OUT-OF-DISTRIBUTION GENERALIZATION OF NEURAL OPERATORS VIA ACTIVE GEOMETRY SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

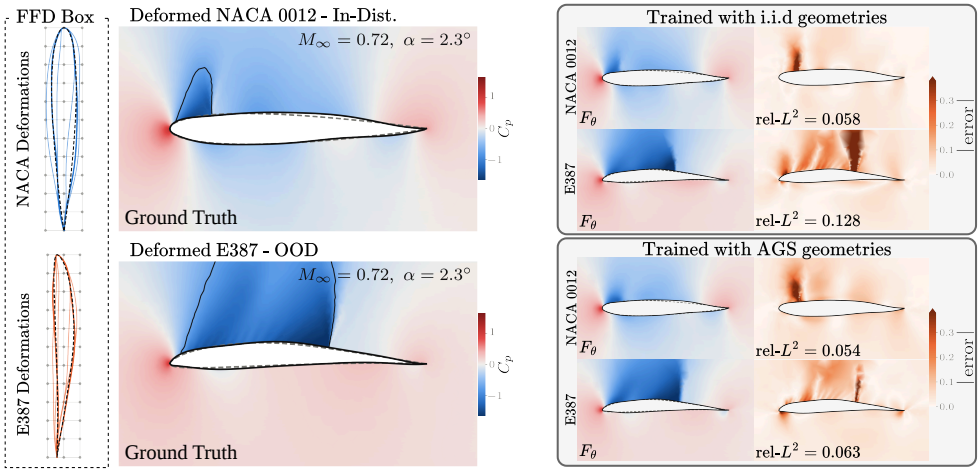
Training neural operators for PDEs over complex geometries requires high-quality simulation datasets that are often designed by careful parameterization of a few specific shapes, yielding geometries that cover a range of physical dynamics close to those of the base shapes. Still, as seemingly small changes in the geometry can induce widely different physical dynamics, neural operators trained with such data generalize poorly to unseen geometric features that may arise in engineering and design optimization. In this work, we propose **A**ctive **G**eometry **S**ynthesis (**AGS**), a novel geometric active learning method that improves the out-of-distribution (OOD) generalization of neural operators by synthesizing the geometries the operator is most uncertain about. Specifically, AGS equips a few base geometries with low-dimensional free-form deformations, and maximizes the predictive variance of the solution field, obtained by the Laplace approximation, with respect to the deformation control points using gradient ascent. As we demonstrate empirically, the Laplace predictive variance provides a surrogate for the operator errors over unseen shapes. As such, AGS samples diverse new geometries in regions of the geometry manifold where the physical dynamics are most challenging for the operator, expanding the coverage of the training dataset over the geometry manifold, and significantly improving the OOD generalization compared to i.i.d. uniform sampling. Meanwhile, by avoiding the manual collection and augmentation of geometries, AGS does not restrict the neural operator to a predetermined set of shapes.

1 INTRODUCTION

Scientific and engineering problems rely on solving partial differential equations (PDEs) for modelling various physical phenomena, such as fluid dynamics, wave propagation, and heat dissipation. While classical numerical solvers have long been studied for finding accurate and reliable solutions to PDEs, their computational cost prohibits rapid large-scale resolution of parameterized PDEs, especially when defined over complex geometries and boundary conditions that require careful meshing. Alternatively, neural operators have been developed as machine-learning surrogates, capable of fast and accurate approximation of parameterized PDE solutions (Kovachki et al., 2023). Pioneered by graph and Fourier neural operators (Li et al., 2020; 2021), neural operators made mesh-invariant, continuous physics surrogates possible by formulating the neural network as an operator that maps function spaces to function spaces, demonstrating promising accuracy and generalization for nonlinear PDEs. Recent works have further enabled neural operators to solve PDEs over complex domains and geometries by adopting attention blocks (Hao et al., 2023; Wu et al., 2024), efficiently scaling to large-scale 3D problems (Luo et al., 2025; Wen et al., 2026) and significantly reducing the computational costs associated with adaptive meshing and numerical solvers for such problems.

To train the neural operators over geometric domains, well-designed and high-quality numerical simulations are required in order to perform well when deployed. Such datasets are often designed by first predetermining the target geometric regime, and then manually collecting unique geometric objects (e.g. ShapeNet car simulations (Umetani & Bickel, 2018)) or parameterizing certain base shapes to generate augmented or deformed objects, as in airfoil simulations in Wen et al. (2026). Nevertheless, neural operators struggle to generalize to out-of-distribution (OOD) geometric features that may arise in engineering and design optimization problems, as small changes in geometric features can result in significantly different physical dynamics. See Figure 1a, for instance. When

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.



(a) Deformed NACA 0012 and E387 airfoils.

(b) Neural operator predictions.

Figure 1: (a) Airfoils and pressure fields sampled from NACA 0012 (training distribution) and E387 (OOD) base shapes by uniform FFD perturbations, with the same Angle of Attack α and Mach number M_∞. Note the difference in the shock pattern (dark blue region) between the two geometries. (b) Predictions from neural operators trained with i.i.d. uniform (top) and AGS (bottom) deformations of the NACA 0012 airfoil.

simulated under the same physical parameters, two airfoils generated by deforming NACA 0012 and E387[1] airfoils exhibit significantly different shock patterns (the dark blue region). Thus, when the neural operator is trained on i.i.d. uniform deformations of NACA 0012, it fails to generalize well to a new airfoil family (See Figure 1b, top panel). Such failures render neural operators unreliable in deployment regimes where novel geometric features appear frequently, unless strictly restricted to a predetermined geometric regime.

This limitation motivates us towards an active learning (AL) strategy where new geometries are sampled considering their induced physical dynamics and the uncertainty of the neural operator. In this work, we propose a novel geometric AL framework for neural operators, **A**ctive **G**eometry **S**ynthesis (AGS), where new geometries are synthesized by optimizing an acquisition function that targets a bound on worst-case error of the neural operator across the geometry manifold. Specifically, AGS first converts a deterministic neural operator to a function-valued Gaussian process (GP) by applying the Laplace approximation, following Magnani et al. (2025), resulting in a kernel that provides predictive variance over the solution fields. Since the kernel is induced by the trained operator, its variance grows larger in regions of the geometry manifold where the physical dynamics are most challenging for the operator. Then, to make the geometry optimizable, we equip a set of base shapes with differentiable free-form deformations (FFD) (Sederberg & Parry, 1986), reducing the search space to a small set of control points whose perturbations form a continuous manifold of geometries. Finally, we maximize the predictive variance over the FFD control points by gradient ascent, and synthesize new geometries greedily by conditioning the kernel to enforce diversity. By sampling the new geometries in regions of the geometry manifold where the physical dynamics are most challenging for the neural operator, AGS significantly improves the OOD generalization of the neural operator and expands the coverage of the training dataset over the manifold.

Few previous works have studied AL for neural operators with scalar PDE parameters or initial conditions and have shown promising results in terms of labelling budget compared to i.i.d. sampling (Musekamp et al., 2025; Polanska et al., 2026). Notably, Wang et al. (2026) introduced plug-and-play query synthesis (PaPQS), a synthesis-based AL method where parameters or initial conditions are synthesized by maximizing the expected information gain. However, none of the previous works study the complex geometric parameters of PDEs and, crucially, the OOD generalization across geometric distributions. Thus, their generated datasets closely match the uniform samples centred at

[1]NACA 0012 and E387 are standard airfoils obtained from the UIUC Applied Aerodynamics database.

the base geometries. AGS, on the other hand, does not assume an exact target geometric distribution and instead targets nearby geometries that are challenging for the operator.

2 BACKGROUND

2.1 OPERATOR LEARNING

Following the neural operator setup of Li et al. (2022), we are interested in learning mappings between infinite-dimensional function spaces governed by a system of PDEs. Specifically, for a spatial domain $\Omega \subset \mathbb{R}^d$, we are interested in approximating the operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ that maps an input function $a \in \mathcal{A}$ to a solution field $u \in \mathcal{U}$. To accommodate PDEs with diverse geometric features, we define the input function space as $\mathcal{A} = \mathcal{A}_p \times \mathcal{A}_g$, where $a_p \in \mathcal{A}_p$ represents physical quantities, such as PDE parameters or initial conditions, and $a_g \in \mathcal{A}_g$ represents the geometric features, such as boundary conditions. We write $a = (a_p, a_g)$ and take $\mathcal{G}$ to be realized by a numerical PDE solver (the oracle).

A neural operator F_θ parameterized by θ approximates $\mathcal{G}$ and is trained to the maximum a posteriori (MAP) weights $\theta^\star$ on a labelled dataset $\mathcal{D} = \{(a^i, u^i)\}_{i=1}^{N_{tr}}$ with $u^i = \mathcal{G}(a^i)$. In practice, the neural operator processes the discretized realization of a and queries the solution field on a grid $\{x_\ell\}_{\ell=1}^{L}$. Along with the operator formulation F_θ, we also use the *curried* form of the operator (Magnani et al., 2025), indexed by the coordinate and the output channel,

$$f_\theta(a, x, i) \;=\; F_\theta(a)(x)_i \;=\; \hat{u}_i(x) \;\in\; \mathbb{R}, \qquad i \in [c] = \{1, \dots, c\},$$

where $f_\theta : \mathcal{A} \times \Omega \times [c] \to \mathbb{R}$ returns the i-th channel of the predicted field at a coordinate $x \in \Omega$.

2.2 NEURAL OPERATORS AS FUNCTION-VALUED GAUSSIAN PROCESSES

Laplace approximation (LA) is a well-known approximation for the posteriors of neural networks (Daxberger et al., 2021). LA equips the trained operator with predictive uncertainty in two steps. First, it replaces the intractable posterior over the network weights with a Gaussian centred at the MAP estimate, $\theta \mid \mathcal{D} \sim \mathcal{N}(\theta^\star, \Sigma)$, so that uncertainty about the operator is expressed as uncertainty about its weights. Second, Magnani et al. (2025) push this weight posterior forward through the operator; *linearizing* f_θ about $\theta^\star$ propagates the Gaussian over weights to a Gaussian over output functions, yielding the operator's predictive distribution as a *function-valued Gaussian process* (GP). Since the curried operator is scalar-valued, the function-valued GP over $\mathcal{A}$ is equivalently a scalar-valued GP on the index set $\mathcal{A} \times \Omega \times [c]$ of triples (a, x, i). Writing $J(a, x, i) = \nabla_\theta f_\theta(a, x, i)\big|_{\theta^\star} \in \mathbb{R}^{1 \times M}$ for the Jacobian of the curried operator with respect to the M weights, the pushforward gives a predictive kernel that factorizes through a feature map Φ,

$$\bar{k}\big((a, x, i), (a', x', i')\big) = J(a, x, i)\, \Sigma\, J(a', x', i')^\top = \Phi(a, x, i)^\top \Phi(a', x', i'), \tag{1}$$

$$\Phi(a, x, i) = \Sigma^{1/2} J(a, x, i)^\top \in \mathbb{R}^M. \tag{2}$$

The predictive variance at a coordinate $x \in \Omega$ and channel $i \in [c]$ is

$$\sigma^2(a, x, i) = \|\Phi(a, x, i)\|_2^2. \tag{3}$$

Similarly, by stacking the features over the output grid and the channels as $\Phi(a) = \big[\Phi(a, x_\ell, i)\big]_{\ell \in [L],\, i \in [c]} \in \mathbb{R}^{M \times cL}$, the feature map induces a *physics-aware* distance function between input functions,

$$d_k(a, a')^2 = \big\|\Phi(a) - \Phi(a')\big\|_F^2. \tag{4}$$

We use the predictive variance and d_k in Section 3 to design our AL acquisition function.

