Compute-optimal data scaling for neural surrogates via multi-fidelity training

Paul Setinek[*1] **Fabian Paischer**[1,2] **Nils Thuerey**[3,4]

Johannes Brandstetter[1,2] **Felix Koehler**[*3,4]

[1] Institute for Machine Learning, JKU Linz [2] Mistral AI

[3] Technical University of Munich [4] Munich Center for Machine Learning (MCML)

Code Data

Abstract

Neural surrogates for Partial Differential Equations (PDEs) are usually trained on synthetic data from numerical simulators. As models grow and scaling laws demand larger datasets, the cost of data generation often outweighs the training cost and becomes the bottleneck in the scaling quest. Unlike data in other domains, however, each PDE sample has a tunable cost that depends on its numerical fidelity, with the resulting cost-accuracy trade-off governed by known convergence rates. We exploit this property by formalizing neural surrogate training as a function of both data quantity and fidelity under a fixed data generation compute budget, treating fidelity as a fine-grained discrete spectrum rather than a binary distinction. Building on this, we show that multi-fidelity training enables Pareto-optimal error scaling against data generation compute. By using simulations at lower fidelities as an additional training signal, training on the optimal fidelity mixture matches the accuracy of high-fidelity-only training, with up to $6.3\times$ less data generation compute across our settings. We validate these findings across six PDE problems spanning structured grids and unstructured meshes, two distinct error sources (discretization and iterative-solver error), and four surrogate architectures, totaling over 7,000 training runs.

1 Introduction

Modern machine learning owes a large portion of its success to scale. This holds across various domains, like computer vision, time series, and especially natural language processing, where models reach hundreds of billions of parameters (LLaMA-3 Team, 2024; Mistral AI, 2024). Following the seminal works on neural scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022), this growth has been enabled by jointly scaling up model size, training compute and datasets.

The Scientific Machine Learning (SciML) community has started to adopt this paradigm, leading to notable breakthroughs in weather and climate forecasting (Keisler, 2022; Pathak et al., 2022; Nguyen et al., 2023; Price et al., 2025; Bodnar et al., 2025), material design (Merchant et al., 2023; Zeni et al., 2025; Yang et al., 2024) or protein folding (Jumper et al., 2021; Abramson et al., 2024). These advances have partly been enabled by large curated public datasets, such as ERA5 in weather modeling (Hersbach et al., 2020) or the Protein Data Bank (PDB) (Berman et al., 2000). However, only few scientific domains benefit from such large readily available datasets.

Neural surrogates for PDEs, which have also been scaled to billions of parameters (Paischer et al., 2025; Wang et al., 2024; Bodnar et al., 2025), are typically trained on synthetic data generated by

*Equal contribution. Correspondence to setinek@ml.jku.at

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

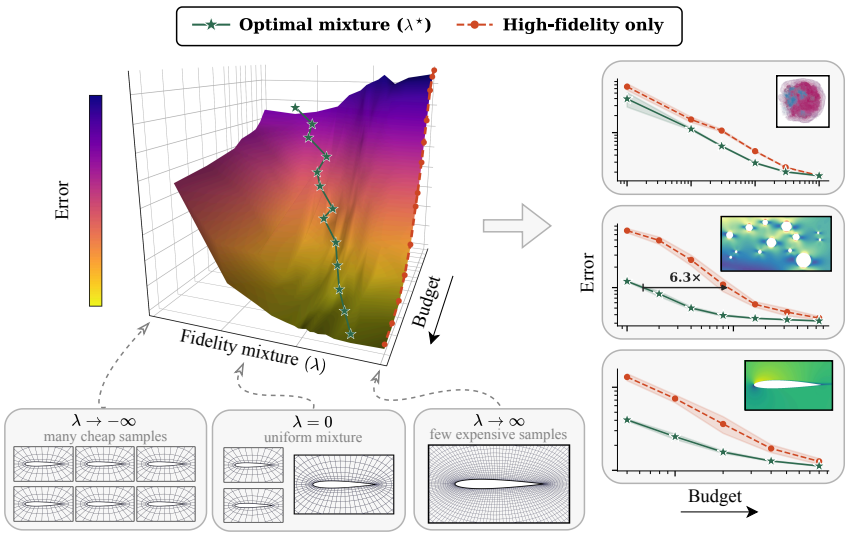


Figure 1: Overview of our multi-fidelity data scaling approach. We split the data axis into generation budget and fidelity mixture, parametrized by λ. **Bottom:** example discretization fidelity mixtures. **Right:** Across diverse settings, multi-fidelity training at optimal mixtures (green) reaches the error of high-fidelity-only training (orange, dashed) with up to $6.3\times$ less data generation compute.

