate on the `DeepJEB` dataset,

Table 3: `DeepJEB`: the 6000 and 24 000 models evaluated at $\pm 33\%$ of their training density. Mean $\pm$ std over sampling seeds; each seed is averaged over the held-out test brackets.

Train pts.	Eval pts.	rel-$L^2 \downarrow$	w-rel-$L^2 \downarrow$	IoU@1% $\uparrow$	IoU@5% $\uparrow$	IoU@10% $\uparrow$	$r \uparrow$
6000	4000	$0.698_{\pm 0.002}$	$0.648_{\pm 0.002}$	$0.416_{\pm 0.007}$	$0.386_{\pm 0.002}$	$0.446_{\pm 0.002}$	$0.689_{\pm 0.002}$
6000	6000	$0.645_{\pm 0.004}$	$0.575_{\pm 0.004}$	$0.523_{\pm 0.007}$	$0.479_{\pm 0.004}$	$0.545_{\pm 0.002}$	$0.764_{\pm 0.002}$
6000	8000	$0.695_{\pm 0.003}$	$0.595_{\pm 0.004}$	$0.538_{\pm 0.006}$	$0.490_{\pm 0.004}$	$0.554_{\pm 0.002}$	$0.779_{\pm 0.002}$
24 000	16 000	$0.734_{\pm 0.002}$	$0.658_{\pm 0.003}$	$0.404_{\pm 0.003}$	$0.381_{\pm 0.005}$	$0.439_{\pm 0.002}$	$0.666_{\pm 0.002}$
24 000	24 000	$0.638_{\pm 0.002}$	$0.564_{\pm 0.003}$	$0.510_{\pm 0.005}$	$0.483_{\pm 0.005}$	$0.541_{\pm 0.001}$	$0.771_{\pm 0.002}$
24 000	32 000	$0.654_{\pm 0.003}$	$0.561_{\pm 0.004}$	$0.517_{\pm 0.004}$	$0.493_{\pm 0.004}$	$0.552_{\pm 0.002}$	$0.785_{\pm 0.003}$
6000 (lumped)	6000	$0.666_{\pm 0.004}$	$0.593_{\pm 0.004}$	$0.512_{\pm 0.004}$	$0.446_{\pm 0.005}$	$0.519_{\pm 0.002}$	$0.750_{\pm 0.002}$
24 000 (lumped)	24 000	$0.673_{\pm 0.002}$	$0.591_{\pm 0.002}$	$0.492_{\pm 0.004}$	$0.430_{\pm 0.002}$	$0.499_{\pm 0.002}$	$0.745_{\pm 0.000}$
24 000 (lumped)	32 000	$0.730_{\pm 0.007}$	$0.614_{\pm 0.007}$	$0.496_{\pm 0.005}$	$0.431_{\pm 0.002}$	$0.498_{\pm 0.002}$	$0.749_{\pm 0.002}$

see Appendix F.4. We train GRIFDIR to generate stress fields across different bracket geometries (see Figure 1 for examples). The model is conditioned on the point cloud of each bracket.

We train separate models at 6000 and 24 000 points and evaluate each at its training resolution and at $\pm 33\%$ of it. Unlike `GaussianBlobs` and `Pinball`, where every sample shares a common domain and MMD/SWD compare two populations of draws from one distribution, each `DeepJEB` bracket defines its own geometry and has a single ground-truth field rather than a distribution. We therefore evaluate goodness of fit against that single reference instance, mirroring the per-instance metrics used for conditional sampling in Section 2.1 rather than the distributional metrics. In particular, we compute the relative L^2 error and a stress-weighted variant (w-rel-L^2), the Pearson correlation r between predicted and true stress magnitude (evaluating whether the spatial *pattern* of stress is correct), and the intersection-over-union of the predicted and true top-$q\%$ stress regions (IoU@q, against a chance baseline of roughly $q/2$). Training and sampling time across resolutions are in Section F.4, additional samples and ground truth stress fields are shown in Figure 19.

The results are summarised in Table 3. At both training resolutions, GRIFDIR places roughly half of the true top-1% stress correctly (IoU@$1\% \approx 0.52$) with $r \approx 0.77$ indicating that the overall pattern is captured well. Evaluating at a *finer* resolution than training is mildly better on IoU and r (e.g. at 6000 nodes: IoU@1%: $0.523 \to 0.538$ and r: $0.764 \to 0.779$), whereas evaluating at a *coarser* resolution degrades performance, e.g., 9-14% for r. The table also includes a lumped-quadrature variant (Section 3.3) at matched evaluation resolutions. Unlike the graded 2D meshes in Table 8, lumping consistently underperforms the default aggregation here on every metric, as the `DeepJEB` point clouds are fairly uniform. For a single sample, GRIFDIR achieves a relative L^2 error of 0.64 at both training resolutions.

Comparison to supervised baselines For context, GINOT (Liu et al., 2025) reports 0.356, GI-FNO 0.314, and GNO 0.673 relative error on the same JEB benchmark. However, these models are trained in a supervised way to directly minimise the relative L^2 error.

5 Conclusion

In this work, we introduced GRIFDIR, a multi-scale graph neural operator making use of FEM convolution, designed for generative modelling over irregular geometries. By construction, GRIFDIR is discretisation invariant and supports sampling at arbitrary resolution, making it well-suited as a backbone for score-based diffusion models defined in infinite-dimensional function spaces. We demonstrated the effectiveness of our approach across four datasets spanning a variety of domains, e.g., including squared, cross-shaped, and L-shaped geometries, consistently producing high-quality samples without retraining across resolutions.

Several directions remain for future work. On the conditional generation side, we plan to extend our framework to non-linear forward operators and to benchmark against learned guidance approaches such as conditional training (Baldassari et al., 2023) and supervised guidance methods like SGT (Baker et al., 2026). Further, ablation regarding the underlying infinite-dimensional framework, such as the choice of covariance C or SDE, would be valuable.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs (proof-checking) and to implement methods (coding some of the model components and several plotting utilities). We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, clean and reformat datasets, or support qualitative and thematic data analysis or the interpretation of results; assisting with translation is not applicable to this work.

Additionally, we used generative AI tools for brainstorming and for creating or modifying scientific figures or images, including adapting plotting functions in matplotlib and writing TikZ figure code. We have reviewed all AI-assisted work. LLM-generated code was verified by three authors, and all AI-assisted figures were reviewed and evaluated by the authors before inclusion. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

REFERENCES

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*, 2019.

Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024.

Elizabeth Louise Baker, Alexander Denker, and Jes Frellsen. Supervised guidance training for infinite-dimensional diffusion models. In *Forty-third International Conference on Machine Learning*, 2026.

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

Francesca Bartolucci, Emmanuel De Bézenac, Bogdan Raonic, Roberto Molinaro, Siddhartha Mishra, and Rima Alaifari. Representation equivalent neural operators: a framework for alias-free operator learning. *Advances in Neural Information Processing Systems*, 36:69661–69672, 2023.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

Nacime Bouziani and Nicolas Boullé. Structure-preserving operator learning, 2024. URL `https://arxiv.org/abs/2410.01065`.

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

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

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023.

Masoumeh Dashti and Andrew M. Stuart. *The Bayesian Approach to Inverse Problems*, pp. 311–428. Springer International Publishing, Cham, 2017. ISBN 978-3-319-12385-1. doi: 10.1007/978-3-319-12385-1_7.

Alexander Denker, Francisco Vargas, Shreyas Padhy, Kieran Didi, Simon Mathis, Riccardo Barbano, Vincent Dutordoir, Emile Mathieu, Urszula Julia Komorowska, and Pietro Lio. DEFT: Efficient fine-tuning of diffusion models by learning the generalised h-transform. *Advances in Neural Information Processing Systems*, 37:19636–19682, 2024.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Kevin Doherty, Cooper Simpson, Stephen Becker, and Alireza Doostan. Quadconv: Quadrature-based convolutions with applications to non-uniform PDE data compression. *J. Comput. Phys.*, 498: 112636, 2024. doi: 10.1016/J.JCP.2023.112636.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 869–877, 2018.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. POT: Python Optimal Transport. *Journal of Machine Learning Research*, 22(78): 1–8, 2021. URL `http://jmlr.org/papers/v22/20-451.html`.

Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *International Conference on Machine Learning*, pp. 2083–2092. PMLR, 2019.

Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

Paul Hagemann, Sophie Mildenberger, Lars Ruthotto, Gabriele Steidl, and Nicole Tianjiao Yang. Multilevel diffusion: Infinite dimensional score-based diffusion models for image generation. *SIAM Journal on Mathematics of Data Science*, 7(3):1337–1366, 2025.

Xu Han, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Liping Liu. Predicting physics in mesh-reduced space with temporal attention. In *International Conference on Learning Representations*, 2022.

Seongjun Hong, Young-Min Kwon, Dongwon Shin, Jinsung Park, and Namwoo Kang. DeepJEB: 3d deep learning-based synthetic jet engine bracket dataset. *Journal of Mechanical Design*, 147(4): 041703, 2025.

Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. DiffusionPDE: Generative PDE-solving under partial observation. *Advances in Neural Information Processing Systems*, 37: 130291–130323, 2024.

Xin Ju, Jiachen Yao, Anima Anandkumar, Sally M Benson, and Gege Wen. Function-space decoupled diffusion for forward and inverse modeling in carbon capture and storage. *arXiv preprint arXiv:2602.12274*, 2026.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Gavin Kerrigan, Giosue Migliorini, and Padhraic Smyth. Functional flow matching. In *The 27th International Conference on AI and Statistics (AISTATS)*, 2024.

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.

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.

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.

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.

Jae Hyun Lim, Nikola B Kovachki, Ricardo Baptista, Christopher Beckham, Kamyar Azizzadenesheli, Jean Kossaifi, Vikram Voleti, Jiaming Song, Karsten Kreis, Jan Kautz, et al. Score-based diffusion models in function space. *Journal of Machine Learning Research*, 26(158):1–62, 2025.

Thomas YL Lin, Jiachen Yao, Lufang Chiang, Julius Berner, and Anima Anandkumar. Decoupled diffusion sampling for inverse problems on function spaces. *arXiv preprint arXiv:2601.23280*, 2026.

Levi Lingsch, Mike Y Michelis, Emmanuel De Bézenac, Sirani M Perera, Robert K Katzschmann, and Siddhartha Mishra. Beyond regular grids: Fourier-based neural operators on arbitrary domains. *arXiv preprint arXiv:2305.19663*, 2023.

Qibang Liu, Vincient Zhong, Hadi Meidani, Diab W. Abueidda, Seid Koric, and Philippe H. Geubelle. Geometry-informed neural operator transformer. *CoRR*, abs/2504.19452, 2025. doi: 10.48550/ARXIV.2504.19452. URL https://doi.org/10.48550/arXiv.2504.19452.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5425–5434. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.576.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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

Jakiw Pidstrigach, Youssef Marzouk, Sebastian Reich, and Sven Wang. Infinite-dimensional diffusion models. *Journal of Machine Learning Research*, 25(414):1–52, 2024.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel De Bézenac. Convolutional neural operators for robust and accurate learning of PDEs. *Advances in Neural Information Processing Systems*, 36: 77187–77200, 2023.