Remark 1. Following Magnani et al. (2025), the posterior covariance Σ is calculated by a low-rank approximation to the Generalized Gauss-Newton (GGN) matrix (Daxberger et al., 2021; Dangel et al., 2023) and adding a validation-calibrated isotropic prior variance. Furthermore, the posterior distribution and Jacobians J are derived only for an active subset $\theta_{\text{act}} \in \mathbb{R}^M$ of all the model parameters θ. We drop the subscript from θ_{act} for brevity when clear. See Appendix B for a detailed description of the posterior derivation and the predictive distribution.

2.3 Free-Form Deformation and Geometric Distributions

Free-form Deformations. Our method synthesizes new geometries by performing gradient ascent on an acquisition function w.r.t. the geometry (elaborated in Section 3). Optimizing the geometry directly in its discretized form a_g (e.g. a point cloud) is poorly suited to this objective, as the gradient steps can easily drift into self-intersecting or otherwise degenerate shapes. Inspired by Allen et al. (2022), we instead parameterize the geometry with free-form deformation, which deforms a base shape through a small number of control points. For a base shape s, FFD gives a differentiable map

$$\gamma^{(s)} : \mathcal{Z}^{(s)} \to \mathcal{A}_g, \qquad a_g = \gamma^{(s)}(z),$$

from a bounded, low-dimensional design space $\mathcal{Z}^{(s)} \subset \mathbb{R}^{N_{\text{FFD}} \times d}$ of control point displacements z to the geometry input function a_g, deforming the base shape as the design z varies. The FFD perturbations are bounded to $[-z_{max}, +z_{max}]$ to avoid degenerate shapes, and we check for validity constraints to make sure the geometries are valid for the numerical solver (see Appendix C), rejecting the geometry otherwise. This parameterization supplies the two properties our acquisition relies on. First, $\gamma^{(s)}$ is smooth in z, so any objective defined through the operator is differentiable with respect to the design and can be optimized by gradient ascent. Second, the bounded box $\mathcal{Z}^{(s)}$ confines the deformations to a range that preserves valid, non-degenerate geometries.

Geometric Distributions. Both training and evaluation are described by distributions over geometries. A distribution π over control points in $\mathcal{Z}^{(s)}$ pushes forward through the FFD map to a geometric distribution $\mu = \gamma^{(s)}_{\#}\pi$ over shapes. We construct i.i.d. uniform datasets by randomly perturbing the control points for each base shape. Perturbations of a set of training bases S_{tr} define the training distribution μ_{tr}, a mixture over S_{tr}, while each held-out base $s \in S_{\text{ood}}$, disjoint from S_{tr}, defines an OOD test distribution $\mu^{(s)}_{\text{ood}} = \gamma^{(s)}_{\#}\pi^{(s)}$, which we evaluate separately. Our active learning method draws geometries from the same training bases S_{tr} but optimizes the FFD perturbations rather than randomly sampling them, such that both random sampling and this optimization are confined to the same reachable geometry manifold,

$$\mathcal{M} = \bigcup_{s \in S_{\text{tr}}} \gamma^{(s)}\big(\mathcal{Z}^{(s)}\big) \subseteq \mathcal{A}_g,$$

i.e. the set of all shapes FFD can produce from the training bases. Since S_{ood} is disjoint from S_{tr}, each $\mu^{(s)}_{\text{ood}}$ differs from μ_{tr}, a covariate shift under which neural operators typically degrade.

Remark 2. Ideally, the FFD family is expressive enough that the OOD distributions still lie on the reachable manifold, $\text{supp}(\mu^{(s)}_{\text{ood}}) \subseteq \mathcal{M}$, so the shift is a change of distribution over $\mathcal{M}$ rather than a move onto a disjoint set. As the deployment distribution is unknown in advance and may place its mass anywhere on $\mathcal{M}$, we seek uniform accuracy across $\mathcal{M}$ rather than accuracy tuned to μ_{tr}. However, we note that the support containment in $\mathcal{M}$ is a modelling assumption, and with a finite number of bounded FFD control points, a held-out shape may not be exactly reachable by deforming a training base.

3 Methodology

We first introduce the total predictive variance as a computable surrogate for the worst-case error of the neural operator over the geometry manifold $\mathcal{M}$, which motivates synthesizing geometries by variance maximization (Section 3.1). We then extend the objective to batch acquisition through kernel conditioning (Section 3.2), and present the full AGS algorithm (Section 3.3). Throughout this section, we fix the physical parameters at a reference value $a_p = \bar{a}_p \in \mathcal{A}_p$ and write $a = (\bar{a}_p, a_g)$ for the input function with geometry $a_g \in \mathcal{M}$, so that the quantities below vary with the geometry alone, and expand on the selection of a_p in Section 3.3. We set $\bar{a}_p$ to the midpoint of the parameter ranges.

3.1 Variance Maximization as a Worst-Case Surrogate

Worst-Case Error. Since small geometric deformations can induce significantly different physical dynamics and the exact geometric deployment setting is not foreseeable, we are interested in the

worst-case error of the neural operator over the geometries in the manifold $\mathcal{M}$,

$$\mathcal{E}(\mathcal{M}) \;=\; \sup_{a_g \in \mathcal{M}} \big\| F_{\theta^\star}(a) - \mathcal{G}(a) \big\|_F^2, \tag{5}$$

where the norm sums the squared error over the output grid and the channels. Evaluating $\mathcal{E}(\mathcal{M})$ requires the oracle $\mathcal{G}$ at every geometry in $\mathcal{M}$ and is therefore intractable.

Variance as a Worst-Case Surrogate. For a GP conditioned on noise-free observations of a function g in the reproducing kernel Hilbert space $\mathcal{H}$ of its kernel, the posterior mean m and the posterior variance σ^2 satisfy

$$\big(g(x) - m(x)\big)^2 \;\le\; \|g\|_{\mathcal{H}}^2 \, \sigma^2(x) \tag{6}$$

at every input x (Kanagawa et al., 2018, Corollary 3.11), where σ is known as the power function in kernel interpolation (De Marchi et al., 2005). We treat the linearized neural operator of Section 2.2 as such a GP with posterior mean $f_{\theta^\star}$, and assume that the residual $\mathcal{G} - f_{\theta^\star}$ is approximately captured by the linearized operator, which gives

$$\big(f_{\theta^\star}(a,x,i) - \mathcal{G}(a)(x)_i\big)^2 \;\lesssim\; C^2 \, \sigma^2(a,x,i), \tag{7}$$

with a constant C independent of (a,x,i). Summing over the query points and the channels and taking the supremum over $\mathcal{M}$ bounds the worst-case error of Eq. 5 by the worst-case total predictive variance,

$$\mathcal{E}(\mathcal{M}) \;\lesssim\; C^2 \sup_{a_g \in \mathcal{M}} v(a), \tag{8}$$

where $v(a) = \|\Phi(a)\|_F^2$ is the sum of the total predictive variance over all coordinates and channels.

Acquisition Objective. To reduce the right-hand side of Eq. 8, we synthesize the geometry that attains it by maximizing the total predictive variance over the FFD design z of a base shape $s \in \mathcal{S}_{tr}$,

$$z^\star \;=\; \arg\max_{z \in \mathcal{Z}^{(s)}} \; v\big(a(z)\big), \qquad a(z) \;=\; \big(\bar{a}_p,\, \gamma^{(s)}(z)\big). \tag{9}$$

Within an acquisition round, conditioning the GP on the labelled geometry $a(z^\star)$ sets its predictive variance to zero and cannot increase the variance at any other geometry (Appendix B.2), so each acquisition targets the largest term of Eq. 8 rather than the average error over $\mathcal{M}$. This greedy rule is the operator-valued analogue of P-greedy selection in kernel interpolation (De Marchi et al., 2005). AGS performs the acquisition by gradient ascent over a differentiable geometry parameterization.

Remark 3. Eq. 8 serves as a heuristic motivation for our acquisition objective rather than a guarantee. It relies on the neural operator approximately interpolating its training data and on the residual being captured by the linearized operator, the latter being least plausible far from the training data. Moreover, the operator is retrained after every acquisition round, which changes the kernel itself. We therefore verify empirically that the total predictive variance tracks the observed error of $F_{\theta^\star}$ (Section 4.3) and examine where it becomes overconfident (Appendix E.3).