numerical solvers (Holzschuh et al., 2026). Therefore, every data sample incurs a non-negligible computational cost, and scaling along the data axis cannot be done by further scraping of existing sources. Ashton et al. (2025) give a prominent example of this, showing that the dominating cost for scaling surrogate models in Computational Fluid Dynamics (CFD) is not the network training, but the data generation. Furthermore, this cost directly affects the practical viability of such models: the more expensive the dataset, the further the amortization point is pushed back, delaying the stage at which the surrogate becomes cost-effective for classical downstream tasks such as design optimization.

With this in mind, current datasets, regardless of their scale, are typically generated entirely at a single, fixed fidelity, assumed to be high enough for the task at hand. However, this paradigm neglects a fundamental property of numerical solvers: the computational cost of a sample is not fixed but depends on its fidelity, and the resulting cost-accuracy trade-off follows known convergence rates. Data quality is thus a controllable, predictable function of compute, unlike in many other domains, where it emerges from curation and filtering (Sorscher et al., 2022). This raises a natural question:

> *At the same data generation cost, can using a mixture of samples at different fidelities yield better surrogate accuracy than using high-fidelity data alone?*

We provide empirical evidence that the answer is yes (Fig. 1). Our main contributions are:

- We formalize the neural surrogate training pipeline as a function of both data quantity and fidelity under a fixed data generation compute budget, revealing the inherent trade-off between sample quality and dataset size. We are the first to treat fidelity as a fine-grained discrete spectrum throughout the pipeline, in contrast to the binary formulations prevalent in existing literature.
- We show empirically that neural surrogates learn the fidelity of their training data, and that fidelity conditioning enables steering across the fidelity spectrum at inference.
- We validate that multi-fidelity training at optimal mixtures enables Pareto-optimal data scaling: across multiple PDEs, discretization types and neural surrogate architectures, it consistently matches or outperforms single-fidelity training at matched data generation compute. To the best of our knowledge, this is the broadest empirical study of multi-fidelity neural surrogate data scaling, comprising over 7,000 training runs and more than 12,000 GPU-hours.

2 Related work

Data scaling and data quality. Neural scaling laws for language models established that data must be scaled jointly with model size and training compute to achieve optimal performance (Kaplan

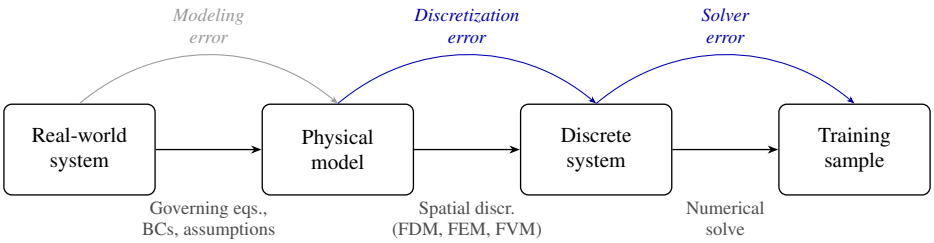


Figure 2: Data generation pipeline for neural surrogates, highlighting the primary sources of numerical error. *Discretization error* and *solver error* (blue) exhibit known convergence behaviors and can be reduced at the expense of additional compute.

et al., 2020; Hoffmann et al., 2022). Subsequent work has increasingly highlighted the importance of data quality rather than sheer quantity (Gunasekar et al., 2023; Li et al., 2023; Sachdeva et al., 2024; DeepSeek-AI, 2025), a finding that has also been acknowledged in time series (Boileau et al., 2025; Ansari et al., 2024; Auer et al., 2025) and vision-language modeling (Sorscher et al., 2022; Nguyen et al., 2022; Gadre et al., 2023). Closest to our work, Goyal et al. (2024) show that for vision-language models, the optimal data quality–quantity balance depends on the available compute budget. However, their analysis targets the budget spent on model training, whereas we focus on the budget for dataset generation, being the primary bottleneck for many AI4Science applications. In the context of neural surrogates for PDEs, recent work studies optimal model scaling in the weather domain (Subramanian et al., 2026; Yu et al., 2026). However, these are based on the openly available ERA5 dataset (Hersbach et al., 2020) and thus do not account for simulation cost or quality.