L Chris G Rogers and David Williams. *Diffusions, Markov processes, and martingales*, volume 2. Cambridge university press, 2000.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Yaozhong Shi, Zachary E Ross, Domniki Asimaki, and Kamyar Azizzadenesheli. Mesh-informed neural operator : A transformer generative approach. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 29–38. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.11.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Matteo Tomasetto, Jan P Williams, Francesco Braghin, Andrea Manzoni, and J Nathan Kutz. Reduced order modeling with shallow recurrent decoder networks. *Nature Communications*, 16(1):10260, 2025.

Mario Lino Valencia, Tobias Pfaff, and Nils Thuerey. Learning distributions of complex fluid simulations with diffusion graph networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Shizheng Wen, Arsh Kumbhat, Levi Lingsch, Sepehr Mousavi, Yizhou Zhao, Praveen Chandrashekar, and Siddhartha Mishra. Geometry Aware Operator Transformer as an efficient and accurate neural surrogate for PDEs on arbitrary domains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

Tobias Würth, Niklas Freymuth, Gerhard Neumann, and Luise Kärger. Diffusion-based hierarchical graph neural networks for simulating nonlinear solid mechanics. *arXiv preprint arXiv:2506.06045*, 2025.

Jiachen Yao, Abbas Mammadov, Julius Berner, Gavin Kerrigan, Jong Chul Ye, Kamyar Azizzadenesheli, and Anima Anandkumar. Guided Diffusion Sampling on Function Spaces with Applications to PDEs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

Bingliang Zhang, Wenda Chu, Julius Berner, Chenlin Meng, Anima Anandkumar, and Yang Song. Improving diffusion inverse problem solving with decoupled noise annealing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 20895–20905, 2025.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268, 2021.

A Related Work

Graph Diffusion It is important to distinguish the infinite-dimensional framework from recent works that apply GNNs within standard finite-dimensional diffusion models. For instance, Valencia et al. (2025) propose *Graph Diffusion Models* to learn the conditional distribution of a fluid's equilibrium state given a short simulation rollout. While this approach uses GNNs for diffusion, it operates within the finite-dimensional framework on discrete states corrupted by i.i.d. Gaussian noise. By contrast, our infinite-dimensional formulation ensures the diffusion model operates in a resolution-invariant manner by design.

Functional Flow Matching Functional Flow Matching (FFM) (Kerrigan et al., 2024) is a related framework to infinite-dimensional diffusion. FFM generalises flow matching to operate in infinite-dimensional function spaces. Rather than learning a vector field on a finite-dimensional latent, FFM defines a path of probability measures interpolating between a Gaussian measure (often $\mathcal{N}(0, C)$) and the data distribution, and learns a function-space vector field that generates this path. As in our setting, extending such generative models to function spaces requires discretisation-invariant network architectures capable of operating on irregular domains. This need is addressed by MINO (Shi et al., 2025), which makes use of graph neural operators and cross-attention mechanisms. Our work shares this motivation: the infinite-dimensional diffusion model we develop similarly requires mesh-aware architectures to handle unstructured spatial domains, and the resulting model is likewise discretisation-invariant by design.

B Background

B.1 The story in finite-dimensions

One way of using diffusion models for function space data, is first by discretising the data, and then applying the finite-dimensional formulation. For this, suppose we have a data distribution p_{data} with support in state space $\mathbb{R}^d$. In the case of function space data, p_{data} could, for instance, be a distribution of 2D magnetic fields discretised on a uniform grid.

To learn to generate samples from p_{data} we follow the score-based diffusion model setup (Song et al., 2021). We consider a forward stochastic differential equation (SDE) that noises data samples, such that for large T the marginal distribution is approximately Gaussian:

$$\mathrm{d}x_t = f(x_t, t)\mathrm{d}t + g(t)\mathrm{d}W_t, \quad x_0 \sim p_{\text{data}}(x), \tag{14}$$

where $f : \mathbf{R}^d \times [0, T] \to \mathbf{R}^d$ and $g : [0, T] \to \mathbf{R}_{>0}$ are the drift and diffusion coefficients, chosen such that $p_T \approx \mathcal{N}(0, I)$. By simulating trajectories of the time-reversal of this SDE, run backwards in time from $x_T \sim p_T \approx \mathcal{N}(0, I)$, we obtain samples from p_{data} at time $t = 0$:

$$\begin{aligned} \mathrm{d}x_t &= \hat{f}(x_t, t)\mathrm{d}t + g(t)\mathrm{d}W_t, \\ \hat{f}(x, t) &\coloneqq [f(x, t) - g(t)^2 \nabla_x \log p_t(x)]. \end{aligned} \tag{15}$$

The reverse SDE depends on the unknown score function which we approximate with a neural network $s_\theta(x, t) \approx \nabla_x \log p_t(x)$ trained via denoising score matching.

Now consider an inverse problem of the form

$$y = Ax + \eta, \quad x \sim p_{\text{data}}(x), \tag{16}$$

where A is a (possibly non-linear) forward operator and η is Gaussian observation noise. Then, instead of sampling from the prior distribution p_{data}, we wish to sample from the posterior, $p_{\text{data}}(\cdot \mid y)$. This can be achieved by replacing the reverse SDE with a conditional reverse SDE:

$$\begin{aligned} \mathrm{d}x_t &= \tilde{f}(x_t, t)\mathrm{d}t + g(t)\mathrm{d}W_t, \\ \tilde{f}(x_t, t) &\coloneqq [f(x, t) - g(t)^2 \nabla_x \log p_t(x \mid y)] \end{aligned} \tag{17}$$

which relies on the conditional score $\nabla_x \log p_t(x \mid y)$. Using Bayes' theorem we can decompose the conditional score as

$$\nabla_x \log p_t(x \mid y) = \nabla_x \log p_t(x) + \nabla_x \log p_t(y \mid x), \tag{18}$$

where the first term is approximated by the learned unconditional score and $\nabla_x \log p_t(y \mid x)$ is the guidance term, a noisy version of the data likelihood $p_{\text{lkhd}}(y \mid x_0)$. The guidance term is intractable as can be seen by writing the guidance term as a conditional expectation

$$p_t(y \mid x_t) = \int p_{\text{lkhd}}(y \mid x_0) p(x_0 \mid x_t) dx_0 \tag{19}$$

$$= \mathbb{E}[p_{\text{lkhd}}(y \mid x_0) \mid x_t]. \tag{20}$$

Therefore, to sample from the posterior, we either need to approximate this guidance term, or a way to learn the conditional score directly.

B.2 Guidance for Conditional Sampling

In order to sample from the conditional distribution π^y, the fundamental step is to formulate or approximate a conditional score $s^y(t, a)$. We detail several paradigms for achieving this below.

Conditional Training For the conditional sampling problem, outlined in Section 2.1, there are multiple ways to sample from π^y. One of such is learning the conditional score

$$s^y(t, a) = -v_t^{-1} \left(a - m_t \mathbb{E}[A_0 \mid A_t = a, Y = y]\right), \tag{21}$$

, where v_t and m_t denote the variance and mean coefficients of the forward transition, i.e. $\text{Law}(A_t \mid A_0) = \mathcal{N}(m_t A_0, v_t C)$, as outlined by Baldassari et al. (2023). Using the time-reversal with the unconditional score as defined in (2) leads to samples from the distribution π. However, Baldassari et al. (2023) show that replacing the unconditional score by the conditional one in (21), the time-reversal leads to samples from π^y. The conditional score is learned directly (as opposed to relying on knowledge of the unconditional score), using the loss

$$\mathbb{E}_{(a_0,y)\sim\text{Law}(A_0,Y) a_t \sim \text{Law}(A_t \mid A_0 = a_0)} \| - (1 - e^{-t})^{-1} (a_t - e^{-\frac{t}{2}} a_0 - s_\theta(t, a_t, y)\|_{\mathcal{H}}^2, \tag{22}$$

where $\mathcal{H}$ is the Hilbert space to which the data a belongs. Baldassari et al. (2023) show that if the conditional expectation of the score function is bounded, then the above loss is equivalent to the intractable loss function

$$\mathbb{E}_{(a_t,y)\sim\text{Law}(A_t,Y)} \| s^y(t, a_t) - s_\theta(t, a_t, y)\|_{\mathcal{H}}^2. \tag{23}$$

Joint Embedding Models Rather than explicitly computing the guidance term (9), this approach circumvents it by learning a joint score. Using paired data (a_0, y), an unconditional diffusion model is trained to estimate $\pi_{\text{joint}}(a_0, y)$. During training, the observed data y is treated as a fixed mask. By marginalising over this joint measure conditioned on this mask, one samples from the posterior π^y (Yao et al., 2025; Huang et al., 2024). This approach requires training a new joint model for every new forward operator $\mathcal{L}$.

Diffusion Priors and Guidance: (Fun-)DPS Another approach relies on training an unconditional diffusion prior solely on the measure π of the parameter a_0, and then approximating the intractable guidance term (9). For instance, Lin et al. (2026) utilise a training-free approximation. At each denoising step, they use the unconditional score to compute the Tweedie estimate $\hat{a}_t(a) = \mathbb{E}[A_0 \mid A_t = a]$ and approximate the guidance via

$$\log h^u(t, a) \approx -\Phi(\hat{a}_t(a), u). \tag{24}$$

This estimate is then used to correct the unconditional trajectory toward the conditional distribution. This mechanism is closely related to posterior samplers like DPS (Chung et al., 2023) in finite dimensions and FunDPS (Yao et al., 2025) in infinite dimensions. In its full form, FunDPS combines a joint embedding model with this Tweedie-based approximation, though in our setting we can replace the learned joint model with a standalone diffusion prior and solely adopt the approximated guidance term (24).