3.2 Batch Synthesis via Kernel Conditioning

Synthesizing a batch of geometries by maximizing Eq. 9 tends to collapse the batch, as the objective is independent for different designs z and all the new geometries converge to the same maximizer of $v(a)$. To enforce diversity, we synthesize the new geometries one at a time and condition the predictive kernel $\bar{k}$ on the geometries acquired so far. Let $B_{b-1} = \{a^1, \dots, a^{b-1}\}$ be the geometries acquired in the current batch before the slot b. For a fixed linearization, conditioning the function-valued Gaussian process of Section 2.2 on B_{b-1} with noise-free labels gives the conditioned kernel $\bar{k}_{B_{b-1}}$, which admits the projection form

$$\bar{k}_{B_{b-1}}(a,a') \;=\; \Phi(a)^\top \big(I - Q_{b-1} Q_{b-1}^\top\big) \Phi(a'), \tag{10}$$

where Q_{b-1} is a matrix with orthonormal columns spanning the range of $\big[\Phi(a^1) \;\cdots\; \Phi(a^{b-1})\big]$, as we derive in Appendix B.2. This conditioning is a synthesis-based AL counterpart of the maximum-determinant batch selection of Holzmüller et al. (2023) and allows for conditioning before querying

Algorithm 1 Active Geometry Synthesis (AGS)

Require: Initial dataset $\mathcal{D}_0$, base shapes S_{tr}, oracle $\mathcal{G}$, batch size N_b, AL iterations N_{al}
for $i = 1$ to N_{al} **do**
  Train the neural operator F_θ on $\mathcal{D}_{i-1}$ to obtain MAP weights $\theta^\star$
  Apply LA to obtain the posterior Σ over $\theta^\star_{act}$ and the kernel $\bar{k}$
  Select N_b seeds $\{a_b^{\text{seed}}\}_{b=1}^{N_b}$ from $\mathcal{D}_{i-1}$ by variance and farthest-point traversal in d_k
  $Q_0 \leftarrow [\,]$
  for $b = 1$ to N_b **do**
    $z_b^\star \leftarrow$ gradient ascent on Eq. 12, initialized at the design of a_b^{seed}
    $\bar{a}_b \leftarrow (\bar{a}_p, \gamma^{(s)}(z_b^\star))$, base shape $s \in S_{\text{tr}}$ determined by a_b^{seed}
    $Q_b \leftarrow$ extend Q_{b-1} with the new directions of $\Phi(\bar{a}_b)$
  end for
  $a^\star_{p,b} \leftarrow$ sample uniformly from $\mathcal{A}_p$ for all b
  $a_b \leftarrow (a^\star_{p,b}, \gamma^{(s)}(z_b^\star))$ for all b
  $\mathcal{D}_i \leftarrow \mathcal{D}_{i-1} \cup \{(a_b, \mathcal{G}(a_b))\}_{b=1}^{N_b}$
end for
return $\mathcal{D}_{N_{al}}$ and $F_{\theta^\star}$ trained on $\mathcal{D}_{N_{al}}$

the oracle. Taking the trace of Eq. 10 at $a' = a$, the conditioned total predictive variance of a candidate is

$$v_{B_{b-1}}(a) \;=\; \big\|\big(I - Q_{b-1}Q_{b-1}^\top\big)\Phi(a)\big\|_F^2, \tag{11}$$

which measures the part of the predictive variance of a not already explained by the acquired geometries in B_{b-1}. For an empty B_0, $v_{B_0}(a)$ reduces to $v(a)$, and a candidate that duplicates an acquired geometry receives a score of zero, so duplicates are eliminated rather than only discouraged. We elaborate on the construction of Q in Appendix B.3.

3.3 Seeding, Trust Region, and the AGS Algorithm

We now present the full AGS algorithm. In each AL iteration $i \in [1, \ldots, N_{al}]$ and starting with an initial dataset $\mathcal{D}_0$, we train the neural operator on $\mathcal{D}_{i-1}$ and apply the Laplace approximation to find the posterior and $\bar{k}$, select N_b seed geometries from $\mathcal{D}_{i-1}$, and synthesize N_b new geometries by maximizing the conditioned variance from each seed.

Seeding the Design Variables. The conditioned variance in Eq. 11 is non-convex in the design z, so the geometry that the gradient ascent converges to depends on its starting point. We therefore seed each batch slot with the design of an existing geometry in $\mathcal{D}_{i-1}$, which places the ascent in a region of $\mathcal{M}$ already known to be informative. We shortlist the geometries in $\mathcal{D}_{i-1}$ with the highest variance $v(a)$, and select N_b seeds from the shortlist by farthest-point traversal in the distance d_k of Eq. 4. Since high-variance geometries are typically clustered, scoring alone would select seeds from a single region of $\mathcal{M}$, whereas the traversal yields seeds that are both high-variance and spread out in d_k. In all our experiments, the shortlist size is set to $4 \times N_b$.

Trust Region. We regularize the acquisition by penalizing the drift of the geometry from its initial seed, measured by d_k. While this regularization may seem contradictory to the variance-maximization objective, it serves two main goals. First, note that the predictive variance given by the Laplace approximation is reliable only when the oracle remains a weight perturbation away from the trained neural operator, which is least likely to hold for geometries far from $\mathcal{D}_{i-1}$. Thus, distance regularization keeps the posterior reliable. Second, since the FFD perturbations are bounded, the variance-maximization objective may guide the geometry towards degenerate or unreachable geometries outside the FFD box, as such geometries can induce high variance. Since we project the design onto the bounded box after each step, unregularized synthesis results in geometries that saturate the boundary of the FFD box, harming the diversity.

The AGS Acquisition Function. Combining the conditioned variance of Eq. 11 with the trust region, the design of the batch slot b is synthesized by solving

$$z_b^\star \;=\; \underset{z \in \mathcal{Z}^{(s)}}{\arg\max}\; v_{B_{b-1}}\big(a(z)\big) \;-\; \mu\, d_k\big(a(z),\, a_b^{\text{seed}}\big)^2, \tag{12}$$

Table 1: Relative errors of the predicted field and of the drag coefficient integrated from its surface pressure. All models are trained with $N_{tr} = 2048$ ($N_{al} = 15$). Reported are the means over three training seeds.

	In-distribution	RAE 2822	E387	FX 63-137	MH 113
	Volumetric rel-L^2 error				
i.i.d. uniform	0.037	**0.079**	0.130	0.244	0.246
PaPQS	**0.035**	0.083	0.127	0.231	0.243
AGS (ours)	0.047	0.088	**0.102**	**0.147**	**0.148**
	Drag coefficient rel. $\lvert C_D \rvert$ error				
i.i.d. uniform	0.051	0.125	0.297	0.665	0.648
PaPQS	**0.041**	**0.103**	0.260	0.559	0.513
AGS (ours)	0.066	0.106	**0.141**	**0.264**	**0.183**

which reduces to Eq. 9 for the first slot with $\mu = 0$. We solve Eq. 12 by gradient ascent on z from the design of the seed a_b^{seed}, projecting z back onto $\mathcal{Z}^{(s)}$ after each step to keep the geometries valid. We then set $\bar{a}_b = (\bar{a}_p, \gamma^{(s)}(z_b^\star))$, extend the basis Q_{b-1} with the new directions of $\Phi(\bar{a}_b)$ to obtain Q_b, and proceed to the next slot. Note that, so far, we used the fixed $\bar{a}_p$ for the non-geometric parameters. Once all N_b geometries are synthesized, we randomly sample parameters $a_p^\star \in \mathcal{A}_p$ for each synthesized geometry, label them with the oracle $\mathcal{G}$ and append them to the dataset. Algorithm 1 summarizes the AGS loop.

4 Experiments

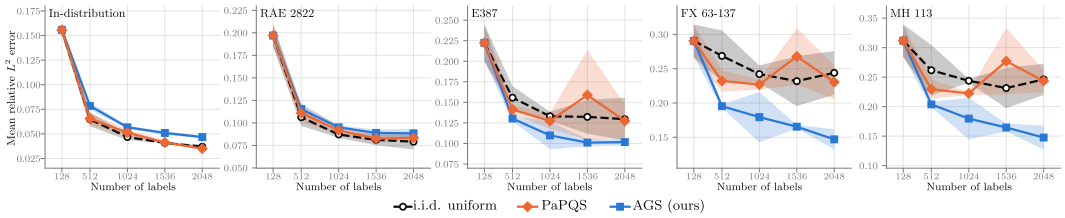
4.1 Experiment Setting

In order to evaluate AGS, we are interested in problems where changes in geometry can produce widely different physical dynamics. As such, we choose two problems with this property: 2D Euler simulations on airfoils (**Airfoil**) and a bistable reaction-diffusion equation with a geometric source term (**RD**). In both problems, geometries can change the induced dynamics, and a model with poor OOD generalization fails. In addition to the geometry, the airfoil simulations are also parameterized by their Mach number and Angle of Attack. Similarly, reaction-diffusion simulations are parameterized by the source amplitude and the reaction threshold. Non-geometric parameters are sampled uniformly for labelling. See Appendix C for a detailed description of the datasets.

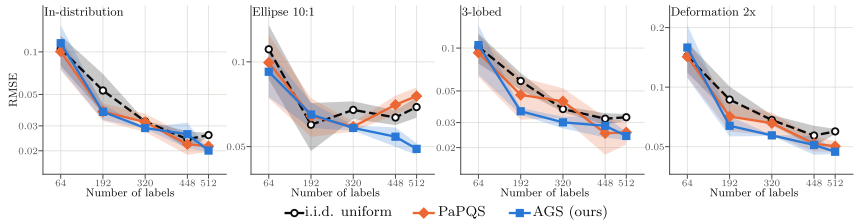
As baselines, we consider i.i.d. uniform geometries (Uniform) and a recent synthesis-based AL method, PaPQS Wang et al. (2026), extended to the same FFD geometry parameterization. Note that for both Airfoil and RD, the geometry manifold $\mathcal{M}$ is identical across AGS and both baselines, so the reachable geometries are the same and only the density differs. Throughout this section, we use Transolver (Wu et al., 2024) as the neural operator backbone. Finally, for both AGS and PaPQS, we first acquire the datasets by performing N_{al} acquisition iterations, and then retrain the neural operators with three different random initializations for all three of the AGS, PaPQS, and Uniform datasets, and report the mean errors. We applied early stopping on an in-distribution validation dataset for all models. See Appendix D for the detailed experiment settings.

4.2 In-distribution and OOD Generalization

Airfoil. For airfoil simulations, we use NACA 0012 as the single base geometry for the training distribution, and evaluate the OOD generalization on uniform FFD perturbations of E387, MH 113, RAE 2822, and FX 63137 airfoils. We run AGS with up to $N_{al} = 15$ acquisition iterations and an acquisition batch size of $N_b = 128$, resulting in a total of $N_{tr} = 2048$ samples when starting from an initial dataset of 128 uniform samples. We report the mean relative squared error over the volumetric velocity and pressure errors (rel-L^2), as well as the relative absolute error for the Drag coefficient (rel. $|C_D|$), which, in our inviscid setting, mainly measures how well the shock is captured.



(a) Airfoil



(b) Reaction-Diffusion

Figure 2: Airfoil (top) and Reaction-Diffusion (bottom) in-distribution and OOD errors against the number of labelled samples.

Figure 2a illustrates the relative errors against the number of training samples for each distribution, and Table 1 reports the volumetric and Drag errors for $N_{tr} = 2048$ samples. We first note that, while following closely and consistently, AGS is outperformed by both PaPQS and Uniform on the in-distribution test set, while PaPQS achieves the lowest error. However, as reported in Table 1, AGS improves volumetric errors by up to 40% and the C_D error by 50% to 70% for the OOD airfoil families at $N_{tr} = 2048$. The only exception is the RAE 2822 OOD family, where all methods achieve comparable performance; an observation we return to in Section 4.3. For all airfoil families, Figure 2a shows a consistent reduction in error with AGS as the number of labelled samples increases, while Uniform and PaPQS plateau for OOD families.