Multi-fidelity neural surrogates. Recognizing the scarcity of high-fidelity data, prior work has combined inexpensive low-fidelity and scarce high-fidelity simulations to train neural surrogates for field predictions, e.g. via composite DeepONets that learn explicit corrections on top of a low-fidelity network (Lu et al., 2022; Howard et al., 2023). Others employ transfer learning from low- to high-fidelity data with FNOs (Tang et al., 2024; Carey et al., 2025) or U-Nets (Zhang et al., 2024; Jiang and Durlofsky, 2023). While motivated by data scarcity, these approaches neither analyze how surrogate error scales with data generation cost, nor the trade-off between sample fidelity and dataset size at fixed compute. Closer to our setting, Song and Tartakovsky (2021) and Lyu et al. (2023) vary the mixture of training samples at two discretization fidelities under a fixed data generation budget for two-phase porous media flow and heat conduction, respectively. Setinek et al. (2025) explore this for a transformer trained on Reynolds-Averaged Navier-Stokes (RANS) simulations using two fidelity levels obtained from different boundary layer assumptions. Moving beyond binary fidelity levels, Sakarvadia et al. (2026) introduce multi-resolution training to remedy the failure of neural operators to generalize zero-shot across resolutions. Their low-fidelity samples, however, are created by downsampling high-fidelity simulations, therefore neither mimicking real discretization errors nor accounting for real data cost. Choudhary et al. (2025) present a different approach for data-cost-aware training of neural surrogates by mixing simulations of varying problem difficulties (Reynolds number and obstacle count in the domain) that require different compute to generate.

With our work, we unify and extend this fractured landscape: we present an extensive study of error scaling against data generation compute and fidelity mixture, treating fidelity as a fine-grained discrete spectrum spanning multiple error sources (discretization and solver), which naturally mimics real-world pipelines. We report results across various PDEs, discretization types (regular grids and unstructured meshes), and surrogate architectures including scalable transformers.

3 Methodology

3.1 Errors in numerical simulators

For most differential equations governing physical systems of interest, there exists no analytical solution, necessitating the use of numerical methods. We therefore assume access to a *numerical simulator* $\mathcal{P}_h$ to generate training data for neural surrogates. Given a (potentially vector-valued) discrete input field $\mathbf{f}_h$ at discretization h sampled from distribution $\mathcal{F}_h$, $\mathcal{P}_h$ produces a target solution

$$\mathbf{u}_h = \mathcal{P}_h(\mathbf{f}_h; \xi). \tag{1}$$

Unavoidably, the output $\mathbf{u}_h$ is an imperfect approximation of the true solution. We therefore introduce ξ as a for now abstract scalar *fidelity parameter* controlling the accuracy of the simulator. By our convention, ξ is ordered such that larger values correspond to higher accuracy. We can then define the numerical error of the simulation as:

$$e(\xi) = \mathbb{E}_{\mathbf{f}_h \sim \mathcal{F}_h} \left[\frac{\|\mathcal{P}_h(\mathbf{f}_h; \xi) - \lim_{\xi' \to \infty} \mathcal{P}_h(\mathbf{f}_h; \xi')\|_2}{\|\lim_{\xi' \to \infty} \mathcal{P}_h(\mathbf{f}_h; \xi')\|_2} \right]. \tag{2}$$

Fig. 2 illustrates the simulation pipeline, highlighting the three primary sources of numerical error.

Modeling error. The first step is to formulate a *physical model* of the underlying real-world system. This involves selecting governing equations and boundary conditions, as well as simplifying assumptions (e.g. turbulence closure models). This introduces *modeling error*, which we do not address in this work, as to date there is limited evidence of meaningful transfer across modeling fidelities (Zhang et al., 2024; Setinek et al., 2025).