DAPS Decoupled Annealing Posterior sampling (DAPS) (Zhang et al., 2025; Lin et al., 2026) is a posterior sampling method that decouples the denoising step from the data-consistency update. In contrast to (Fun-)DPS, DAPS avoids backpropagation through the score model and instead performs

Algorithm 1 Function-Space Decoupled Annealing Posterior Sampling (Fun-DAPS)

Require: Score operator s_θ, measurements y, covariance operator C, noise schedule σ_t, $\{t_i\}_{i=0}^{N_A}$, likelihood potential $\Phi(\mathbf{x})$, $r_t = \sigma_t$, step size η_t

Sample $a_T \sim \mathcal{N}(0, \sigma_T^2 C)$
for $i = N_A, N_A - 1, \ldots, 1$ **do**
  Tweedie Estimate (or e.g. probability flow)
  $\hat{a}_0^{(0)} = a_{t_i} + \sigma_{t_i}^2 C\, s_\theta(a_{t_i}, t_i)$
  for $j = 0, \ldots, N-1$ **do**
    Function-space (preconditioned) Langevin dynamics

$$\hat{a}_0^{(j+1)} \leftarrow \hat{a}_0^{(j)} - \eta_t C \left[\frac{1}{r_{t_i}^2} C^{-1} \left(\hat{a}_0^{(j)} - \hat{a}_0^{(0)} \right) + \nabla \Phi(\hat{a}_0^{(j)}) \right] + \sqrt{2\eta_t}\, \xi_j, \quad \xi_j \sim \mathcal{N}(0, C)$$

  end for
  Annealed Gaussian update

$$a_{t_{i-1}} \sim \mathcal{N}\left(\hat{a}_0^{(N)}, \sigma_{t_{i-1}}^2 C \right)$$

end for
Return: a_0

Langevin update steps in the denoised space. We adapt DAPS to the function space setting, leading to the Function-Space DAPS (Fun-DAPS) algorithm, shown in Algorithm 1. Similar to the finite-dimensional setting, the conditional distribution $A_0|A_t = a_t$ is generally intractable. Following Zhang et al. (2025), we use a Gaussian approximation

$$A_0|A_t = a_t \sim \mathcal{N}(\hat{a}_0(a_t), r_t^2 C), \tag{25}$$

where C is the prior covariance operator and $\hat{a}_0(a_t)$ denotes a Tweedie estimate computed from the score model. Assume measurements are generated according to

$$y = \mathcal{L}(a) + \xi, \qquad \xi \sim \mathcal{N}(0, \sigma_\xi^2 I), \tag{26}$$

with $\mathcal{L} : \mathcal{H} \to \mathbb{R}^m$ (e.g., finite-dimensional observations), which results in a likelihood potential $\Phi(a) = \frac{1}{2\sigma_\xi^2} \|\mathcal{L}(a) - y\|^2$. The function-space (preconditioned) Langevin dynamics then take the form

$$\hat{a}_0^{(j+1)} \leftarrow \hat{a}_0^{(j)} - \eta_t C \left[\frac{1}{r_t^2} C^{-1} \left(\hat{a}_0^{(j)} - \hat{a}_0^{(0)} \right) + \nabla \Phi(\hat{a}_0^{(j)}) \right] + \sqrt{2\eta_t}\, \epsilon_j, \quad \epsilon_j \sim \mathcal{N}(0, C), \tag{27}$$

with a decaying step size η_t.

Trained Guidance (DEFT/SGT) In finite dimensions Denker et al. (2024) propose a training method, DEFT, to learn the guidance term $\nabla_a \log p_t(y \mid a)$. This is extended into infinite-dimensions by Baker et al. (2026) who introduce SGT to learn the infinite-dimensional guidance term $\nabla_a \log h^y(t, a) \coloneqq \nabla \log \xi^{-1} \mathbb{E}_{\mathbb{P}}[\exp(-\Phi(A_0, y)) \mid A_t = a]$ as defined in (9). Here, we detail the infinite-dimensional counterpart SGT, though the finite-dimensional DEFT works similarly. In order to learn the guidance term, Baker et al. (2026) work with the variance preserving SDE (2) and define the loss function

$$L(\phi) \coloneqq \mathbb{E}\left[\left\| \frac{A_t - e^{-\frac{t}{2}} A_0}{1 - e^{-t}} + s(t, A_t) + u_\phi(t, A_t, Y) \right\|_K^2 \right], \tag{28}$$

where the expectation is taken over $t \sim \mathcal{U}[0, T]$, $A_0, Y \sim \text{Law}(A_0, Y)$, $A_t \sim \text{Law}(A_t \mid A_0)$, where K is an appropriately chosen Hilbert space that depends on the regularity of the data distribution. Then for the minimiser ϕ^* of $L(\phi)$, it holds that $u_{\phi^*} = C \nabla \log h^y(t, a)$.

C PROOFS

Proposition C.1 (Generalisation of CNNs). *Let $\Omega \subset \mathbb{R}^2$. Let the mesh be the uniform Cartesian grid $\{x_i + h(m,n) : m,n \in \mathbb{Z}\}$ with the bilinear basis, so $\operatorname{supp}\psi_j = \{y : \|y - x_j\|_\infty \le h\}$. Let P be odd and set the FEM patch width to $r = \frac{(P-1)h}{2}$. Then, we have*

$$(\mathcal{K}^{\text{FEM-Conv}}a)(x_i) = h^2 \sum_{m,n=-(P-1)/2}^{(P-1)/2} w_{p(m,n)}\, a\big(x_i + h(m,n)\big),$$

i.e. $\mathcal{K}^{\text{FEM-Conv}}$ reduces exactly *to a standard discrete $P \times P$ cross-correlation with kernel $\{w_p\}_{p=1}^{P^2}$, up to the constant rescaling h^2.*

Proof. For the uniform grid the neighbours are given by $\mathcal{N}(i) := \{x_i + h(m,n) : m,n \in \{-\frac{P-1}{2}, \dots, \frac{P-1}{2}\}\}$. For every neighbour $x_j \in \mathcal{N}(i)$ we have that

$$\Phi_{x_i}(x_j) = \frac{h(m,n)}{r} = \frac{2(m,n)}{P-1} =: \xi_{p(m,n)}$$

is exactly a reference grid point, i.e., $(m,n) \mapsto \xi_{p(m,n)}$ a bijection between $\mathcal{N}(i)$ and the P^2 points $\{\xi_p\}$ of the reference patch. Conversely, if $x_j \notin \mathcal{N}(i)$, then $\|x_j - x_i\|_\infty > r$, so $x_j \notin B_r(x_i)$. Then for every $x_j \in \mathcal{N}(i)$, and using $\phi_p(\xi_q) = \delta_{pq}$, we have

$$\kappa\big(\Phi_{x_i}(x_j)\big) = \sum_{p=1}^{P^2} w_p\, \phi_p\big(\xi_{p(m,n)}\big) = w_{p(m,n)}.$$

For the uniform cartesian grid we have $V_{i,j} = h^2$ on $\mathcal{N}(i)$ and as such

$$(\mathcal{K}^{\text{FEM-Conv}}a)(x_i) = \sum_{x_j \in \mathcal{N}(i)} h^2\, w_{p(m,n)}\, a(x_j) = h^2 \sum_{m,n=-(P-1)/2}^{(P-1)/2} w_{p(m,n)}\, a\big(x_i + h(m,n)\big),$$

which is exactly a discrete $P \times P$ convolution with kernel $\{w_p\}$, up to the constant h^2. □

Proposition C.2 (Mesh-refinement consistency). *Let Ω be compact. For a fixed kernel $\kappa(\xi) = \sum_{p=1}^{P^2} w_p\, \phi_p(\xi)$ and $a \in C(B_r(x_i))$ we have*

$$(\mathcal{K}^{\text{FEM-Conv}}a)(x_i) \;\longrightarrow\; (\mathcal{K}a)(x_i) = \int_{B_r(x_i)} \kappa\big(\Phi_{x_i}(y)\big)\, a(y)\, \mathrm{d}y \tag{29}$$

as $h_n \to 0$.

Proof. Let $f(y) := \kappa(\Phi_{x_i}(y))\, a(y)$. Since κ is a finite sum of continuous basis functions, it is continuous on $\hat{\Omega}$, Φ_{x_i} is continuous, and a is continuous by assumption. So, f is continuous. Further, f is uniformly continuous as $B_r(x_i)$ is compact.

Using $\sum_{j \in N(i)} \psi_j \equiv 1$ on $B_r(x_i)$ we can rewrite the integral

$$\int_{B_r(x_i)} f(y)\, dy = \sum_{j \in N(i)} \int_{B_r(x_i)} \psi_j(y)\, f(y)\, dy.$$

With this we can bound

$$\begin{aligned}
\Big|(\mathcal{K}^{\text{FEM-Conv}}a)(x_i) - (\mathcal{K}a)(x_i)\Big| &= \Big|\sum_{j \in N(i)} V_{i,j}\, f(x_j) - \int_{B_r(x_i)} f(y)\, dy\Big| \\
&= \Big|\sum_{j \in N(i)} \int_{B_r(x_i)} \psi_j(y)\big(f(x_j) - f(y)\big)\, dy\Big| \\
&\le \sum_{j \in N(i)} \int_{B_r(x_i)} \psi_j(y)\,\big|f(x_j) - f(y)\big|\, dy
\end{aligned}$$

Fix $\varepsilon > 0$. By uniform continuity of f, choose $\delta > 0$ such that $|y - y'| < \delta \Rightarrow |f(y) - f(y')| < \varepsilon$. If $h_n < \delta$, then every $y \in \operatorname{supp} \psi_j$ satisfies $|y - x_j| \le h_n < \delta$ for $j \in N(i)$. We then get

$$\Big|(\mathcal{K}^{\text{FEM-Conv}} a)(x_i) - (\mathcal{K}a)(x_i)\Big| \le \varepsilon \sum_{j \in N(i)} \int_{B_r(x_i)} \psi_j(y)\, dy = \varepsilon\, |B_r(x_i)|,$$

using the identity $\sum_{j \in N(i)} V_{i,j} = |B_r(x_i)|$ from before. As $\varepsilon > 0$ was arbitrary, $(\mathcal{K}^{\text{FEM-Conv}} a)(x_i) \to (\mathcal{K}a)(x_i)$ as $h_n \to 0$. □

D GRIFDIR ARCHITECTURE DETAILS

We parametrise the score network $s_\theta(t, a)$ as a multi-resolution graph U-Net whose intra-level operator is the FEM-basis convolution of Equation (12). We describe the forward pass; encoder, V-cycle across L mesh resolutions, global latent transformer at the coarsest level, and decoder and embedded supporting components (FiLM timestep conditioning, learned restriction/prolongation, position re-injection, multi-domain heads).

Mesh hierarchy. Meshes are built once per domain and cached. For square conductivity (32×32 DG-0 cells) we use a uniform refinement chain $32 \to 16 \to 8 \to 4$ and store sparse pool/unpool matrices per level. For unstructured domains (E.1.1-L-shape, circle, plus and E.1.2 Pinball), Gmsh produces independent triangulations at decreasing $h_{\max}$ and the pool/unpool maps are built by k-NN matching of node positions.

Per-node input encoding. Each node i of the finest mesh carries a noisy observation $\mathbf{a}_{t,i} \in \mathbb{R}^C$ and a physical coordinate $x_i \in \mathbb{R}^d$. The diffusion timestep $t \in [0, T]$ is mapped to a d_t-dimensional random Fourier-feature embedding $\gamma(t) = [\sin(2\pi\boldsymbol{\omega}t); \cos(2\pi\boldsymbol{\omega}t)]$ with $\boldsymbol{\omega} \sim \mathcal{N}(0, \sigma^2 I)$. A two-layer MLP with SiLU activations lifts the per-node vector $[\mathbf{a}_{t,i};\, x_i;\, \gamma(t)]$ to the hidden dimension H, producing the level-0 representation $h^{(0)} \in \mathbb{R}^{N_0 \times H}$. The same embedding $\gamma(t)$ is reused later as the conditioning input to per-level FiLM modulation in the V-cycle and to AdaLN-Zero conditioning in the latent transformer.