Reaction-Diffusion. We consider reaction-diffusion PDEs with a sustained geometric forcing term solved by the pseudo-spectral solver Exponax Koehler et al. (2024). We study a setting under which the radius and shape of the forcing determine the igniting or quenching physical regimes. We start with $N_{tr} = 64$ samples and perform $N_{al} = 7$ AL iterations with $N_b = 64$, reaching up to $N_{tr} = 512$ samples. We sample the training geometries from two base shapes, a circle and an ellipse, and evaluate the OOD generalization on deformed ellipses (with a different aspect ratio than training), three-lobed stars, and a larger ($2\times$) FFD deformation box over the training base shapes. We note that 74 samples synthesized by AGS were rejected and replaced with uniform samples, as elaborated in Appendix C.2. As illustrated in Figure 2b, AGS consistently reduces in-distribution and OOD errors with increasing number of labels. At $N_{tr} = 512$, AGS outperforms the baselines on all four evaluation sets, with PaPQS following closely. In the OOD Ellipse distribution, however, both PaPQS and Uniform degrade after passing $N_{tr} = 192$, and AGS outperforms both by a significant margin.

4.3 Distance to OOD and Manifold Coverage

To better understand the effect of AGS and the Laplace variance on the training distribution, we examine the training and OOD datasets. To this end, we apply the Laplace approximation to the neural operator trained with the i.i.d. dataset $\mathcal{D}_{\text{Uniform}}$ to obtain the kernel distance d_k^{uni}, and measure the nearest-neighbour distance between the OOD airfoils and all three training datasets (i.e. Uniform, PaPQS, and AGS datasets). Figure 3a depicts the distances and the spread of each family. Naturally, for all training datasets, the in-distribution NACA 0012 airfoils sit closest to the training dataset and form the baseline density (dashed lines). We observe that AGS has the lowest baseline density (largest distance) in-distribution compared to the baselines. Furthermore, RAE 2822 is the closest OOD family to the training distribution, where AGS has the lowest density again. These two observations are in line with the higher errors from AGS for NACA 0012. However, for E387, FX63137, and MH113 airfoils, the distances to the PaPQS and Uniform datasets grow rapidly, which

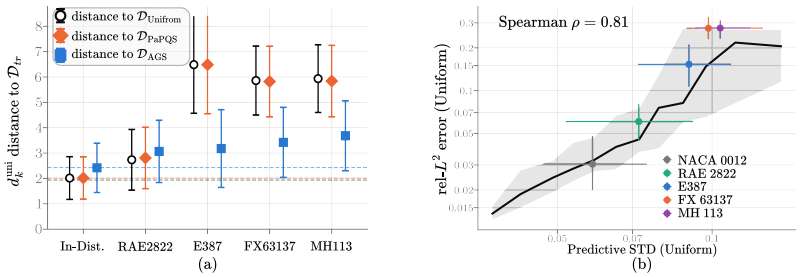


Figure 3: (a) Nearest-neighbour kernel distance from each airfoil family to each training dataset. (b) Predictive standard deviation and relative errors from the Uniform model. Solid line is the median over bins.

aligns with their high generalization error. In contrast, all three OOD families remain much closer to the AGS dataset, and their distances grow gracefully, consistent with the superior generalization of AGS over those families. The smaller distance to the AGS dataset shows how variance maximization expands the dataset coverage within the geometry manifold.

Additionally, as discussed in Section 3.1, variance serves as a surrogate for the error. To verify this, we again use the predictive variance from the Uniform model and measure the correlation between the errors and the standard deviations. As illustrated in Figure 3b, the standard deviation increases monotonically with the error, achieving a Spearman's rank correlation of 0.81, until saturating at the two farthest airfoil families. The monotonic correlation at the in-distribution and RAE 2822 airfoils shows how the Laplace kernel with the linearized neural operator can serve as a surrogate for the errors. However, as d_k^{uni} is induced by training on $\mathcal{D}_{\text{Uniform}}$ only, the Laplace variance becomes overconfident, and the correlation stops. This observation is in line with Mouli et al. (2024), where Laplace approximation is shown to be prone to overconfidence in OOD samples. As such, by applying the trust-region regularization and incrementally expanding the coverage, AGS still performs well on the distant OOD families.

We further investigate the Laplace variance and the overconfidence at the solution field level in Appendix E.3. In Appendix E, we also report AGS results when trained with GNOT (Hao et al., 2023) as the backbone neural operator, and examine the transferability of the data across the architectures.

5 Conclusion

Motivated by the large variations in the physical dynamics induced by changes in geometric features, we introduced Active Geometry Synthesis, a synthesis-based active learning framework where new geometries are synthesized by maximizing their predictive variance, resulting in significantly improved generalization to out-of-distribution geometries. By applying Laplace approximation to the linearized neural operator, we obtained the variances for deterministic neural operators, and by parameterizing geometries using differentiable free-form deformations, AGS synthesized new geometries via gradient ascent. We empirically showed that the Laplace kernel provides a reliable surrogate for erroneous regions near the training distribution, and the kernel conditioning together with the trust-region regularization yields diverse geometries that consistently improve OOD generalization with increasing label budget.

Despite the promising results, we acknowledge the necessary considerations and limitations of AGS. First, AGS may degrade the in-distribution performance and thus is most favourable in settings where the deployment geometric distribution cannot be exactly predetermined and geometric variations meaningfully change the physical dynamics. Second, we note that while free-form deformation is flexible, it provides limited prior over the plausible geometries and may need careful treatment to avoid degenerate geometries. Furthermore, although trust-region regularization and incremental expansion of the training distribution help, Laplace approximation becomes increasingly overconfident farther from the training distribution and may fail to indicate the errors. Finally, we acknowledge that the additional computational costs of AGS are not negligible. While these costs are justifiable when the numerical simulations are expensive, the extension to large-scale problems may still require further optimization.

AI USE STATEMENT

In this work, we used generative AI tools for help with formulating mathematical claims, providing ingredients for proving mathematical claims, assisting in writing proofs, providing feedback on methodology, and implementation. We have not used generative AI tools to develop theoretical or conceptual models, generate synthetic datasets, propose or refine hypotheses, interpret results, or support qualitative and thematic data analysis, and assistance with translation is not applicable to this work. Additionally, we used generative AI tools to assist in writing code, brainstorming, and polishing the writing. We have reviewed all AI-assisted work. We have checked the LLM-assisted code and mathematical formulations for correctness and have manually reviewed the literature. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Kelsey Allen, Tatiana Lopez-Guevara, Kimberly L Stachenfeld, Alvaro Sanchez Gonzalez, Peter Battaglia, Jessica B Hamrick, and Tobias Pfaff. Inverse design for fluid-structure interactions using graph network simulators. *Advances in neural information processing systems*, 35:13759–13774, 2022.

Joan Bruna, Benjamin Peherstorfer, and Eric Vanden-Eijnden. Neural galerkin schemes with active learning for high-dimensional evolution equations. *Journal of Computational Physics*, 496: 112588, 2024. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2023.112588. URL https://www.sciencedirect.com/science/article/pii/S0021999123006836.

Christopher Bülte, Philipp Scholl, and Gitta Kutyniok. Probabilistic neural operators for functional uncertainty quantification. *arXiv preprint arXiv:2502.12902*, 2025.

Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34:24924–24940, 2021.

Felix Dangel, Lukas Tatzel, and Philipp Hennig. Vivit: Curvature access through the generalized gauss-newton's low-rank structure. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=DzJ7JfPXkE.

Nima Hosseini Dashtbayaz, Ghazal Farhani, Boyu Wang, and Charles X Ling. Physics-informed neural networks: Minimizing residual loss with wide networks and effective activations. *arXiv preprint arXiv:2405.01680*, 2024.

Nima Hosseini Dashtbayaz, Hesam Salehipour, Adrian Butscher, and Nigel Morris. Physics-informed reduced order modeling of time-dependent pdes via differentiable solvers. *arXiv preprint arXiv:2505.14595*, 2025.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in neural information processing systems*, 34:20089–20103, 2021.

Stefano De Marchi, Robert Schaback, and Holger Wendland. Near-optimal data-independent point locations for radial basis function interpolation. *Advances in Computational Mathematics*, 23(3): 317–330, 2005.

Thomas D Economon, Francisco Palacios, Sean R Copeland, Trent W Lukaczyk, and Juan J Alonso. Su2: An open-source suite for multiphysics simulation and design. *Aiaa Journal*, 54(3):828–846, 2016.

Ghazal Farhani, Nima Hosseini Dashtbayaz, Alexander Kazachek, and Boyu Wang. A simple remedy for failure modes in physics informed neural networks. *Neural Networks*, 183:106963, 2025.

Wenhan Gao and Chunmei Wang. Active learning based sampling for high-dimensional nonlinear partial differential equations. *Journal of Computational Physics*, 475:111848, 2023. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2022.111848. URL https://www.sciencedirect.com/science/article/pii/S0021999122009111.

Shailesh Garg and Souvik Chakraborty. Variational bayes deep operator network: a data-driven bayesian solver for parametric differential equations. *arXiv preprint arXiv:2206.05655*, 2022.

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.

Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. *Physics-Informed Deep Neural Operator Networks*, pp. 219–254. Springer International Publishing, 2023. doi: 10.1007/978-3-031-36644-4_6. URL https://link.springer.com/content/pdf/10.1007/978-3-031-36644-4_6.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53 (2):217–288, 2011.

Zhongkai Hao, Chengyang Ying, Zhengyi Wang, Hang Su, Yinpeng Dong, Songming Liu, Ze Cheng, Jun Zhu, and Jian Song. Gnot: A general neural operator transformer for operator learning. *International Conference on Machine Learning*, 2023. doi: 10.48550/arXiv.2302.14376. URL https://arxiv.org/abs/2302.14376v3.

David Holzmüller, Viktor Zaverkin, Johannes Kästner, and Ingo Steinwart. A framework and benchmark for deep batch active learning for regression. *Journal of Machine Learning Research*, 24 (164):1–81, 2023.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of bayesian neural nets via local linearization. In *International conference on artificial intelligence and statistics*, pp. 703–711. PMLR, 2021.

Minseok Jeong, Giup Seo, and Euiseok Hwang. Mitigating instability in high residual adaptive sampling for PINNs via langevin dynamics. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=hq2CkcEY7h.

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

Yegon Kim, Hyunsu Kim, Gyeonghoon Ko, and Juho Lee. Active learning with selective time-step acquisition for pdes. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/kim25m.html.

Felix Koehler, Simon Niedermayr, Rudiger Westermann, and Nils Thuerey. APEBench: A benchmark for autoregressive neural emulators of PDEs. *Advances in Neural Information Processing Systems (NeurIPS)*, 38, 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.