Discretization error. The physical model must then be discretized in space to obtain a finite-dimensional system of equations. Different methods exist for this purpose, such as Finite Difference Method (FDM), Finite Element Method (FEM), or Finite Volume Method (FVM), each with specific requirements on the spatial grid or mesh. This step introduces *discretization error*. Here, the fidelity parameter ξ corresponds to a measure of resolution (e.g., grid points or total mesh nodes), and the error decays *algebraically*, i.e., $e(\xi) = \mathcal{O}(\xi^{-\alpha})$, where $\alpha > 0$.

Solver error. The discretized system must then be solved numerically. For linear problems, this yields a system of linear equations that can be tackled by direct methods (e.g., LU factorization) or iterative solvers (e.g., Conjugate Gradient, Gauss–Seidel, Multigrid). For nonlinear problems, an outer nonlinear iteration (e.g., Newton–Raphson) is required, each step of which involves solving a linearized sub-problem. When iterative methods are used, early termination introduces *solver error*. Here, ξ corresponds to the number of iterations, and the error decays *exponentially*, i.e. $e(\xi) = \mathcal{O}(\rho^{\xi})$, with $\rho \in (0, 1)$. The same axis has been exploited at training time by Bhatia et al. (2025) in the context of differentiable physics.

Cost–accuracy trade-off. In both regimes, increasing fidelity ξ reduces the error but incurs additional computational cost $c(\xi)$. Under a fixed compute budget B to generate a dataset at a single fidelity ξ_{fix}, one can afford at most $M = \lfloor B/c(\xi_{\text{fix}}) \rfloor$ training samples.

3.2 Multi-fidelity formalism

We now relax this requirement of a fixed fidelity and instead introduce a support of admissible fidelities $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{|\Xi|}\}$, ordered by accuracy, and with endpoints $\xi_{\min}$ and $\xi_{\max}$ (see Section C.1 for desiderata of how to select Ξ). These define a *discrete spectrum* over which we will form fidelity mixtures parametrized by $\lambda \in \mathbb{R}$. Because raw fidelity values have inconsistent units across error sources (e.g. N grid points vs K solver iterations) and span different orders of magnitude, we apply a normalization and clamping $\hat{\xi} \leftarrow \xi$ involving ξ_{min} and ξ_{max} (Section C.2). This ensures comparability of the mixture parameter λ across experiments.

Now, let $p(\hat{\xi}; \lambda, \Xi)$ be a probability mass function for admissible fidelities parametrized by λ and Ξ. For this work, we employ a softmax distribution over the discrete spectrum of fidelities:

$$p(\hat{\xi}_i; \lambda, \Xi) = \frac{\exp(\lambda\, \hat{\xi}_i)}{\sum_{j=1}^{|\Xi|} \exp(\lambda\, \hat{\xi}_j)}. \tag{3}$$

Then, $\lambda > 0$ biases towards higher fidelities, $\lambda < 0$ towards lower fidelities, and $\lambda = 0$ equals a uniform distribution. In the extreme cases, $\lambda \to \infty$ yields exclusively $\xi_{\max}$, while $\lambda \to -\infty$ results in only $\xi_{\min}$. Ultimately, λ is a hyperparameter controlling the trade-off between sample accuracy and dataset size. We can now draw a dataset $D \sim \mathcal{D}(\lambda, B)$ of fidelity-tagged input-output pairs with $\mathbb{E}[M] = B/\mathbb{E}_{\hat{\xi} \sim p}[c(\hat{\xi})]$ expected samples. Additionally, we define the notion of a "normalized budget" as $\tilde{B} = B/c(\hat{\xi}_{\max})$. We discuss the implementation of the sampling procedure in Section C.3. For illustration, Fig. 3 shows the probability mass function and example dataset compositions.

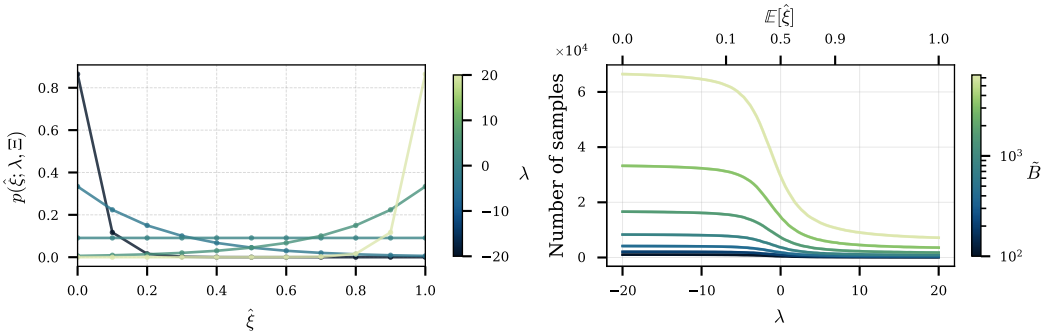


Figure 3: Overview of the multi-fidelity sampling approach. **Left:** Probability mass function with higher λ biasing towards higher fidelities $\hat{\xi}$. **Right:** Example dataset compositions for the linear elasticity solver error experiment.