Multi-resolution V-cycle. At each level $\ell \in \{0, \dots, L-1\}$, K FEM-convolutional layers (Section 3.3 update node features in place. Between levels, learned restriction $\mathcal{R}^{(\ell)}$ and prolongation $\mathcal{P}^{(\ell)}$ operators move features fine-to-coarse and coarse-to-fine. Restriction averages each coarse node's pre-image and applies a small MLP,

$$h_c^{(\ell+1)} = \text{MLP}\Big(\tfrac{1}{|\mathcal{R}^{-1}(c)|} \sum_{i \in \mathcal{R}^{-1}(c)} h_i^{(\ell)}\Big).$$

Prolongation broadcasts each coarse feature back to its fine pre-image and combines it with the down-pass skip through a residual MLP,

$$h_i^{(\ell)} = h_{\mathcal{P}(i)}^{(\ell+1)} + \text{MLP}\big([h_{\mathcal{P}(i)}^{(\ell+1)};\, h_{\text{skip},i}^{(\ell)}]\big).$$

After every FEM-convolutional layer we apply zero-initialised FiLM modulation (Perez et al., 2018), $h \leftarrow h \odot (1 + \boldsymbol{\alpha}^{(\ell)}) + \boldsymbol{\beta}^{(\ell)}$ with $(\boldsymbol{\alpha}^{(\ell)}, \boldsymbol{\beta}^{(\ell)}) = \text{MLP}_\ell(\gamma(t))$, so the diffusion timestep modulates each level's features and the network starts from an identity mapping. To keep the FEM filter coordinate-aware after each restriction we additionally re-inject the level's physical coordinates via a per-level linear projection added to the hidden features.

Cross-attention heads to the latent space. Between the V-cycle bottom and the global latent transformer, the variable-size coarsest-mesh feature tensor $h^{(L-1)} \in \mathbb{R}^{N_{L-1} \times H}$ is mapped to a fixed M-token latent through a cross-attention head whose queries are M learnable tokens that cross-attend to $h^{(L-1)}$. A symmetric decoder head maps the processed latent back to the coarsest mesh, coarse-mesh skip features cross-attend to the M-token latent, where the result is recombined with the V-cycle skip and propagated up. Both heads share their weights across all input meshes, so any coarsest-mesh discretisation is mapped into a single shared latent space of M pseudo-coarse tokens. For multi-domain training across D shapes the queries on both heads additionally receive a domain conditioning bias, an MLP projection of the one-hot domain identity $d \in \{0, 1\}^D$ added to the queries so the same shared latent is reached from any shape.

Table 4: Default architecture hyperparameters.

Hidden dimension H	128
Hierarchy levels L	3–4
MP layers per level K	2
FEM patch resolution P	5
FEM basis type	P1 (bilinear)
Mixing mode	vector
Radius multiplier μ	2.0–4.0
Timestep embedding dim d_t	64
Transformer blocks K_T	4
Transformer heads	4
Pooling type	learned (residual)

Table 5: Parameter counts on `GaussianBlobs` (square domain) at each model's default configuration.

Model	Parameters
GRIFDIR (ours)	2,822,209
MP-GNN (simple_mp)	4,580,417
FNO (grid positional encoding)	2,891,018
CNN U-Net	2,122,946
GAOT (64×64 latent)	3,471,753

Global latent transformer. At the coarsest level $\ell = L-1$ the N_{L-1} tokens are processed by K_T DiT blocks (Peebles & Xie, 2023) with AdaLN-Zero conditioning, capturing long-range dependencies that local message passing cannot resolve:

$$h \leftarrow h + \boldsymbol{\gamma}_1 \odot \mathrm{MHSA}\big(\mathrm{AdaLN}(h;\, \gamma(t))\big),$$
$$h \leftarrow h + \boldsymbol{\gamma}_2 \odot \mathrm{FFN}\big(\mathrm{AdaLN}(h;\, \gamma(t))\big),$$

with output gates $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2$ zero-initialised. Coarse-mesh coordinates are encoded by a small MLP and added as positional embeddings before the first block. The decoder mirrors the encoder, mapping $\mathbb{R}^H$ back to the C-channel signal at the finest mesh.

FEM filter family. Equation (12) parametrises each message through a $P \times P$ filter $\{w_p\}_{p=1}^{P^2}$ evaluated on a fixed reference patch. We use the bilinear nodal P_1 basis: P^2 filter weights at a uniform reference grid, where each sample contributes to its four surrounding weights via bilinear interpolation summing to one. The basis returns $\mathbf{0}$ outside the patch, giving the layer a strict physical receptive field. The filter coefficients combine with source-node features through a per-edge vector gate $m_{ij} = (W_g b_{ij}) \odot (W_m h_j)$ with $W_g \in \mathbb{R}^{H \times P^2}$.

D.1 Multi-Channel FEM Convolution

We now consider the case of a signal $a : \Omega \to \mathbb{R}^C$ with C channels and we want to map to C' output channels. We denote $a^{(c)}$ as the c-th component of a. We write the filter for the (c', c) channel pair as $\kappa^{(c',c)}(\xi) = \sum_p w_p^{(c',c)}\, \phi_p(\xi)$, one $P\times P$ filter image per pair, directly analogous to a standard CNN. The output at channel c' is

$$\begin{aligned}
(\mathcal{K}a)^{(c')}(x_i) &= \sum_{c=1}^{C} \int_{B_r(x_i)} \kappa^{(c',c)}\big(\Phi_{x_i}(y)\big)\, a^{(c)}(y)\, \mathrm{d}y \\
&= \sum_{c=1}^{C} \sum_{p=1}^{P^2} w_p^{(c',c)} \int_{B_r(x_i)} \phi_p\big(\Phi_{x_i}(y)\big)\, a^{(c)}(y)\, \mathrm{d}y \\
&\approx \sum_{c=1}^{C} \sum_{p=1}^{P^2} w_p^{(c',c)} \cdot \sum_{j \in N(i)} V_{i,j} \phi_p\big(\Phi_{x_i}(x_j)\big)\, a^{(c)}(x_j),
\end{aligned} \tag{30}$$

giving a total of $C'\times C\times P^2$ trainable filter weights per layer.

D.2 Hyperparameters

In Table 4 we give the default hyperparameters that were used in our network architecture. We also provide the number of parameters for each model at their default configurations in Table 5.

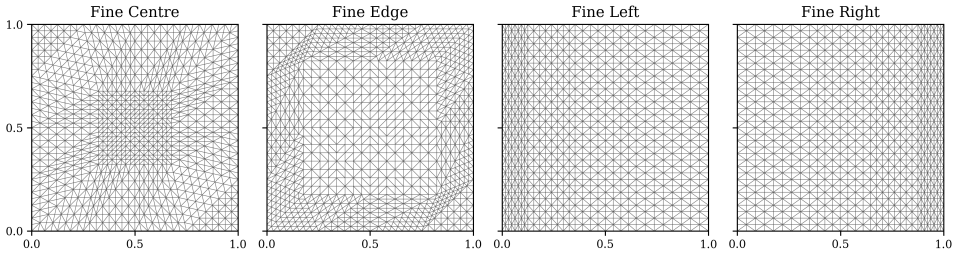


Figure 7: To test the discretisation invariance of the method we use the following graded meshes where the density of nodes is uneven across the mesh.

E Experimental Details

The source code will be made publicly available upon acceptance.

E.1 Datasets

E.1.1 Gaussian Blobs

Each sample is a conductivity field $\sigma : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^2$, generated as

$$\sigma(x) = \min_{i=1,\dots,M} \left[\sigma_b - (\sigma_b - \sigma_i^*) \exp\left(-\frac{1}{2} \left\| \mathbf{A}_i^{-1} \mathbf{R}_i^\top (x - \boldsymbol{\mu}_i) \right\|^2 \right) \right], \tag{31}$$

where $M \sim \mathcal{U}\{1, \dots, N\}$ is the number of inclusions and σ_b is the background conductivity. Each inclusion i is parameterised by:

- a centre $\boldsymbol{\mu}_i$, constrained such that the blob lies fully within Ω;
- semi-axes $a_i \sim \mathcal{U}[a_{\min}, a_{\max}]$ and $b_i = r_i a_i$ with $r_i \sim \mathcal{U}[0.5, 1.0]$, collected in $\mathbf{A}_i = \mathrm{diag}(a_i, b_i)$;
- a rotation matrix $\mathbf{R}_i$ with angle $\alpha_i \sim \mathcal{U}[0, 2\pi)$;
- a minimum conductivity $\sigma_i^* = \sigma_b - d_i(\sigma_b - \sigma_{\min})$ with depth $d_i \sim \mathcal{U}[0.5, 1.0]$.

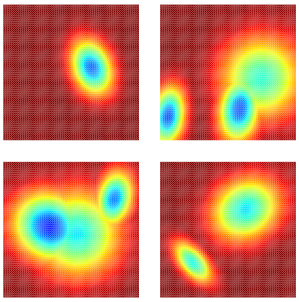


Figure 6: Ground truth samples from `GaussianBlobs`.

The same generator is used to build the following datasets that differ only in the support domain Ω and its triangulation.

Square. The domain $\Omega = [0, 1]^2$ is tiled by a regular 32×32 Cartesian grid with each unit cell split into two triangles ($32 \times 32 \times 2 = 2{,}048$ triangles in total). The score network operates on the *dual graph* of this triangulation: each triangle is a node carrying one DG-0 (piecewise-constant) scalar value, with an edge connecting any two triangles that share a mesh edge. This dual-graph layout gives $N_0 = 2{,}048$ nodes at the finest hierarchy level.

Cross-domain. Eight non-square shapes; `circle`, `circle_with_hole`, `e_shape`, `l_shape`, `plus`, `square_with_hole`, `x_shape`, and a sub-sample of the square at the same hierarchy, each meshed by Gmsh (Geuzaine & Remacle, 2009) at $h_{\max} = 0.1$, used for the cross-shape generalisation experiments.

For every dataset we evaluate σ at the cell-centroid coordinates of its triangulation, store the field as one channel per cell, and apply a $90/10$ train/test split. Table 6 reports total sample counts and per-level node counts of the multi-resolution hierarchy used by the score network.

E.1.2 Pinball

We consider the fluid pinball setup from Tomasetto et al. (2025) (Section D.). The pinball example involves a 2D advection-diffusion problem with an implicit parametric dependence. In particular, they

Table 6: `GaussianBlobs` datasets: total samples ($\lceil 0.9N \rceil$ assigned to train at runtime) and node counts at each hierarchy level N_0, N_1, N_2, N_3 (finest to coarsest).