Shibo Li, Xin Yu, Wei Xing, Robert Kirby, Akil Narayan, and Shandian Zhe. Multi-resolution active learning of Fourier neural operators. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 2440–2448. PMLR, 02-04 May 2024. URL https://proceedings.mlr.press/v238/li24k.html.

Zong-Yi Li, Nikola B. Kovachki, K. Azizzadenesheli, Burigede Liu, K. Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *International Conference on Learning Representations*, 2020.

Zong-Yi Li, D. Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of machine learning research*, 2022. doi: 10.5555/3648699.3649087. URL https://arxiv.org/abs/2207.05209v2.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d PDEs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=86dXbqT5Ua.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in neural information processing systems*, 32, 2019.

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

Emilia Magnani, Marvin Pförtner, Tobias Weber, and Philipp Hennig. Linearization turns neural operators into function-valued gaussian processes. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=4Z04wVQ9FY.

S Chandra Mouli, Danielle C Maddix, Shima Alizadeh, Gaurav Gupta, Andrew Stuart, Michael W Mahoney, and Yuyang Wang. Using uncertainty quantification to characterize and improve out-of-domain learning for pdes. *arXiv preprint arXiv:2403.10642*, 2024.

Daniel Musekamp, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert. Active learning for neural pde solvers. In *International Conference on Learning Representations*, volume 2025, pp. 38777–38825, 2025.

Houman Owhadi. Do ideas have shape? idea registration as the continuous limit of artificial neural networks. *Physica D: Nonlinear Phenomena*, 444:133592, 2023. ISSN 0167-2789. doi: https://doi.org/10.1016/j.physd.2022.133592. URL https://www.sciencedirect.com/science/article/pii/S0167278922002962.

Alicja Polanska, Lorenzo Zanisi, Vignesh Gopakumar, and Stanislas Pamela. Data-efficient neural operator training via physics-based active learning. *arXiv preprint arXiv:2605.21348*, 2026.

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

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pp. 151–160, 1986.

Louis Serrano, Lise Le Boudec, Armand Kassaï Koupaï, Thomas X Wang, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Operator learning with neural fields: Tackling pdes on general geometries. *Advances in Neural Information Processing Systems*, 36:70581–70611, 2023.

Shion Takeno, Yoshito Okura, Yu Inatsu, Aoyama Tatsuya, Tomonari Tanaka, Akahane Satoshi, Hiroyuki Hanada, Noriaki Hashimoto, Taro Murayama, Hanju Lee, Shinya Kojima, and Ichiro Takeuchi. Distributionally robust active learning for gaussian process regression. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=tVDZlyVMIZ.

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

Zhiyuan Wang, Jinwoo Go, Byung-Jun Yoon, Nathan Urban, and Xiaoning Qian. A plug-and-play query synthesis active learning framework for neural pde solvers. *Advances in Neural Information Processing Systems*, 38:73385–73420, 2026.

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. *Advances in Neural Information Processing Systems*, 38:155423–155501, 2026.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Ywl6pODXjB.

Yuan Yin, Matthieu Kirchmeyer, Jean-Yves Franceschi, Alain Rakotomamonjy, and patrick gallinari. Continuous PDE dynamics forecasting with implicit neural representations. In *The Eleventh International Conference on Learning Representations*, 2023.

APPENDIX

A RELATED WORKS

A.1 NEURAL OPERATORS

In recent years, machine learning methods for physics have gained increasing momentum following the introduction of physics-informed neural networks (PINNs) (Raissi et al., 2019; Dashtbayaz et al., 2024; Farhani et al., 2025) and neural operators (Kovachki et al., 2023; Li et al., 2022; Lu et al., 2019). While PINNs traditionally learn specific solutions to a specific PDE, neural operators learn mappings between function spaces and can solve parameterized PDEs. The pioneering Fourier neural operator (Li et al., 2021) introduced a global kernel convolution with the Fourier basis to solve PDEs on regular grids. Later works extended Fourier neural operators to irregular grids and complex domains by learning deformations (Li et al., 2022) and using graph neural networks (Li et al., 2023). Simultaneously, DeepONets used branch and trunk networks for solving parameterized PDEs (Lu et al., 2019) and introduced physics-informed neural operators (Goswami et al., 2023). Later works extended neural operators beyond Fourier blocks by adopting Implicit Neural Representations (Yin et al., 2023; Dashtbayaz et al., 2025; Serrano et al., 2023), and most notably, attention blocks (Cao, 2021; Hao et al., 2023; Wu et al., 2024). Specifically, GNOT (Hao et al., 2023) introduced heterogeneous attention blocks that can process multiple input functions across different domains, including geometric representations (e.g., point clouds). Transolver (Wu et al., 2024) further introduced learnable slices that split the discretized spatial domain into flexible shapes

with similar physics. More recent attention-based methods have further improved the efficiency of neural operators in processing complex geometric domains (Wu et al., 2024; Wen et al., 2026).

A.2 Uncertainty Quantification

With the advancement of neural operators, recent works have studied uncertainty quantification and the predictive distributions for neural operators. Early works proposed extensions of various Bayesian approaches to certain neural operators. Variational Bayes DeepONets (Garg & Chakraborty, 2022) learned distributions of weights, Magnani et al. (2022) applied Laplace approximation, and Mouli et al. (2024) used ensemble models. Overall, many common Bayesian methods are shown to be effective for neural operators (Psaros et al., 2023). Some recent works have also extended neural operators with generative modelling, mainly for chaotic problems, and learn distributions over the solution function space Bülte et al. (2025). In this work, we specifically use LUNO Magnani et al. (2025), which extends the Laplace approximations in Magnani et al. (2022) to the general function-mapping formulation of neural operators, and obtains a function-valued Gaussian process (Owhadi, 2023). Furthermore, LUNO adopts low-rank approximations (Dangel et al., 2023; Daxberger et al., 2021), last-layer posterior (Daxberger et al., 2021), and linearization about MAP weights (Immer et al., 2021). While we found LUNO to be effective for geometry synthesis, we note that the Laplace kernel is inherently a local approximation of the posterior and is prone to overconfidence. Mouli et al. (2024) specifically demonstrates this behaviour for OOD samples in neural operators, and we also discuss this in Section 4.3 and Appendix E.3.

A.3 Active Learning

Active learning (AL) methods have been studied in various areas of machine learning for physics. Adaptive sampling methods in PINNs use AL for faster convergence and mitigating failure modes (Gao & Wang, 2023; Bruna et al., 2024; Jeong et al., 2025). AL has also been used in neural operators for selective time stepping (Kim et al., 2025), multi-resolution training (Li et al., 2024), and, notably, solving parameterized PDEs Musekamp et al. (2025); Wang et al. (2026); Polanska et al. (2026). Specifically, Musekamp et al. (2025) introduced AL4PDE, a benchmark of common pool-based batch AL methods (Holzmüller et al., 2023) for neural operators, and demonstrated the effectiveness of AL for sampling scalar PDE parameters and initial conditions. Plug-and-play query synthesis (PaPQS) extended AL4PDE by introducing synthesis-based AL for PDE parameters. PaPQS used SWAG (Maddox et al., 2019) for posterior estimation and maximized an expected Information gain (EIG) objective via gradient ascent to synthesize new scalar parameters and initial conditions, improving the label efficiency compared to AL4PDE baselines.

However, none of the previous works studied geometries as a target parameter and mainly focused on in-distribution generalization. In this work, motivated by the PaPQS synthesis-based approach, we used LUNO posterior estimation and the predictive variance as the target objective specifically for geometry synthesis. Furthermore, we focus on out-of-distribution generalization to new geometric families, as predetermining the exact geometric setting in applications like design optimization is very restrictive. As such, a synthesis-based approach with a prior on the plausible geometries can improve the OOD generalization, while alleviating the costs associated with collecting geometric datasets. Closely related to our work, distributionally robust AL (Takeno et al., 2025) studies AL for GP regression tasks where specifying the target distribution is challenging, and reduces the expected worst-case error.

B Extended Methodology

B.1 Laplace Approximation

We follow the LUNO framework Magnani et al. (2025), where the posterior obtained by the Laplace approximation is pushed forward through the linearized (about the MAP weights) neural operator in the curried formulation to obtain function-valued Gaussian processes. LUNO's posterior Σ is given by a low-rank approximation to the Generalized Gauss-Newton (GGN) matrix (Dangel et al., 2023), which itself is a positive semi-definite approximation to the Hessians of the model. Similarly, we compute GGN approximations and form the posterior only for a subset θ_{act} of all model parameters,

which is comprised of Transolver's final attention block and linear projection layer in our experiments ($M = 85{,}323$ for Airfoil and $M = 49{,}019$ for RD). With the GGN matrix G, the posterior is then a Gaussian centred at MAP weights $\theta^\star_{\text{act}}$ with covariance Σ defined as

$$\Sigma = (G + \tau I)^{-1},$$

where τI is the isotropic prior precision. We recompute G over the current training dataset at every AL iteration, and calibrate τ by minimizing the negative log-likelihood over the validation samples as in LUNO.

In practice, the $M \times M$ matrix G is never formed, and we only access it through GGN-vector products implemented in JAX. We approximate $G \approx U\Lambda U^\top$ by its top-r eigenpairs, computed by randomized subspace iteration (Halko et al., 2011), with $r = 512$ for Airfoil and $r = 256$ for RD. The posterior then has the closed form $\Sigma = U(\Lambda + \tau I)^{-1}U^\top + \tau^{-1}(I - UU^\top)$, i.e. the Laplace variance $(\lambda_j + \tau)^{-1}$ along the eigenvectors of G and the prior variance τ^{-1} along every other direction, and its square root $\Sigma^{1/2}$ follows by replacing each variance with its square root. Thus, neither Σ nor $\Sigma^{1/2}$ is formed or inverted explicitly, and only U and Λ are stored.

By linearizing the neural operator, LUNO pushes the posterior forward to derive the predictive distribution with mean $f_{\theta^\star}$ and covariance given by the kernel $\bar{k}$ of Eq. 1. For the acquisition, we realize the feature block $\Phi(a) = \Sigma^{1/2}J(a)^\top \in \mathbb{R}^{M\times cL}$ densely, computing $J(a)^\top$ by one vector-Jacobian product per output coordinate, whereas the kernel blocks $\bar{k}(a, a') \in \mathbb{R}^{cL\times cL}$ are never formed, as the variance, d_k, and the conditioned variance are all computed from the feature blocks directly. Since the gradient ascent differentiates through $\Phi(a(z))$ at every step, its $O(McL)$ memory and cL vector-Jacobian products dominate the cost of AGS, which is why the acquisition evaluates the variance on a coarser set of query points than the one used for training (Appendix C).

B.2 Derivation of the Conditioned Kernel

In this section, we derive the projection form of the conditioned kernel in Eq. 10, which the batch acquisition of Section 3.2 uses to enforce diversity within a batch.