To fully leverage multi-fidelity data, our surrogate model $\mathcal{S}_\theta$ accepts $\hat{\xi}$ as an additional conditioning signal besides the input field $\mathbf{f}_h$. The normalization in $\hat{\xi}$ gives the surrogate a conditioning input that is linear in the (theoretical) error, which we will see (Section 4.2) enables steering and interpolation across fidelities at inference. We train it via empirical risk minimization

$$\hat{\theta}(\lambda, B) = \arg\min_{\theta} \; \mathbb{E}_{(\mathbf{f}_h, \xi) \sim D(\lambda, B)} \left[\zeta \left(\mathcal{S}_\theta \left(\mathbf{f}_h; \hat{\xi}(\xi) \right), \; \mathcal{P}_h(\mathbf{f}_h; \xi) \right) \right] \tag{4}$$

over the available dataset (for a definition of the loss function ζ, see Section C.4). The learned parameters $\hat{\theta}(\lambda, B)$ depend on the dataset composition defined by the mixture hyperparameter λ and the available budget B. We evaluate the surrogate on the highest in-support fidelity $\xi_{\max}$[2], via

$$\mathcal{E}(\lambda, B) = \mathbb{E}_{\mathbf{f}_h \sim D_{\text{test}}} \left[\tilde{\zeta} \left(\mathcal{S}_{\hat{\theta}(\lambda, B)}(\mathbf{f}_h; \hat{\xi}(\xi_{\max})), \; \mathcal{P}_h(\mathbf{f}_h; \xi_{\max}) \right) \right], \tag{5}$$

over a hold-out dataset $D_{\text{test}} \sim \lim_{\lambda \to \infty} \mathcal{D}(\lambda, B_{\text{test}})$ (for a definition of the test metric $\tilde{\zeta}$, see Section C.5). As such, the test error is a function of the budget B and the mixture hyperparameter λ besides the model architecture, training recipe and hyperparameters. For each budget B, define the *compute-optimal mixture*

$$\lambda^\star(B) = \arg\min_{\lambda \in \mathbb{R}} \mathcal{E}(\lambda, B). \tag{6}$$

The associated *compute-optimal test-error Pareto frontier* is $B \mapsto \mathcal{E}\big(\lambda^\star(B), B\big)$.

4 Validating the multi-fidelity framework

4.1 Formulating fidelity as a fine-grained spectrum

Treating fidelity as a fine-grained spectrum is natural given the underlying numerical simulation pipeline, where discretization and solver settings can be varied in small increments. Nevertheless, most prior multi-fidelity work uses a binary formulation with two distinct fidelity levels (Section 2). We compare the two approaches at the same data generation budget. The binary approach uses only the coarsest and finest resolution, whereas our spectrum approach samples 19 additional intermediate resolutions. Fig. 4 shows that the spectrum formulation achieves a 33% lower error than the binary baseline, with a noticeably wider minimum in λ. Both mixtures outperform a surrogate trained on high-fidelity data alone. The spectrum formulation samples densely along the fidelity axis, forcing the surrogate to learn a coherent mapping that varies smoothly with ξ. This effect grows for wider endpoints and, consistent with this, the gap narrows when the fidelity range is smaller, as for solver error on a fixed mesh (Section L.1).

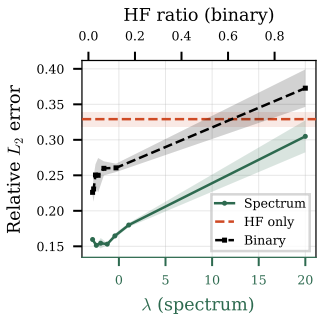


Figure 4: Spectrum vs. binary approach for a Transolver model on the hyperelasticity dataset (discretization error) at $\tilde{B} = 200$.