Domain / pattern	N_{samples}	N_0	N_1	N_2	N_3
Square					
`nx32_ny32`	10,000	2,048	512	128	32
Cross-domain (8 shapes, $h_{\max} = 0.1$)					
circle	1,000	2,974	757	212	119
circle_with_hole	1,000	2,736	714	202	104
e_shape	1,000	3,380	860	246	81
l_shape	1,000	2,814	732	190	58
plus	1,000	4,718	1,218	318	130
square	1,000	2,048	512	128	32
square_with_hole	1,000	3,642	942	248	173
x_shape	1,000	2,804	726	206	98

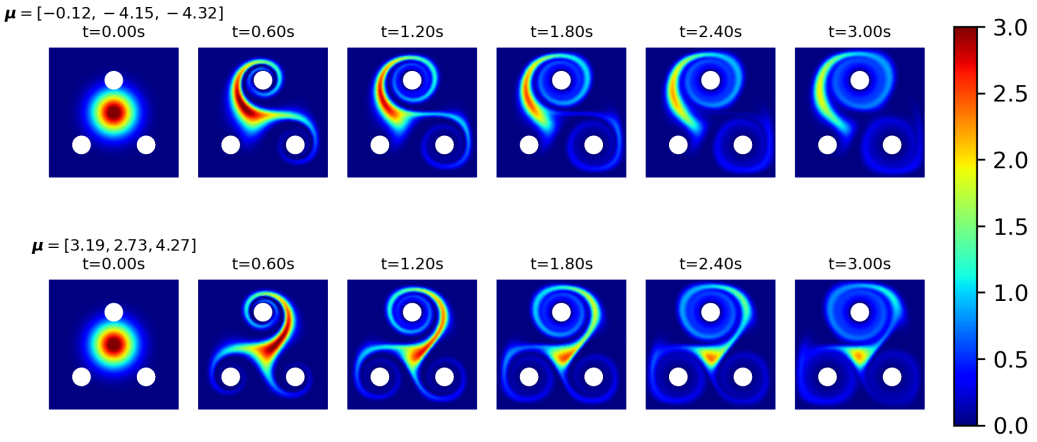


Figure 8: Two rollouts of `Pinball` for different velocities $\boldsymbol{\mu}$ from the training set.

consider a (incompressible) fluid in a square domain $[-1, 1]^2$ with three rotating cylinders centred at $(-0.5, -0.5), (0.5, -0.5)$ and $(0.0, 0.5)$ with radius 0.15. The constant rotation is given by velocities $\boldsymbol{\mu} = [v_1, v_2, v_3]$. The rotation leads to a fluid motion inside the square domain, where the velocity $v : [-1, 1]^2 \to \mathbb{R}^2$ and pressure $p : [-1, 1]^2 \to \mathbb{R}$ are determined by the (steady-state) Navier-Stokes equation. Tomasetto et al. (2025) use viscosity $\nu = 1.0$ and no-slip boundary conditions on the external walls and Dirichlet boundary conditions on the cylinders. They then consider a quantity (e.g., mass or particle density) $\rho : [-1, 1]^2 \times [0, T] \to \mathbb{R}$ that spreads inside the domain according to

$$\rho_t + \nabla \cdot (-\eta \nabla \rho + v(\boldsymbol{\mu}) y) = 0, \tag{32}$$

with homogeneous Neumann boundary conditions, $\eta = 0.001$, $T = 3$ and $v(\boldsymbol{\mu})$ as the solution of the Navier-Stokes equation given the velocity of the cylinders $\boldsymbol{\mu}$. For a fixed initial condition

$$\rho(x, 0) = \frac{10}{\pi} \exp(-10x_1^2 - 10x_2^2) \tag{33}$$

the distribution ρ depends directly on the fluid velocity $v(\boldsymbol{\mu})$. We collect roll-outs of ρ_t for 400/50/50 varying $\boldsymbol{\mu}$ as our training/validation/test sets and aim to learn this distribution (independent of t). This is a challenging distribution as the domain has three holes (where the cylinders are). Examples of the pinball dataset are shown in Figure 8. We plot the mesh for different length scales in Figure 9. In addition to the meshes shown there we use meshes with a length scale 0.03 and 0.025 containing 5,266 and 7,525 mesh vertices.

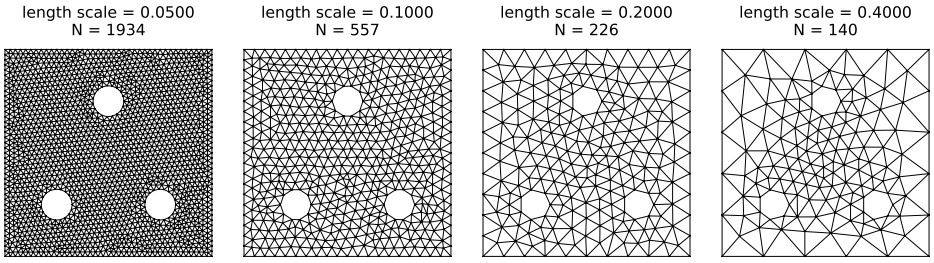


Figure 9: The `Pinball` mesh for different length scales and number of elements N.

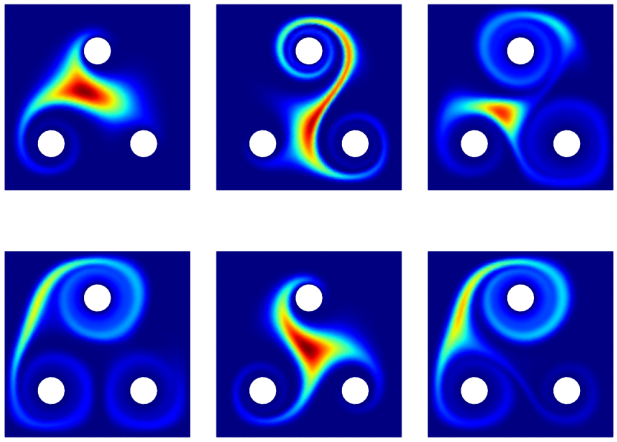


Figure 10: Unconditional samples for `Pinball`. See Figure 8 for samples from the dataset. The same colour scale is used in both figures, with a minimum of 0 and a maximum of 3.

E.1.3 CYLINDER FLOW

We make use of the `CylinderFlow` dataset from Han et al. (2022). The dataset describes the flow around a cylinder on a fixed (unstructured) mesh of 1699 nodes. The dataset consists of 101 simulations each for 400 timesteps. Every sample has three channels, i.e., velocity in x-direction (u), velocity in y-direction (v) and pressure p. We use the same train-test split as Shi et al. (2025) and use 30 000 samples for training and 5000 for testing.

E.1.4 DEEPJEB

`DeepJEB` (Hong et al., 2025) contains 2138 3D bracket geometries, each with the von Mises stress field induced by a vertical load. The stress concentrates in small hotspots near the mounting points, and the geometries themselves vary a lot, so the model has to generalise across shapes. We use 1924 brackets for training and the remaining 100 for testing.

We use the 3D version of our model (via a trilinear interpolation of the kernel), with the same multi-scale architecture. The brackets are represented as point clouds and we create a graph using kNN with 8 neighbours. We train two models on different resolutions, one capping the sampled nodes at 6000 points and one at 24 000 points. For both, we use three downsampling ratios, taking $100\%, 25\%$ and 6.25% of the input points.

E.2 BASELINE APPROACHES

MP-GNN GNN based on the message passing framework, modelled after the MP-PDE Solver (Brandstetter et al., 2022). Each node corresponds to a spatial point on the mesh, and edges encode local spatial connectivity. For each edge (i, j), the message is computed from the hidden states h_i, h_j, the difference in input values $a_i - a_j$, and the relative position $x_i - x_j$:

$$m_{ij} = \phi_m\big([h_i, h_j, a_i - a_j, x_i - x_j]\big)$$

where ϕ_m is a two-layer MLP with Swish activations. Using differences rather than raw values makes the messages translation-equivariant in both signal and position space. Aggregated messages (mean pooling) are combined with the current hidden state via a residual connection:

$$h_i \leftarrow h_i + \phi_u\big([h_i, , \bar{m}_i]\big),$$

followed by instance normalisation. The time step t is embedded via a Fourier feature embedding and concatenated with the node features in the encoder, allowing the network to function as a time-conditioned score network $s_\theta(a, t, x)$ for diffusion-based generative modelling over function spaces. The configuration used hidden dimension 128, 4 message-passing layers, mean aggregation, time embedding dimension equal to the hidden dimension.

FNO The Fourier Neural Operator (Li et al., 2021) parametrises the score in the spectral domain. Each FNO block lifts the input with a 1×1 convolution, applies a spectral convolution that truncates the 2D FFT to its top-K low-frequency modes, multiplies by a learned complex weight tensor at those modes, and inverse-transforms back to the spatial domain; the spectral output is added to a residual 1×1 convolution and passed through a GELU non-linearity:

$$h \leftarrow \sigma\big(\mathcal{F}^{-1}\big(R \cdot \mathcal{F}(h)\big)_{|k| \le K} + Wh\big).$$

Because the network operates on a regular grid, the unstructured centroid mesh is first interpolated onto a 32×32 image grid (matching the conductivity dataset's reference resolution), processed by the FNO stack, and bilinearly interpolated back to the mesh nodes for loss computation. Time conditioning follows the same Fourier-feature embedding as the GNN baselines, concatenated as an extra channel before the first lift. Default configuration: 4 FNO blocks, $K = 8$ modes per spatial axis, hidden dimension 128, we use grid-coordinate positional encoding and optional 12.5% domain padding to handle non-periodic boundaries.

GAOT The Geometry-Aware Operator Transformer (GAOT) (Wen et al., 2025) is a continuous-kernel encoder/decoder paired with a transformer processor. The diffusion time step t is mapped through a Fourier-feature time embedding and projected to n_{time} extra channels which are concatenated to every node's input alongside the function value $a(x_j)$ (and an optional domain one-hot d_j in the

multi-domain setting), giving a per-node feature vector $\tilde{a}(x_j) = [a(x_j), \tau(t), d_j]$. The MAGNO encoder then lifts these features onto a fixed regular grid of latent tokens via a continuous radius-ball kernel: for each latent location ξ,

$$z(\xi) = \frac{1}{|N(\xi)|} \sum_{x_j \in B_r(\xi)} \kappa_\theta\big(\xi - x_j,\, \tilde{a}(x_j)\big),$$

where κ_θ is a per-edge MLP and the aggregation is normalised by neighbour count. The latent token grid is patchified and processed by a stack of standard ViT-style transformer blocks with multi-head self-attention. A symmetric MAGNO decoder lifts the processed tokens back onto the mesh, and a learned signed-distance geometric embedding is added to the latent tokens before the processor. Default configuration: latent grid 64×64, MAGNO radius 0.033 in unit-square coordinates, 3-layer MAGNO MLP with hidden dimension 64 and lifting dimension 64, transformer hidden dimension 256, time embedding dimension 128 projected to $n_{\text{time}} = 8$ per-node channels, 3 transformer blocks, patch size 2, geometric embedding and self-attention enabled, absolute positional encoding.

CNN We additionally evaluate a CNN-based baseline. As CNNs operate on regular grids, we first interpolate the mesh representation onto a fixed image grid. A standard Attention U-Net (Dhariwal & Nichol, 2021) is then applied in image space, after which the predictions are interpolated back onto the original mesh. Default configuration: hidden dimension 128, 4 down/up-sampling stages with single convolutions per block, residual connections, attention at the bottleneck, max-pool downsampling and bilinear upsampling, no dropout.

MINO The Mesh-Informed Neural Operator (MINO) (Shi et al., 2025) is a transformer-based neural operator designed for generative modelling of functions on arbitrary discretisations. A graph neural operator encoder aggregates the input function, together with the diffusion time embedding, from the input mesh onto a fixed set of $N \times N$ latent grid in the domain via a radius-ball kernel integral, analogous to Equation (10). The resulting latent tokens are processed by a stack of attention blocks. A cross-attention decoder, in which the coordinates of the output nodes act as queries attending to the latent tokens, returns the score at every mesh node. Because the latent query set is independent of the input mesh, MINO is discretisation-agnostic by construction. Unlike GRIFDIR, it has no hierarchy of local convolutions: all mesh information is compressed into the latent tokens in a single encoding step. Time conditioning follows the same Fourier-feature embedding as the other baselines. We use MINO's code at `https://github.com/yzshi5/MINO` for our experiments, however we swap the flow matching objective for a diffusion objective. Default configuration: latent grid 64×64, hidden dimension 128, encoder transformer depth 4, attention heads 4, decoder depth 2 and a GNO neighbour radius of 0.07.