Let $B = \{a^1, \dots, a^b\}$ be the geometries acquired so far in the current batch, and let

$$\Phi_B \;=\; \big[\Phi(a^1) \;\cdots\; \Phi(a^b)\big] \tag{13}$$

be the matrix stacking their feature blocks. For a Gaussian process with kernel $\bar{k}$ and noise-free observations, conditioning on the observations at B_{b-1} gives the conditioned kernel

$$\bar{k}_B(a, a') \;=\; \bar{k}(a, a') \;-\; \bar{k}(a, B)\,\bar{k}(B, B)^\dagger\,\bar{k}(B, a'), \tag{14}$$

where $\bar{k}(a, B)$ and $\bar{k}(B, B)$ are the cross- and within-set kernel blocks, and $\bar{k}(B, B)^\dagger$ is the pseudo-inverse. We use the pseudo-inverse rather than the inverse, as the block $\bar{k}(B, B)$ is singular whenever the number of stacked output coordinates exceeds the rank of Φ_B, which is the typical situation in our setting. The following projection form is a standard consequence of noise-free GP conditioning (Williams & Rasmussen, 2006) and of the Moore–Penrose projector identity. We include the proof for completeness.

Lemma B.1 (Projection form of the conditioned kernel)**.** *Let Q be any matrix with orthonormal columns spanning the range of Φ_B. Then the conditioned kernel of Eq. 14 satisfies*

$$\bar{k}_B(a, a') \;=\; \Phi(a)^\top\big(I - QQ^\top\big)\Phi(a'). \tag{15}$$

Proof. By the factorization of the predictive kernel in Eq. 1, the kernel blocks in Eq. 14 are $\bar{k}(a, a') = \Phi(a)^\top\Phi(a')$, $\bar{k}(a, B) = \Phi(a)^\top\Phi_B$, and $\bar{k}(B, B) = \Phi_B^\top\Phi_B$. Substituting the three blocks into Eq. 14 gives

$$\bar{k}_B(a, a') \;=\; \Phi(a)^\top\Phi(a') - \Phi(a)^\top\Phi_B\big(\Phi_B^\top\Phi_B\big)^\dagger\Phi_B^\top\Phi(a') \;=\; \Phi(a)^\top\Big(I - \Phi_B\big(\Phi_B^\top\Phi_B\big)^\dagger\Phi_B^\top\Big)\Phi(a'). \tag{16}$$

The matrix $\Phi_B\big(\Phi_B^\top\Phi_B\big)^\dagger\Phi_B^\top$ is the orthogonal projector onto the range of Φ_B, and an orthogonal projector is determined by the subspace it projects onto. Since the columns of Q are orthonormal and span the range of Φ_B, the projector equals $QQ^\top$, which gives Eq. 15. □

Algorithm 2 Seeding of geometries

Require: training geometries $\mathcal{D}_{i-1}$, batch size N_b, shortlist size S
1: $\mathcal{S} \leftarrow$ the S geometries $a_g \in \mathcal{D}_{i-1}$ with the largest $v(a_g)$
2: $\mathcal{A} \leftarrow \{\arg\max_{a_g \in \mathcal{S}} v(a_g)\}$
3: **while** $|\mathcal{A}| < N_b$ **do**
4: $\quad \mathcal{A} \leftarrow \mathcal{A} \cup \{ \arg\max_{a_g \in \mathcal{S} \setminus \mathcal{A}} \min_{a'_g \in \mathcal{A}} d_k(a_g, a'_g) \}$
5: **end while**
6: **return** $\mathcal{A}$

Taking the trace of Eq. 15 at $a' = a$ gives the conditioned total predictive variance of Eq. 11,

$$v_B(a) \;=\; \operatorname{tr} \bar{k}_B(a,a) \;=\; \big\| \big(I - QQ^\top\big)\Phi(a) \big\|_F^2, \tag{17}$$

which is the squared distance between the feature block $\Phi(a)$ and the subspace spanned by the feature blocks of the acquired geometries in B. Note that Eq. 17 is computed from the basis Q alone, and therefore the batch acquisition never forms the kernel block $\bar{k}(B,B)$ and requires no matrix inverse.

Since $I - QQ^\top$ is an orthogonal projector, we have $\bar{k}_B(a,a) \preceq \bar{k}(a,a)$ for every input function a, and thus conditioning can only decrease the predictive variance. Furthermore, a candidate whose feature block $\Phi(a)$ lies in the range of Q is collapsed by the projector and receives a conditioned variance of zero, which is the case for a candidate that duplicates a geometry already acquired in the current batch.

B.3 Extending the Orthonormal Basis

The basis Q is empty at the start of every batch, and grows after each geometry is synthesized. After the geometry a^b is synthesized at the batch slot b, we project its feature block onto the basis and take the residual $R = \Phi(a^b) - Q_{b-1}\big(Q_{b-1}^\top \Phi(a^b)\big)$, and compute the thin QR decomposition $R = \tilde{Q}T$. Each diagonal entry $|T_{jj}|$ is the norm of the component of the j-th column of R that is not spanned by its preceding columns, i.e. how much new variance the direction $\tilde{q}_j$ carries. We keep only the directions whose magnitude is a non-negligible fraction of the feature block,

$$Q_b \;=\; \big[\, Q_{b-1},\; \tilde{Q}_{:,\mathcal{J}} \,\big], \qquad \mathcal{J} \;=\; \big\{\, j \;:\; |T_{jj}| \;\geq\; \epsilon \,\|\Phi(a^b)\|_F \,\big\}, \tag{18}$$

with $\epsilon = 10^{-3}$, as the discarded directions carry no new variance and would otherwise corrupt the orthonormality of Q_b once normalized.

B.4 Seeding of Geometries

Each batch slot starts its gradient ascent from a geometry of the current training dataset, which also serves as the anchor of its trust region. Throughout the seeding and the ascent, the non-geometric parameters are fixed to a reference value $\bar{a}_p$, so that differences in the predictive variance are only attributed to the geometry. We select the N_b seeds to be both uncertain and diverse in two stages (Algorithm 2). First, we compute the total predictive variance $v(a_g) = \|\Phi(a_g, \bar{a}_p)\|_F^2$ of every training geometry, and shortlist the $S = 4N_b$ geometries with the highest variance. Second, we select the seeds from the shortlist by farthest-point sampling in the kernel distance $d_k(a_g, a'_g) = \|\Phi(a_g, \bar{a}_p) - \Phi(a'_g, \bar{a}_p)\|_F$, starting from the geometry with the highest variance. Without the second stage, the most uncertain geometries tend to cluster in one region of the design space, and anchoring the batch slots to them would counteract the diversity enforced by the conditioning.

B.5 The Complete Algorithm

Algorithm 3 summarizes a single AL iteration of AGS, where we write $\Phi(z) = \Phi(a_g(z), \bar{a}_p)$ for the feature block of the geometry with design z at the reference parameters. We maximize the objective of each batch slot with Adam for $T = 50$ steps, with a learning rate of 0.04 decayed by a cosine schedule to 5% of its initial value. As the design variables have different ranges, the Adam update of each variable is scaled by half of its range, and after every step the design is projected back onto

Algorithm 3 A single AL iteration of AGS

Require: neural operator f_θ trained on $\mathcal{D}_{i-1}$, batch size N_b, trust-region weight μ, ascent steps T, reference parameters $\bar{a}_p$

Build the low-rank posterior Σ over $\mathcal{D}_{i-1}$ ▷ App. B.1
$\{a^1_{g,0}, \dots, a^{N_b}_{g,0}\} \leftarrow$ seeds from Algorithm 2
$Q \leftarrow [\,]$
for $b = 1, \dots, N_b$ **do**
  $z \leftarrow z^b_0$, the design of the seed $a^b_{g,0}$
  for $t = 1, \dots, T$ **do**
    $z \leftarrow \Pi_{\mathcal{Z}}\Big(\textsc{Adam}\big(\nabla_z\big[\,\|(I - QQ^\top)\Phi(z)\|_F^2 - \mu\,\|\Phi(z) - \Phi(z^b_0)\|_F^2\big]\big)\Big)$
  end for
  $a^b_g \leftarrow a_g(z)$, extend Q with $\Phi(z)$ ▷ Eq. 18
  $a^\star_{p,b} \sim \mathcal{U}(\mathcal{A}_p)$
end for
$\mathcal{D}_i \leftarrow \mathcal{D}_{i-1} \cup \big\{\big((a^b_g, a^\star_{p,b}),\, \mathcal{G}(a^b_g, a^\star_{p,b})\big)\big\}_{b=1}^{N_b}$, and retrain f_θ on $\mathcal{D}_i$

the admissible design set $\mathcal{Z}$. We use the trust-region weight $\mu = 0.3$ in both experiments. Once a geometry a^b_g is synthesized, its non-geometric parameters a^b_p are sampled uniformly from their range $\mathcal{P}$, and the sample (a^b_g, a^b_p) is labelled by the oracle. Finally, the neural operator is retrained from scratch on the extended dataset.

C Datasets

Table 2 reports the dataset sizes and the active learning budgets of both problems. The Uniform baseline is trained on the full i.i.d. training split. AGS and PaPQS start from an initial dataset $\mathcal{D}_0$ of i.i.d. uniform samples and acquire N_b geometries in each of N_{al} iterations, so that $|\mathcal{D}_0| + N_{al}N_b = N_{tr}$ and all methods are trained on the same number of labels. The validation split is used for model selection and for calibrating the prior precision of the Laplace approximation as well as early stopping, and the test split measures the in-distribution error. Note that the validation dataset is also i.i.d. uniform, and thus, AL models never see OOD datasets for early stopping or prior calibration.

Table 2: Dataset splits and active learning budgets. $|\mathcal{D}_0|$ is the initial dataset size, N_b is the acquisition batch size, and N_{al} is the number of acquisition iterations.

Dataset	N_{tr}	N_{val}	N_{test}	$\lvert\mathcal{D}_0\rvert$	N_b	N_{al}
Airfoil	2048	128	512	128	128	15
Reaction-diffusion	512	128	256	64	64	7

C.1 Airfoil

The Airfoil dataset contains steady, inviscid, compressible flows around two-dimensional airfoils, governed by the Euler equations. Following Wen et al. (2026), each simulation is parameterized by the free-stream Mach number M_∞ and the angle of attack α, which form the physical input $a_p = (M_\infty, \alpha)$, sampled uniformly from $M_\infty \in [0.5, 0.95]$ and $\alpha \in [0.5^\circ, 5^\circ]$. This is a narrower Mach range than the $[0.5, 1.4]$ of Wen et al. (2026) and excludes supersonic free streams. Over most of this range, the flow is transonic, i.e. a supersonic pocket forms over the airfoil and terminates in a shock, whose existence, position, and strength depend sensitively on the local curvature and thickness of the section, so a small deformation can move a shock or create a new one. The map from geometry to solution is therefore sharply varying, which makes the Airfoil problem a natural test for geometric active learning and OOD evaluation. The neural operator receives the geometry a_g as 200 points resampled uniformly by arc length along the surface and predicts $c = 3$ output channels, namely the pressure coefficient C_p and the velocity components u and v.