[2]Conditioning does not affect surrogate inference speed since the constant conditioning $\hat{\xi}(\xi_{\max})$ can be baked into the weights of the network.

4.2 Inference-time fidelity steering

Conditioning the surrogate on sample fidelity during training enables steering at inference. Feeding a chosen $\hat{\xi}$ produces an error matching the corresponding fidelity level. Fig. 5 compares a surrogate trained with our multi-fidelity approach to one trained on the same training data composition but without the conditioning mechanism. The unconditioned baseline exhibits a single error level regardless of $\hat{\xi}$, having no fidelity input. The conditioned surrogate, by contrast, traces the simulator's error curve closely across the entire training support. At interior $\hat{\xi}$ values unseen during training, the surrogate interpolates between adjacent training levels. We found the smoothness level of the interpolation to be governed by the sample count of the specific fidelity region as well as the resolution of the conditioning encoding (Section L.2). Towards $\hat{\xi} \to 1$, the multi-fidelity surrogate plateaus above the simulator, reflecting its capacity floor relative to the chosen ξ_{max} reference (Koehler and Thuerey, 2025), but reaches a lower error than the surrogate trained on high-fidelity samples only. Beyond the training support ($\hat{\xi} > 1$), the error rises again.

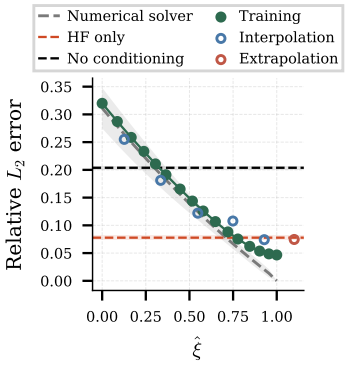


Figure 5: The multi-fidelity surrogate respects the conditioning signal. We show a Transolver model on the hyperelasticity dataset (solver error) trained at $\tilde{B} = 1{,}600$ and $\lambda^\star = -3.59$.

5 Pareto-optimal data scaling

Having validated our framing of multi-fidelity training in Section 4, we now present empirical evidence that multi-fidelity training enables Pareto-optimal data generation cost scaling across PDEs, discretization types, error sources, and surrogate architectures. We evaluate the following claims:

- **C1. Nontrivial optimum under finite budget.** For $B < B_{\text{max}}$ (a problem dependent crossover), the compute-optimal mixture is in the interior, i.e., $\mathcal{E}\big(\lambda^\star(B), B\big) < \mathcal{E}(+\infty, B)$.
- **C2. Asymptotic agreement with single fidelity.** As $B \to \infty$, $\lambda^\star(B) \to +\infty$ and $\mathcal{E}(\lambda^\star(B), B) \to \mathcal{E}(+\infty, B)$. The scaling curves converge at large budgets.

Table 1: Overview of PDEs, data representations, sources of error and architectures.

Application domain	Physics / Equation	Data format	Error source		Architectures
			Discretization	Solver	
Mathematical / PDEs	Poisson equation	1D, 2D, 3D grids	✓	✓	FNO, U-Net
Structural Mechanics	Linear elasticity	2D meshes	✓	✓	AB-UPT, Transolver
	Hyperelasticity (nonlinear)	2D meshes	✓	✓	AB-UPT, Transolver
Fluid Dynamics	RANS (airfoils)	2D meshes	✓	–*	AB-UPT, Transolver

* Solver error not studied for airfoils. For a discussion, see Section G.

We verify these claims across the experiments summarized in Table 1. For this, we test across two data representations, regular grids and irregular meshes, with architectures appropriate to each: Fourier Neural Operator (FNO) (Li et al., 2021) and U-Net (Ronneberger et al., 2015) for grids, and Transolver (Wu et al., 2024) and AB-UPT (Alkin et al., 2025) for meshes. Section 5.1 starts with the Poisson equation on regular grids, where moderate experimental cost allows us to span 1D to 3D cases. Section 5.2 then moves to unstructured meshes for structural mechanics, covering both a linear and nonlinear case. Finally, Section 5.3 demonstrates a real-world example of data generation cost savings for external aerodynamics around airfoils. All experiments are repeated across multiple random seeds for training (and dataset generation, if feasible). We report seed means ± 1 std bands and bootstrap confidence intervals for the savings estimates. Throughout Sections 5.1 and 5.2 we use budget in algorithmic units (nodes or iterations) rather than wall-clock time. Under our otherwise fixed solver configurations, these give hardware-independent, reproducible budgets. Section 5.3 instead uses CPU core-hours as a more interpretable unit. In Section 5.4, we then test an approach for determining the optimal fidelity mixture at larger budgets from a few calibration runs.