E.3 Evaluation Metrics

For conditional sampling we obtain a set of samples $\{\hat{a}_i^{(k)}\}, k = 1, \ldots, K$ for every observation y_i, $i = 1, \ldots, n$. Here n is the number of observations and K the number of posterior samples per observation. We compute the MSE of the posterior mean, the energy score. To evaluate the unconditional models we evaluate the MMD.

MSE/RMSE of the Posterior Mean For each observation y_i, we compute the posterior mean as

$$\bar{a}_i = \frac{1}{K} \sum_{k=1}^{K} \hat{a}_i^{(k)}.$$

The MSE/RSME is then defined as

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \|a_i - \bar{a}_i\|^2, \quad \text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \|a_i - \bar{a}_i\|^2}$$

This metric measures the average deviation between the ground truth parameters a_i and the posterior mean estimates.

Energy Score (ES) (Gneiting & Raftery, 2007) For each observation y_i with posterior samples $\{\hat{a}_i^{(k)}\}_{k=1}^K$, the energy score with $\beta = 1$ is defined as

$$\mathrm{ES}_i = \frac{1}{K}\sum_{k=1}^{K}\left\|\hat{a}_i^{(k)} - a_i\right\|^\beta - \frac{1}{2K^2}\sum_{k=1}^{K}\sum_{l=1}^{K}\left\|\hat{a}_i^{(k)} - \hat{a}_i^{(l)}\right\|^\beta.$$

The final energy score is obtained by averaging over all samples,

$$\mathrm{ES} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{ES}_i.$$

The ES jointly evaluates calibration and sharpness of the posterior distribution. Lower values indicate better probabilistic predictions.

Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) Given the sets of samples $X = \{x_1, \ldots, x_n\}$ and $Z = \{z_1, \ldots z_m\}$ for $n, m \in \mathbb{N}$, we compute the squared MMD using the standard unbiased U-statistic estimator

$$\mathrm{MMD}_u^2(X, Z) = \frac{1}{n(n-1)}\sum_{i\neq j}k(x_i, x_j) + \frac{1}{m(m-1)}\sum_{i\neq j}k(z_i, z_j) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}k(x_i, z_j), \tag{34}$$

for a Gaussian kernel $k(\cdot, \cdot)$ with length scale ℓ, defined as

$$k(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\ell^2}\right). \tag{35}$$

Sliced Wasserstein Distance (SWD) (Bonneel et al., 2015) The sliced Wasserstein distance measures distances between two distributions μ and ν in high-dimensional spaces by choosing random directions in the space and projecting the data onto it, resulting in two 1D distributions for which we can cheaply find the Wasserstein distance. For $p \geq 1$, the sliced p-Wasserstein distance between distributions μ and ν on $\mathbb{R}^N$ is

$$\mathrm{SW}_p(\mu, \nu) = \left(\int_{\mathbb{S}^{N-1}} W_p^p\big(\theta_{\#}\mu, \theta_{\#}\nu\big)\,\mathrm{d}\sigma(\theta)\right)^{1/p}, \tag{36}$$

where σ is the uniform measure on the unit sphere $\mathbb{S}^{N-1}$ and $\theta_{\#}\mu$ is the push-forward of μ under $x \mapsto \langle\theta, x\rangle$. To approximate this, we use the sliced Wasserstein distance function using `POT` (Flamary et al., 2021), with $p = 2$ and 256 projections. We draw compute the SWD with 5 different seeds for the random projection and compute the mean as the final score.

E.4 Computational Resources

Training of the models was performed on a cluster where each node comprises 2 Intel Xeon Platinum 8468 CPUs (96 cores total) and 4 Intel Data Center GPU Max 1550 (Ponte Vecchio) GPUs with 128 GB memory per GPU. Training took about 24h for 100 epochs on the Pinball dataset, 6h for the Gaussian Blobs for 250 epochs, Cylinder Flow 24h for 200 epochs and DeepJEB 20h for 50 epochs. Inference (unconditional and conditional sampling) was performed on a GPU cluster comprising NVIDIA V100, A100 and H100 GPUs.

F Further results and ablations

In this section, we provide further results for the experiments reported in Sections 4.2 to 4.4. Namely, in Section F.1 we provide results from the conditional experiments, in Section F.2 we provide ablations of our model and in Section F.4 we provide scaling details on `DeepJEB`.

Table 7: Reconstruction errors for the Poisson inverse problem on `GaussianBlobs`.

	RMSE (↓)	Energy Score (↓)
Fun-DPS	0.0197 ± 0.0128	0.6709 ± 0.4479
Fun-DAPS	$\mathbf{0.0069 \pm 0.0049}$	$\mathbf{0.2549 \pm 0.1919}$

Table 8: We evaluate the models on graded meshes for the `GaussianBlobs` dataset and evaluate using 1024 samples for all models. We report the MMD and SWD metrics. Best results are in **bold**, second best are underlined.

	Fine left		Fine centre		Fine edge		Fine right	
	MMD (↓)	SWD (↓)	MMD (↓)	SWD (↓)	MMD (↓)	SWD (↓)	MMD (↓)	SWD (↓)
GAOT	0.730	6.078	0.687	4.015	0.756	3.582	0.191	0.077
MINO	**0.136**	**0.046**	**0.056**	**0.021**	0.183	0.063	**0.136**	**0.048**
MP-GNN	0.223	0.076	0.106	0.036	0.218	0.076	0.234	0.083
GRIFDIR	0.231	0.079	0.110	0.044	**0.062**	**0.026**	0.239	0.080

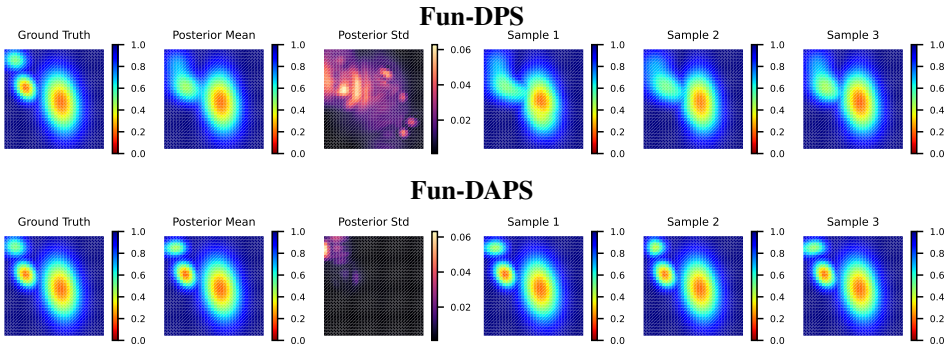


Figure 11: We use Fun-DPS and Fun-DAPS to sample conditionally on the Poisson operator for `GaussianBlobs`. The posterior mean and standard deviation are taken across 100 samples, with three such samples plotted on the right.

F.1 Conditional Sampling

We use the unconditionally trained GRIFDIR as a prior and approximate the guidance term with Fun-DPS and Fun-DAPS, drawing $K = 100$ posterior samples per observation. Fun-DPS backpropagates through the score network at every step, whereas Fun-DAPS is backpropagation-free and hence cheaper.

Sparse Sensors We condition on 10 to 100 randomly placed point observations for the `GaussianBlobs` (see Figure 13) and 25 to 100 for the `Pinball` dataset (Figure 12, Table 9). On both datasets, RMSE and energy score decrease steadily as sensors are added, as we would expect. Fun-DPS is consistently more accurate than Fun-DAPS. However, Fun-DPS has a higher computational cost, as it requires to backpropagate through the score model at every sampling step.

Poisson We apply our model to the Poisson inverse problem in the noiseless setting for `GaussianBlobs`. For Fun-DPS we use a guidance weight of 2.0 and for Fun-DAPS, we set the noise level to $\beta_y = 1e-5$, with 20 Langevin steps. For both we use 400 sampling steps. We report the mean and standard deviation over 100 independent reconstructions for one observation, alongside three samples, in Figure 11. Doing the same, we get the reconstruction errors reported in Table 7, where we see in this case Fun-DAPS is more accurate.

Conditional sampling for `Pinball` We evaluate conditional sampling for `Pinball` on 50 random velocity parameters $\boldsymbol{\mu}$, each rolled out over 30 time steps in $[0, T]$. Here, we train GRIFDIR conditional on the cylinder velocities μ and time t. For each case, we draw one conditional sample and report the mean ℓ^2 error against the ground truth ρ_t. As shown in Figure 14, the error is minimal at $t = 0$ since the initial condition is fixed and independent of $\boldsymbol{\mu}$. It increases during the early transient phase as ρ is advected by $v(\boldsymbol{\mu})$, while the standard deviation remains moderate, indicating consistent generalisation across parameters. Overall, the model captures the parametric dependence of the advection–diffusion dynamics and produces accurate samples over the full time horizon.

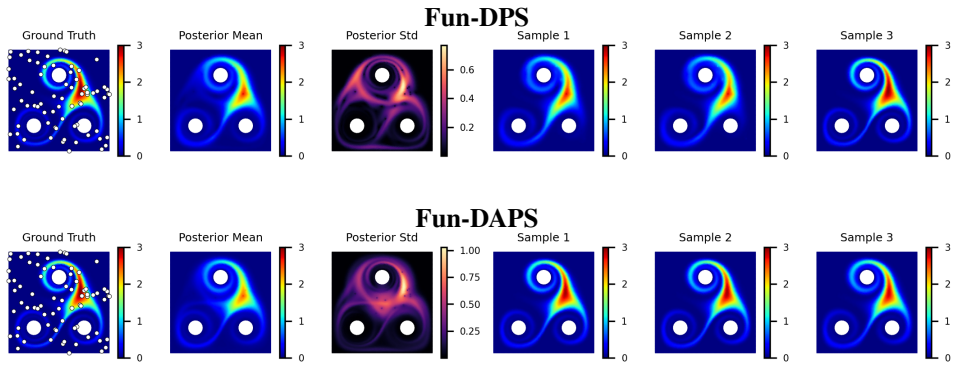


Figure 12: Sparse sensor reconstruction with 75 randomly placed sensors with Fun-DPS and Fun-DAPS for `Pinball`. For each method, we show the ground truth, posterior mean and standard deviation (computed with 100 samples), and three posterior samples.

Table 9: Performance for varying number of sensors on `Pinball`.