Labels are produced by SU2 (Economon et al., 2016) on an unstructured O-type mesh generated by Gmsh (Geuzaine & Remacle, 2009), with a slip wall on the airfoil and a far-field boundary at 50 chords. Each geometry is solved once on a base mesh. The mesh is then refined by a Hessian-based metric of the Mach field so that cells concentrate at the shock, and the flow is solved again; the resulting meshes have roughly $28{,}000$ nodes. Solver and meshing settings are identical for every sample and are given in our released configuration. The labels are stored at the nodes of the adapted mesh without interpolation, so the number of nodes varies between samples, and each sample is presented to the operator as a random subset of L of its nodes, redrawn every training step and held fixed per sample at evaluation. We use $L = 6000$ inside the acquisition loop, where the sample count is small and many models are trained, and retrain and evaluate all reported models at $L = 14000$. All baselines use the same L, so the comparison is unaffected. Because the adapted mesh is not a differentiable function of the FFD design z, AGS instead evaluates the predictive variance on a fixed proxy query set of 1456 points comprised of 256 points on the deformed surface and a 40×30 grid over $[-1.5, 5]\times[-3, 3]$, clustered around the airfoil by a hyperbolic sine spacing. During acquisition, the physical parameters are fixed at the midpoints of their ranges, $a_p^{\mathrm{AL}} = (0.725, 2.75^\circ)$, and each synthesized geometry is then labelled at a single new (M_∞, α) drawn uniformly from the training ranges.

The base shape is the NACA 0012 airfoil, embedded in an FFD lattice of 12×3 control points. We displace the control points only vertically, which preserves the chord, and fix the lattice columns at the leading and trailing edges, which leaves 30 free displacements and the design space $\mathcal{Z}^{(s)} = [-z_{\max}, z_{\max}]^{30}$ with $z_{\max} = 0.06$ chords. The training distribution μ_{tr} draws the 30 displacements uniformly from $\mathcal{Z}^{(s)}$. During the gradient ascent of AGS, the projection onto $\mathcal{Z}^{(s)}$ also rescales the displacement whenever the maximum thickness exceeds 22% of the chord to prevent invalid airfoils. Before labelling, both random and synthesized geometries are rejected if their thickness-to-chord ratio lies outside $[0.04, 0.22]$ or if their upper and lower surfaces nearly touch. In the reported results, none of the geometries synthesized by AGS or PaPQS were rejected. The OOD test sets apply the same FFD lattice to the E387, FX 63-137, MH 113, and RAE 2822 airfoils from the UIUC database, with the maximum displacement of each base chosen so that its median surface displacement matches that of the NACA 0012 training distribution. For each OOD family, we generate 64 samples for evaluation, separate from the N_{test} in-distribution samples.

C.2 Reaction-Diffusion

The reaction-diffusion (RD) dataset contains solutions of a bistable reaction-diffusion equation with a geometric source term on the periodic domain $\Omega = [0, 10]^2$,

$$\partial_t u = D\Delta u + r\, u(1-u)(u-a) + A\, s_g(x), \qquad u(x, 0) = 0, \tag{19}$$

with diffusivity $D = 0.5$ and reactivity $r = 4$. The geometry a_g is a filled point cloud, and $s_g(x) \in [0, 1]$ is a smooth indicator of its support, obtained by summing a Gaussian kernel over the points and clipping, so that the source is differentiable with respect to the point positions. The physical input $a_p = (A, a)$ consists of the source amplitude $A \in [0.3, 0.8]$ and the reaction threshold $a \in [0.32, 0.42]$, both sampled uniformly. Because the reaction is bistable, a sustained source either maintains a bounded hot spot on its support (quenching) or ignites a front that spreads over the whole domain (ignition). Which of the two occurs depends on the size and shape of the support as well as on (A, a), so two supports that are close in the FFD design space can produce entirely different fields. Since the final state is nearly binary, the operator predicts u at five times $t \in \{8, 16, 24, 32, 40\}$, stacked as $c = 5$ output channels.

We compute the labels with the pseudo-spectral solver of Exponax (Koehler et al., 2024) with a time step of 0.02 for 2000 steps up to $T = 40$, on a uniform 64×64 grid. No meshing is required: the neural operator receives the source on the full grid and is trained and evaluated on all $L = 4096$ grid points. For acquisition, AGS evaluates the predictive variance on every second grid point in each direction ($L = 1024$), with the physical parameters fixed at the midpoints of their ranges, $a_p^{\mathrm{AL}} = (0.55, 0.37)$, and each synthesized geometry is then labelled at a new (A, a) drawn uniformly from the training ranges.

The training bases $\mathcal{S}_{tr}$ are a disc and an ellipse of aspect ratio 3, rescaled to a common area so that they differ in shape rather than in their distance to the ignition threshold. Each base is embedded in its own 4×4 FFD lattice whose control points are displaced in both directions, giving the design

space $\mathcal{Z}^{(s)} = [-z_{\max}, z_{\max}]^{32}$ with $z_{\max} = 1.25$, and the training distribution μ_{tr} draws a base and its displacements uniformly. Since the support is a filled cloud, a folded deformation still yields a valid source, so the box is the only constraint during the ascent. Before labelling, a support is rejected if its area falls outside $[0.35, 3]$ times the reference area, if it comes too close to the domain boundary, or if it is too diffuse to reach the full amplitude over at least 10% of its area. No uniform or PaPQS sample was rejected, whereas 74 of the 448 geometries synthesized by AGS were, all for being too diffuse; we replace each with a uniform draw in the same iteration, so that all methods are trained on the same number of labels.

We evaluate on three OOD distributions. Two are unseen support families, an ellipse of aspect ratio 10 and a three-lobed star, which are area-matched to the training bases and deformed only mildly (about a quarter of the training deformation), so that the shift lies in the family rather than in the amount of deformation. The third keeps the training bases but doubles the FFD box to $z_{\max} = 2.5$. The two family sets contain 75 samples each and the deformation set 150, with (A, a) drawn from the training ranges.

D Experiment Settings

D.1 Hyperparameters

In all experiments in Section 4, we used Transolver as the neural operator backbone. We adopted hyperparameters similar to the Airfoil experiments in Wu et al. (2024) and used a smaller version for RD. Table 3 reports the Transolver configuration. For training, we used the AdamW optimizer (Loshchilov & Hutter, 2019) with a Cosine annealing learning rate schedule for up to 500 epochs, a starting learning rate of 10^{-3} and a weight decay rate of 10^{-4}. All models used early stopping on the in-distribution validation dataset.

Table 3: Transolver configuration.

	Airfoil	RD
Num. blocks	8	6
Latent dim.	128	96
Num. slices	64	48
Num. heads	8	6
Lifting width	256	192

D.2 Computational Resources

To run AGS, we used one Nvidia H100 GPU with 80 GB of VRAM. Furthermore, we used 30 CPU cores for running parallel airfoil simulations for both uniform and AL datasets. We report a breakdown of time and memory consumption of AGS in Table 4. In comparison, PaPQS with $N_{al} = 7$ on Airfoil took about 13 Hours.

Table 4: Computational costs for AGS. Note that, other than the total time, other reported times are per AGS iteration. Both Airfoil and RD are run on an H100 GPU but with different N_b. Also, the total time for Airfoil is reported from the first 7 AL iterations.

	Airfoil ($N_b = 128$)	RD ($N_b = 64$)
Peak VRAM Usage	47 GB	18 GB
Total time ($N_{al} = 7$)	22 Hrs	18 Hrs
Laplace posterior	7 Mins	5 Mins
Seed shortlist	38 Mins	30 Mins
Greedy batch acquisition	2 Hrs	1.5 Hrs
Oracle simulation	30 Mins ($\times$30 CPU Cores)	8 Mins (GPU)

E Extended Results

E.1 Errors at Every Budget

In addition to the budget curves in Section 4, we report errors for Airfoil and RD at all training budgets in Tables 5 and 6.

E.2 Model-Agnostic Capabilities and Data Transferability

Throughout the experiments in Section 4, we used Transolver as the main neural operator architecture for synthesizing the geometries using AGS. In this section, we are interested in the model-agnostic capabilities of AGS, i.e. whether AGS delivers the same OOD improvements when used with another neural operator, and whether the dataset acquired with one architecture can be transferred to train another architecture with similar improvements. We note that the Laplace approximation for AGS on GNOT was restricted to only the final projection layers, which results in roughly the same number of active parameters as the active final attention block in Transolver. We used the same $r = 512$ low-rank approximation for GGN.

Similar to the airfoil experiments in Section 4, we train Uniform, PaPQS, and AGS models using the GNOT Hao et al. (2023) architecture. Furthermore, we also train GNOT with the dataset acquired by running AGS using Transolver (the same dataset used in Section 4). Table 7 reports the in-distribution and OOD errors for GNOT. As expected, GNOT performs worse than Transolver across all methods, consistent with the results reported in Wu et al. (2024). Similar to Transolver results, AGS improves OOD generalization compared to i.i.d. geometries, while performing slightly worse on in-distribution data and comparably in RAE 2822. When trained with Transolver AGS data, the improvements are partial and not as consistent. Still, the transferred dataset improves the OOD generalization, and even surpasses the native GNOT AGS data for the E387 airfoils, which may be explained by the last-layer active parameters for posterior estimation in GNOT instead of the attention blocks.

E.3 Error vs. Variance

As discussed in Section 3.1, we use predictive variance as a surrogate for the error. We further showed that the Laplace variance is highly correlated with the errors for geometries close to the training dataset, while the correlation degrades for farther geometries (Section 4.3). Here, we examine the Laplace variance at the solution field level and examine the high-error regions. We again apply the Laplace approximation to the Uniform model (trained with i.i.d. uniform geometries) and obtain the kernel $\bar{k}$. Figure 4 shows the predictive standard deviations as well as the errors in the predicted pressure fields for in-distribution and OOD samples. We observe that in most cases, the predictive variance obtained by Laplace approximation is able to identify the high-error regions, mostly near the shocks. However, we note that as the OOD distributions move farther from the i.i.d. training dataset, the linearized posterior fails to capture some of the high-error regions and becomes overconfident. For instance, with the deformed FX 63-137 airfoil (Figure 4, bottom row), the posterior fails to capture the erroneous regions below the trailing edge and above the front camber. The trust region regularization introduced in Section 3.3 stems from the same observation.