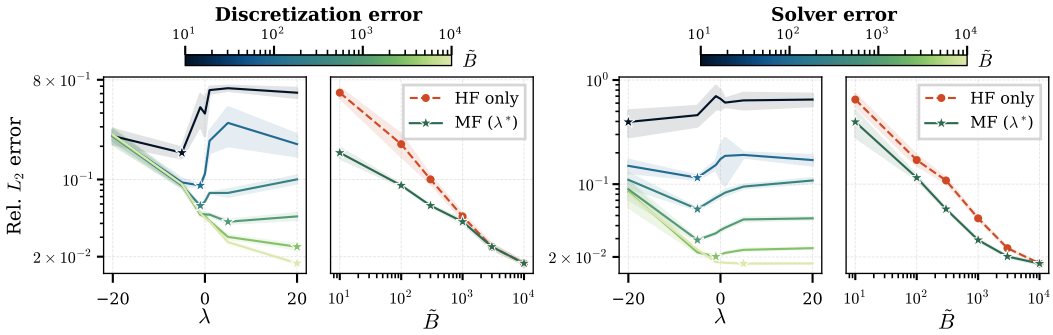


Figure 6: **Poisson 3D experiment** results for the **U-Net model**. Each group shows iso-budget curves over the mixture hyperparameter λ (left) and the compute-optimal Pareto front using $\lambda^\star(B)$ at each budget (right). Multi-fidelity training enables Pareto-optimal scaling for both sources of error. Results are averaged across seeds with shaded bands indicating ± 1 standard deviation.

5.1 Poisson equation as a testbed for large parametric studies

The Poisson equation is fundamental to many scientific simulations, appearing both as a standalone problem and as a substep in larger pipelines (e.g., incompressibility projection (Chorin, 1968)). Moreover, we see it as a suitable testbed for surrogates trying to approximate stationary problems since it ensures global spatial dependencies. We study the discretization error, controlled by the total number of grid points N, and the solver error, controlled by the number of multigrid V-cycles K. For a more detailed description see Section D.

Fig. 6 reveals that the minima in iso-budget curves lie in the interior until B_{max} and shift towards higher λ^* with growing normalized budgets $\tilde{B}$, confirming claims **C1** and **C2**. In 3D, the discretization menu spans a $216\times$ cost ratio $c(\xi_{\text{max}})/c(\xi_{\text{min}})$, against only $4\times$ on the solver axis. This drives a wider gap between High Fidelity (HF) and Multi Fidelity (MF) at any fixed error threshold on the discretization axis with $4.6\times$ over only $2.7\times$ in the solver error axis (see Fig. 9). At a moderate budget of $\tilde{B} = 300$, the 3D U-Net achieves 46.9% / 42.2% error reduction on the solver / discretization axes via MF training by drawing roughly $3\times$ / $6\times$ training samples more than the high-fidelity baseline (Tables 14 and 15).

Across all 1D and 2D experiments, the U-Net reaches a 2.0–$3.1\times$ lower terminal error (at the highest $\tilde{B}$ tested) than the FNO and the FNO recovers only 57–90% of the U-Net's relative MF gain. This suggests that the magnitude of the MF benefit depends on how far the surrogate sits from its capacity floor (full plots and numerical tables are in Section J.1).

5.2 Structural mechanics on unstructured meshes

As our next experiment, we consider the equations of elasticity, central to many structural mechanics applications. Compared to the Poisson setup, we introduce three sources of additional complexity to resemble real-world scenarios: complex geometries discretized with unstructured meshes rather than regular grids, the von Mises stress as the prediction target (a derived quantity with lower convergence rate than the displacement field it is computed from), and material nonlinearity via a Neo-Hookean constitutive model for the hyperelastic case. As before, we vary the number of mesh nodes N for the discretization axis and the number of solver iterations K for the solver axis (see Sections E and F for governing equations, geometries, and error analyses).