	25 sensors		**50 sensors**		**75 sensors**		**100 sensors**	
	RMSE (↓)	ES (↓)	RMSE (↓)	ES (↓)	RMSE (↓)	ES (↓)	RMSE (↓)	ES (↓)
Fun-DPS	0.289 ± 0.052	16.6 ± 3.33	0.188 ± 0.054	9.21 ± 3.59	0.140 ± 0.042	6.18 ± 2.54	0.110 ± 0.042	4.53 ± 2.29
Fun-DAPS	0.332 ± 0.059	20.1 ± 4.45	0.249 ± 0.064	14.03 ± 4.37	0.218 ± 0.057	11.5 ± 3.82	0.183 ± 0.054	8.90 ± 3.65

F.2 Ablations

Transformer, FiLM and Aggregation First, we compare mean aggregation against volume-weighted ("lumped") aggregation and the effect of the latent transformer and the FiLM embedding. We do this on `GaussianBlobs`, trained on 2,048 nodes and report the results in Table 10. We see that for evaluating on higher resolutions, using FiLM, lumped aggregation and the transformer performs best. Keeping FiLM and the transformer fixed, using lumped aggregation tends to improve results and in the cases it doesn't we get roughly the same scores. Indeed, without it, we see many of the experiments on the coarser and finer resolutions collapse to generating near-constant samples.

FEM radius and multiscale hierarchy depth We vary the radius multiplier of the FEM convolution applied to `GaussianBlobs`. Throughout the experiments, we use a multiplier of 2 as the default. The ablation for the radius multiplier is in Table 11. Here, we see that a small radius multiplier ($\times 1.0$) is not working as well, in particular for the coarse 1152 points mesh. This is because in the lowest resolution with a small radius, the number of neighbours inside the radius is too small. Note, that increasing the radius also increases the computational cost as more neighbours are taken into account

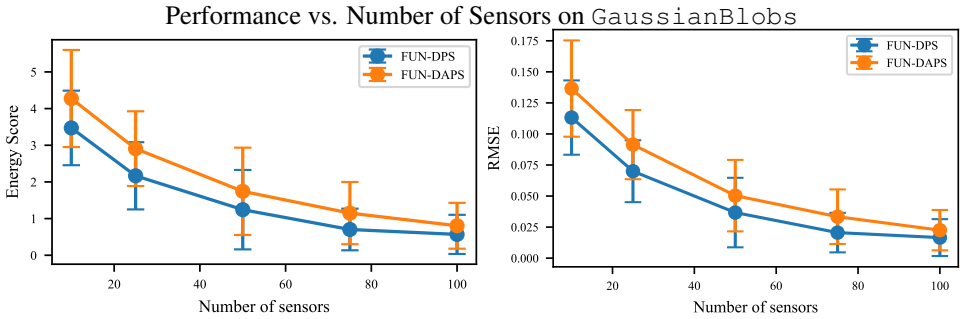


Figure 13: Conditional Sampling for `GaussianBlobs`. We compare the conditional methods Fun-DPS and Fun-DAPS for sampling from the conditional distribution arising from an increasing number of randomly placed sensors.

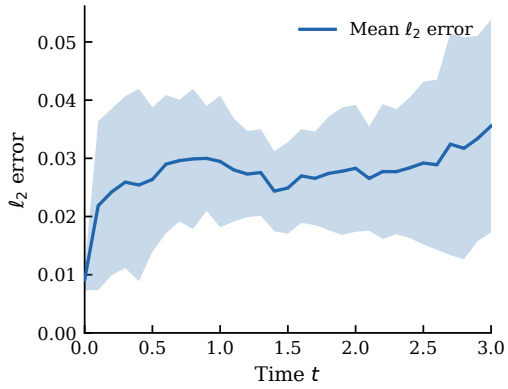


Figure 14: The mean ℓ_2-error of the conditional GRIFDIR model trained on `Pinball` over the rollout time t.

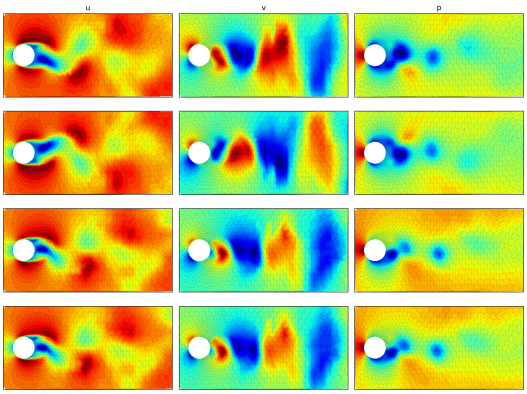


Figure 15: Unconditional samples from `CylinderFlow` dataset using GRIFDIR.

for the graph convolution. The experiments for the number of downsampling steps is in Table 12. We note that, although using the single scale gives good results, since the mesh is not downsampled, it is also more computationally heavy. The metrics remain relatively stable across different downsampling regimes.

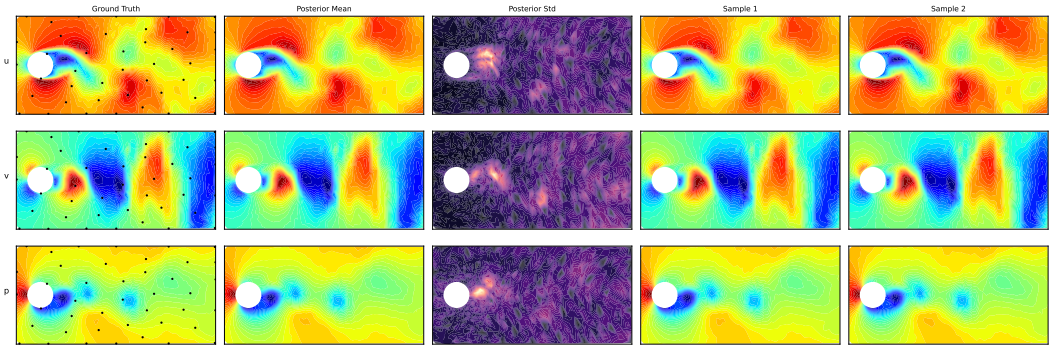


Figure 16: Conditional Samples from `CylinderFlow` dataset using GRIFDIR using 42 sensor points with a farthest-point placement of the sensors. We show the ground truth, the posterior mean, the posterior pointwise standard deviation and two samples.

Table 10: Component ablation of the latent transformer (Trans.), the FiLM time embedding and the lumped aggregation for `GaussianBlobs`. MMD computed with 1024 generated samples against 2000 reference samples; SWD computed with 1024 samples. Columns give the number of nodes in the evaluation discretisation; "Mean" is the average over the three discretisations.

			MMD ↓				SWD ↓			
Trans.	FiLM	Lumped	1152 pts.	2048 pts.	8192 pts.	Mean	1152 pts.	2048 pts.	8192 pts.	Mean
✓	✓	✓	0.064	0.090	**0.053**	**0.069**	0.029	0.033	**0.025**	**0.029**
✓	✓	—	0.471	0.088	0.379	0.313	0.162	0.033	0.130	0.108
✓	—	✓	0.094	<u>0.078</u>	0.253	0.142	0.035	**0.028**	0.089	0.051
✓	—	—	0.652†	0.097	0.650	0.466	0.260†	0.035	0.256	0.184
—	✓	✓	<u>0.055</u>	**0.075**	0.234	<u>0.121</u>	<u>0.026</u>	**0.028**	0.082	<u>0.045</u>
—	✓	—	0.642†	0.111	0.654†	0.469	0.255†	0.040	0.265†	0.187
—	—	✓	0.069	0.101	<u>0.212</u>	0.127	0.028	0.036	<u>0.075</u>	0.046
—	—	—	**0.048**	0.085	0.241	0.125	**0.022**	<u>0.031</u>	0.086	0.046

†Generated samples collapse to a near-constant field.

Table 11: Ablation of the FEM-Conv radius for GRIFDIR on `GaussianBlobs`.

	MMD ↓			SWD ↓		
Radius	1152 pts.	2048 pts.	8192 pts.	1152 pts.	2048 pts.	8192 pts.
radius ×1.0	0.3969	0.1273	0.1992	0.1378	0.0443	0.0709
radius ×2.0	**0.0585**	**0.0975**	**0.0287**	**0.0248**	**0.0353**	**0.0191**
radius ×3.0	0.0943	0.1087	0.1353	0.0345	0.0391	0.0549

Table 12: Component ablation on the Gaussian blob dataset (2048 points).

Depth	MMD ↓	SWD ↓
1 — 2048 (single scale)	0.111	<u>0.040</u>
2 — 2048/512	<u>0.102</u>	**0.036**
3 — 2048/512/128	0.103	**0.036**
4 — 2048/512/128/32	**0.100**	0.059

F.3 KERNEL CHOICE

Comparison of kernels in diffusion model setup We replace our FEM kernel (FEM-Conv) with an MLP kernel (see, e.g., Gilmer et al. (2017)), a spline kernel (Fey et al., 2018) (Spline-Conv), a standard GNO matrix kernel (Anandkumar et al., 2019) (GNO-Conv), and a MoNet-style Gaussian-mixture kernel (GMM-Conv) (Monti et al., 2017), keeping the rest of the architecture fixed. For the Spline kernel, the standard implementation uses a k-hop mesh-adjacency neighbourhood rather than a physical radius. We report both the original k-hop mesh-adjacency neighbourhood and a version using the physical-radius neighbourhood, to separate the effect of the kernel from the effect of the neighbourhood definition. We evaluate the MMD and SWD; Results are in Table 13.

Mexican Hat kernel recovery To isolate the kernel parametrisation from the rest of the architecture, we replace the diffusion model with a supervised operator-learning task solved by a single convolution layer (Figure 17). A fixed target kernel κ^*, a rotated anisotropic Mexican hat supported on the disc $|x| \le r$ with $r = 0.12$, defines the operator $y_i = \sum_j w_j \, \kappa^*(x_i - x_j) \, a_j$ on N uniformly random points in the unit square, where w_j is the Voronoi area of point j and the inputs a are Gaussian random fields with length scale 0.1. The support of each kernel is enforced being applied on an $r-$radius graph formed by the N random point cloud. Each kernel is applied between a linear encoder and decoder and is trained on 64 input-output pairs at $N = 2048$ for 2000 Adam steps; all kernels share the same support, data and training budget. We compare our FEM-Conv with lumped quadrature, the same FEM kernel with mean aggregation, SplineConv which has mean aggregation Fey et al. (2018), and an MLP kernel that takes as input the offset and distance, $(\Delta x, \Delta y, |\Delta x|)/r$, with lumped or mean aggregation. For FEM-Conv and SplineConv the resolution is an $R \times R$ node grid on $[-r, r]^2$;

Table 13: MMD and SWD for `GaussianBlobs` on different resolutions for different choices of the kernel parametrisation in GRIFDIR. All models use the same backbone architecture.