Table 5: Airfoil results for every label budget N_{tr} and every evaluation set. Reported are the mean ± standard deviation over three training seeds.

	In-distribution		RAE 2822		E387		FX 63-137		MH 113	
	rel-L^2	rel. C_D	rel-L^2	rel. C_D	rel-L^2	rel. C_D	rel-L^2	rel. C_D	rel-L^2	rel. C_D
	$N_{tr} = 512$									
i.i.d. uniform	**0.065**±0.007	0.110±0.027	**0.106**±0.010	0.177±0.008	0.156±0.014	0.291±0.027	0.269±0.037	0.645±0.107	0.262±0.043	0.613±0.100
PaPQS	0.066±0.006	**0.109**±0.017	0.111±0.009	**0.165**±0.013	0.141±0.005	0.299±0.069	0.232±0.016	0.548±0.073	0.229±0.016	0.512±0.096
AGS (ours)	0.078±0.004	0.136±0.014	0.115±0.006	0.168±0.024	**0.130**±0.003	**0.246**±0.036	**0.195**±0.003	**0.329**±0.060	**0.204**±0.005	**0.292**±0.069
	$N_{tr} = 1024$									
i.i.d. uniform	**0.047**±0.003	**0.061**±0.005	**0.087**±0.004	0.136±0.020	0.133±0.006	0.245±0.013	0.242±0.013	0.635±0.038	0.244±0.003	0.613±0.008
PaPQS	0.051±0.003	0.069±0.002	0.092±0.005	0.136±0.007	0.127±0.010	0.232±0.041	0.227±0.007	0.526±0.068	0.222±0.003	0.447±0.108
AGS (ours)	0.057±0.002	0.077±0.006	0.095±0.002	**0.107**±0.009	**0.110**±0.017	**0.129**±0.042	**0.179**±0.037	**0.332**±0.062	**0.180**±0.035	**0.234**±0.052
	$N_{tr} = 1536$									
i.i.d. uniform	**0.041**±0.000	**0.057**±0.003	**0.081**±0.006	0.109±0.006	0.132±0.020	0.237±0.065	0.232±0.036	0.521±0.043	0.231±0.034	0.473±0.076
PaPQS	0.041±0.002	0.058±0.005	0.083±0.005	0.116±0.005	0.159±0.055	0.300±0.178	0.268±0.041	0.631±0.126	0.277±0.056	0.589±0.185
AGS (ours)	0.051±0.002	0.075±0.005	0.089±0.004	**0.103**±0.014	**0.101**±0.005	**0.141**±0.014	**0.165**±0.003	**0.298**±0.058	**0.165**±0.007	**0.221**±0.050
	$N_{tr} = 2048$									
i.i.d. uniform	0.037±0.002	0.051±0.007	**0.079**±0.009	0.125±0.027	0.130±0.026	0.297±0.066	0.244±0.032	0.665±0.052	0.246±0.027	0.648±0.082
PaPQS	**0.035**±0.001	**0.041**±0.000	0.083±0.003	**0.103**±0.011	0.127±0.020	0.260±0.103	0.231±0.025	0.559±0.069	0.243±0.019	0.513±0.088
AGS (ours)	0.047±0.001	0.066±0.005	0.088±0.005	0.106±0.015	**0.102**±0.004	**0.141**±0.027	**0.147**±0.015	**0.264**±0.040	**0.148**±0.020	**0.183**±0.054

Table 6: Reaction-diffusion root mean squared errors for every label budget N_{tr} and every evaluation set. Reported are the mean $\pm$ standard deviation over three training seeds.

	In-distribution	Ellipse 10:1	3-lobed	Deformation 2×
	$N = 192$			
i.i.d. uniform	0.053±0.017	**0.060±0.012**	0.059±0.008	0.087±0.014
PaPQS	0.038±0.005	0.064±0.010	0.047±0.016	0.071±0.011
AGS (ours)	**0.038±0.005**	0.065±0.007	**0.036±0.002**	**0.064±0.007**
	$N = 320$			
i.i.d. uniform	0.032±0.000	0.068±0.005	0.037±0.002	0.068±0.004
PaPQS	0.032±0.005	0.059±0.002	0.042±0.010	0.066±0.009
AGS (ours)	**0.029±0.000**	**0.058±0.001**	**0.030±0.003**	**0.057±0.000**
	$N = 448$			
i.i.d. uniform	0.024±0.003	0.063±0.004	0.032±0.003	0.057±0.007
PaPQS	**0.022±0.003**	0.070±0.005	**0.025±0.007**	0.052±0.005
AGS (ours)	0.026±0.005	**0.054±0.004**	0.028±0.005	**0.051±0.006**
	$N = 512$			
i.i.d. uniform	0.026±0.001	0.069±0.006	0.033±0.002	0.060±0.002
PaPQS	0.022±0.002	0.076±0.006	0.026±0.005	0.050±0.003
AGS (ours)	**0.020±0.000**	**0.049±0.002**	**0.024±0.002**	**0.047±0.001**

Table 7: Relative errors of the predicted field and of the drag coefficient integrated from its surface pressure. All rows are GNOT models. "AGS (Transolver data)" is GNOT trained on the data acquired by the Transolver AGS run. Reported are the means over three training seeds.

	In-distribution		RAE 2822		E387		FX 63-137		MH 113	
	rel-L^2	rel. C_D	rel-L^2	rel. C_D	rel-L^2	rel. C_D	rel-L^2	rel. C_D	rel-L^2	rel. C_D
	$N_{tr} = 512,\ N_{al} = 3$									
i.i.d. uniform - GNOT	0.067	0.104	0.106	0.173	0.140	0.287	0.220	0.495	0.241	0.413
PaPQS - GNOT	**0.067**	0.109	0.112	0.172	0.144	0.314	0.234	0.408	0.248	0.355
AGS (Transolver data)	0.073	0.113	**0.104**	0.138	**0.127**	**0.182**	0.217	0.465	**0.229**	0.374
AGS (native GNOT data)	0.073	**0.103**	0.109	**0.133**	0.142	0.216	**0.209**	**0.263**	0.234	**0.253**
	$N_{tr} = 1024,\ N_{al} = 7$									
i.i.d. uniform - GNOT	0.055	0.081	0.094	0.163	0.130	0.257	0.235	0.542	0.259	0.483
PaPQS - GNOT	**0.052**	**0.075**	**0.091**	0.118	0.123	**0.180**	0.247	0.443	0.269	0.401
AGS (Transolver data)	0.059	0.082	0.098	0.138	**0.111**	0.187	0.233	0.390	0.261	0.369
AGS (native GNOT data)	0.062	0.092	0.103	**0.115**	0.129	0.191	**0.181**	**0.262**	**0.211**	**0.243**

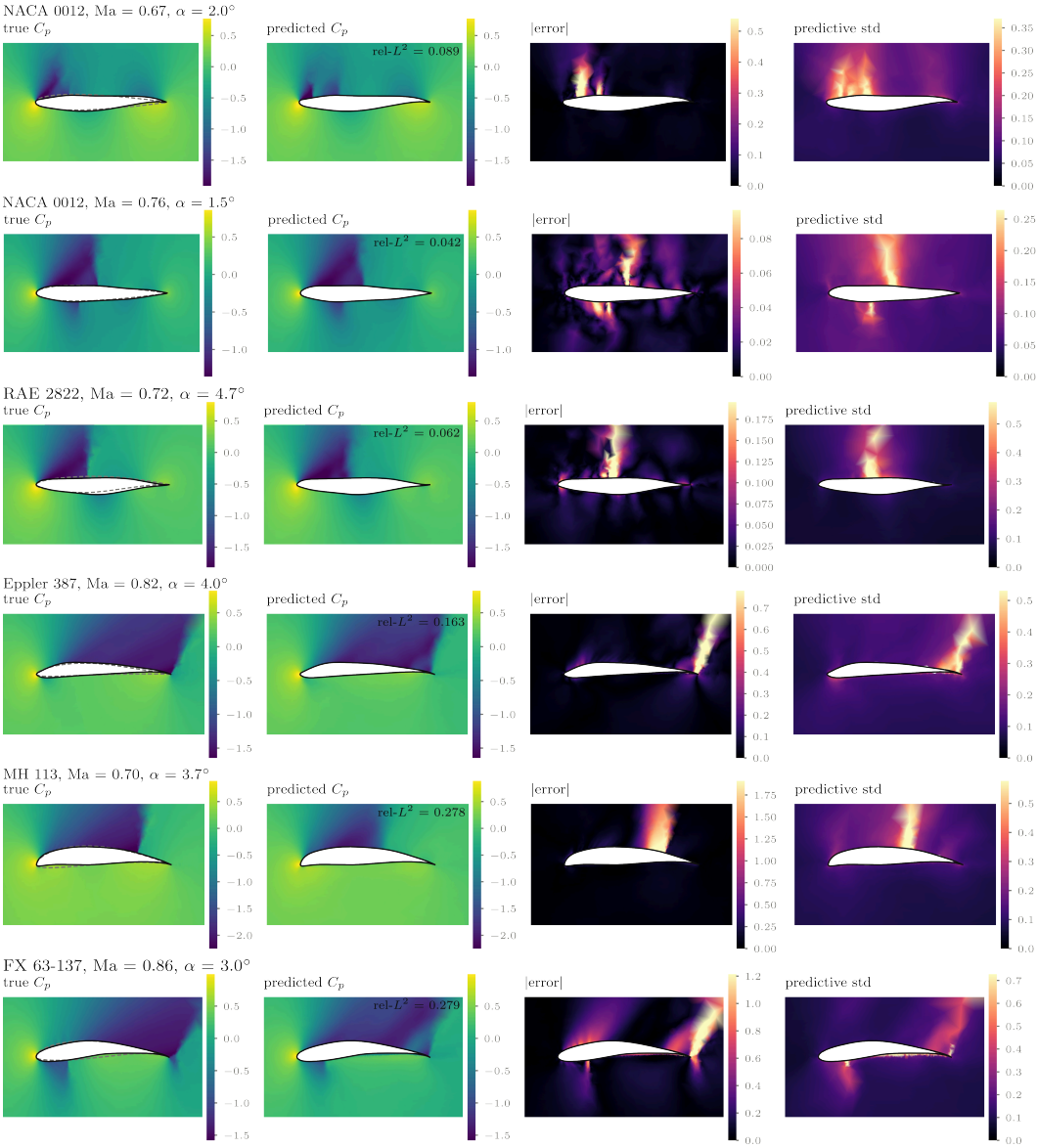


Figure 4: Predictions, errors, and predictive standard deviations of the pressure field for in-distribution (top two rows) and OOD airfoils.