	MMD ↓			SWD ↓		
Kernel	1152 pts.	2048 pts.	8192 pts.	1152 pts.	2048 pts.	8192 pts.
MLP-Conv	0.281	0.284	0.222	0.101	0.104	0.084
GNO-Conv	0.329	0.339	-	0.111	0.115	-
GMM-Conv	0.491	0.402	0.218	0.162	0.127	0.084
Spline-Conv (kNN)	0.315	0.098	0.384	0.106	0.035	0.134
Spline-Conv (radius)	0.647	0.085	0.070	0.256	0.030	0.039
FEM-Conv (ours)	0.163	0.088	0.109	0.060	0.033	0.041

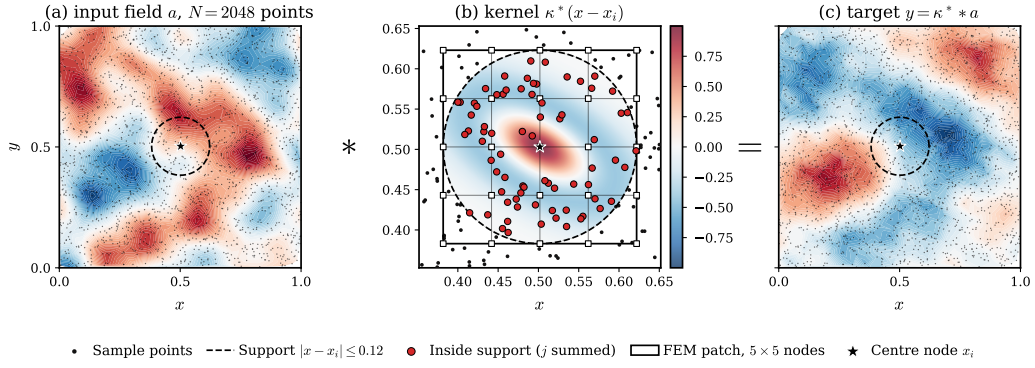


Figure 17: Operator-recovery task at $N = 2{,}048$ points. (a) An input field a. (b) The target kernel κ^* around one centre point x_i: the points inside the support $|x - x_i| \le 0.12$ (red) are summed, and the square is a 5×5 FEM-Conv patch. (c) The target y that a single kernel layer must learn.

for the MLP it is a hidden width of R^2. Each model is linear in its input, so its learned kernel can be read out exactly and compared with κ^*.

After training, each model is evaluated without retraining on new input fields and new point clouds of 512, 2048 and 8192 points; we report the relative MSE of its output y and the relative L^2 error of its kernel against κ^*. Recovered kernels are evaluated on the relative L^2 error on the support (dashed circle), outside which no convolution evaluates the kernel. All mean aggregation kernels divide by the number of neighbours, so each kernels graph is smaller than κ^* by a near-constant factor; each kernel is therefore scaled back towards κ^* by a single least-squares factor and the remaining error is reported. Full results are shown in Table 14 and the recovered kernels are plotted in Figure 18.

In the special case of using a degree-1 B-spline kernel, the SplineConv (mean aggregation) recovers the FEM-conv (mean aggregation) and give the same kernels and errors at every resolution as both use piecewise-linear basis functions on a regular grid. SplineConv applies its basis to mesh-normalised pseudo-coordinates and averages over neighbours, whereas FEM-Conv defines the kernel on a fixed physical patch and aggregates with the lumped finite-element quadrature. Only lumped FEM-Conv recovers the rotated, anisotropic target converges to the true kernel. Its kernel error falls from 0.70 at 3×3 to 0.13 at 9×9 and 0.03 at 21×21. This is in accordance with Proposition C.2. The MLP kernel failed on the task. Although its input, $(\Delta x, \Delta y, |\Delta x|)/r$, carries the information to learn the direction, it learns a nearly isotropic filter.

F.4 DeepJEB Scaling and additional Samples

Training We consider training two models, one capped at 6000 nodes and the second at $24\,000$ nodes. From this we find that training scales sub-linearly with the number of points: For both models we train for 30 epochs and get a $1.48\times$ wall time increase for $4\times$ the number of points (7.7 hours for 6000 nodes to 11.4 hours for $24\,000$ nodes).

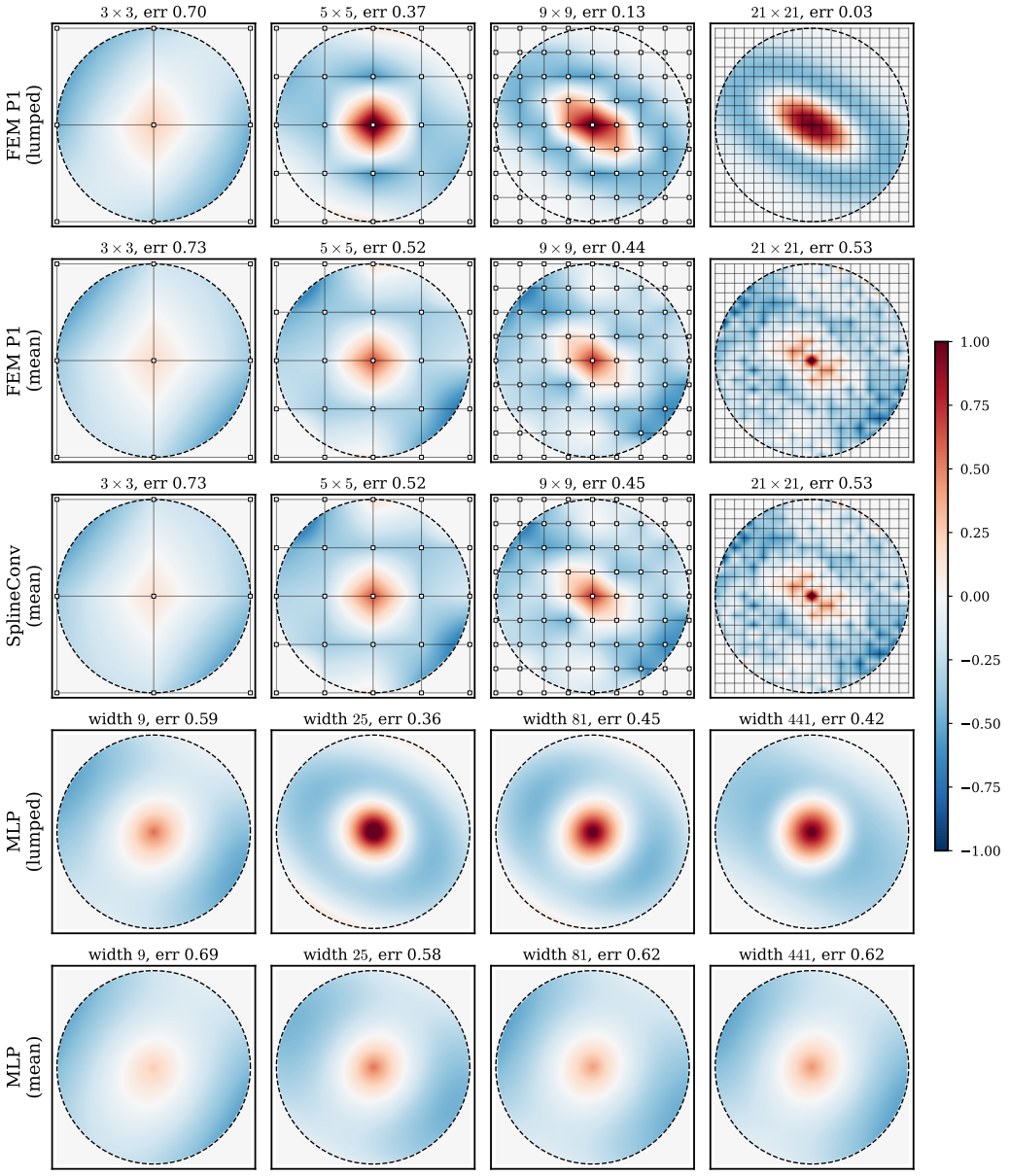


Figure 18: Kernels recovered by a single layer trained at $N = 2{,}048$ points, at increasing resolution: $R \times R$ nodes for FEM-Conv and SplineConv, hidden width R^2 for the MLP.

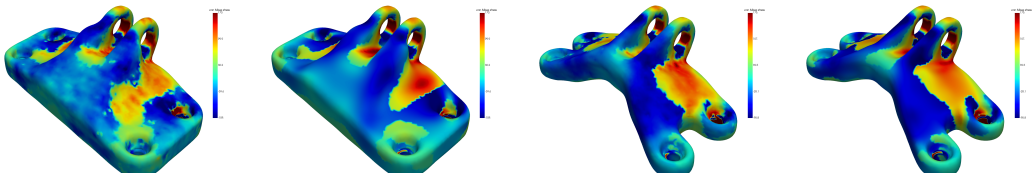
Table 14: Mexican-hat kernel recovery with a single convolution layer trained on $2{,}048$ points: relative MSE of the output y on new point clouds, and relative L^2 error of the recovered kernel against κ^*; mean of two seeds. Best value **bold**.

		Output rel. MSE ↓			Kernel
Kernel	Kernel info.	512 pts.	2048 pts.	8192 pts.	rel. L^2 ↓
FEM Conv (lumped)	3×3 patch	0.22	0.016	0.0046	0.70
FEM Conv (lumped)	5×5 patch	0.059	0.0056	0.00079	0.37
FEM Conv (lumped)	9×9 patch	**0.011**	0.0018	**0.00039**	0.13
FEM Conv (lumped)	21×21 patch	0.016	**0.00059**	0.00043	**0.02**
FEM Conv (mean)	3×3 patch	0.28	0.087	0.059	0.73
FEM Conv (mean)	5×5 patch	0.19	0.070	0.046	0.52
FEM Conv (mean)	9×9 patch	0.16	0.067	0.046	0.44
FEM Conv (mean)	21×21 patch	0.25	0.067	0.054	0.56
SplineConv	3×3 knots	0.31	0.079	0.048	0.73
SplineConv	5×5 knots	0.20	0.071	0.046	0.52
SplineConv	9×9 knots	0.16	0.067	—	0.47
SplineConv	21×21 knots	0.27	0.070	0.052	0.56
MLP (lumped)	width 9	0.17	0.013	0.0039	0.64
MLP (lumped)	width 25	0.14	0.011	0.0036	0.52
MLP (lumped)	width 81	0.076	0.0072	0.0011	0.45
MLP (lumped)	width 441	0.063	0.0053	0.0016	0.34
MLP (mean)	width 9	0.28	0.078	0.051	0.70
MLP (mean)	width 25	0.23	0.078	0.053	0.64
MLP (mean)	width 81	0.24	0.079	0.053	0.63
MLP (mean)	width 441	0.31	0.17	0.14	0.62

Sampling Table 15 shows a scaling law for the sampling time for increasing node densities. We report the sampling cost per bracket for 200 sampling steps and average over 8 held-out brackets. At the higher resolution caps some test bracket data samples have fewer nodes than the cap, so we also report the average number of nodes. Sampling cost grows sub-linearly with resolution: a $4.6\times$ increase in nodes (i.e. going from 6000 to 27 700 nodes) costs $3.5\times$ the sampling time.

Table 15: Sampling time of DeepJEB

Cap	Avg. nodes	Sampling (s)	Node scale	Time scale	Time / node
6000	6000	45.6	1.00	1.00	1.00
12 000	12 000	59.3	2.00	1.30	0.65
24 000	23 427	122.7	3.90	2.69	0.69
32 000	27 702	158.6	4.62	3.48	0.75



(a) Generated sample (load index 0)

(b) Ground truth (load index 0)

(c) Generated sample (load index 10)

(d) Ground truth (load index 10)

Figure 19: Conditional generation with `DeepJEB`. The GRIFDIR model is conditioned on both the 3D bracket geometry and distance to the bracket surface. The figure compares generated von Mises stress fields with the corresponding ground-truth fields for two vertical-load conditions. Samples are generated from the model trained and evaluated at 24 000 points without lumping from Table 3 with 200 sample steps.