Fig. 7 shows the multi-fidelity data scaling results for the hyperelasticity experiment with the Transolver model. In general, we observe the same trends as presented above. For both error sources, the optimal λ^* lies in the interior at moderate budgets and shifts towards higher fidelities as $\tilde{B}$ increases, consistent with **C1** and **C2**. At a moderate budget of $\tilde{B} \approx 400$, MF training reduces the test error by 59% / 49.4% on the solver / discretization axes by drawing $8\times$ / $10.4\times$ more training samples than the HF baseline (see Tables 17 and 18).

The cross-axis comparison, however, differs from the Poisson case. Although the cost ratio is much larger for the discretization error than for the solver error (138.7 vs. 9.5 due to polynomial versus exponential error scaling and linear cost scaling for both, see Section F and Table 9), the surrogate

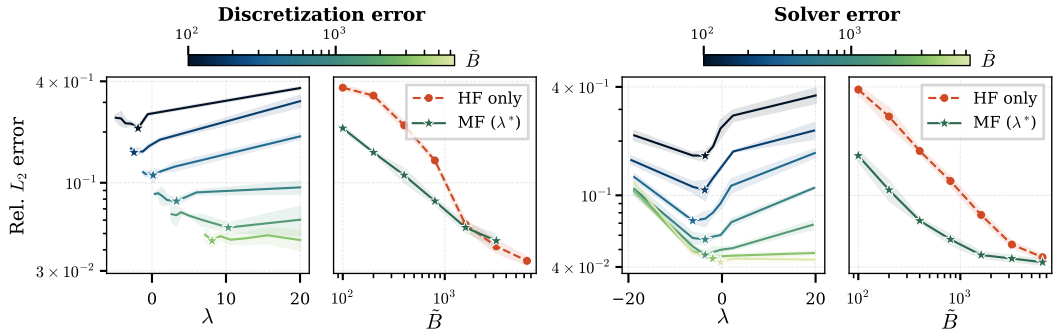


Figure 7: **Hyperelasticity experiment** results for the **Transolver model**. Multi-fidelity training also enables data savings in these cases on unstructured meshes with nonlinear PDEs. Solver error savings are higher (larger distance between the two scaling curves) due to operating at a consistent resolution.

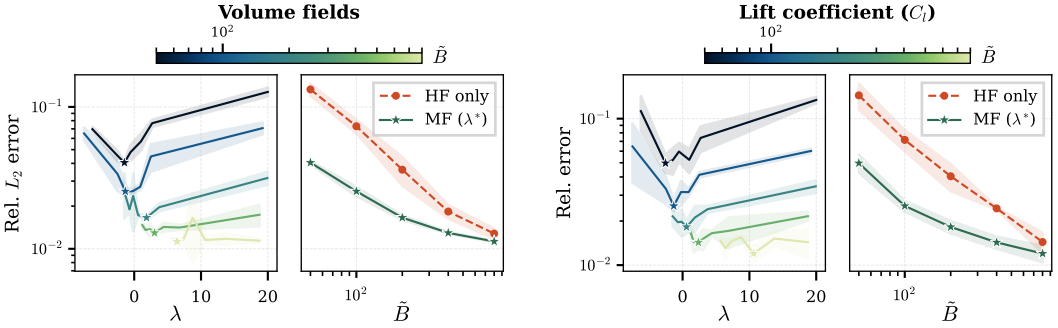


Figure 8: **Airfoil experiment** results on the **discretization error** for the **AB-UPT model**. Multi-fidelity training cuts simulation time by $1,820$ core-hours at a matched volumetric error of 2%. The savings also extend to derived engineering quantities like the lift coefficient (right plot).

cannot exploit this asymmetry. The discretization axis yields only a $2.1\times$ data generation cost reduction at 1×10^{-1} test error, while the solver axis yields a $4.8\times$ reduction at the same error level (see Fig. 9). We attribute this to the general difficulties of having neural surrogates generalize across a large range (here two orders of magnitude) of resolution, as reported in prior literature (Sakarvadia et al., 2026; Gao et al., 2025; Hu et al., 2026). This suggests that multi-fidelity over solver error, which we are the first to investigate, is the practically more suitable axis.

The same trends as presented above also hold for the AB-UPT model on this setup and both models on the easier setup of linear elasticity, where MF training delivers up to $6.3\times$ cost savings on the solver axis at the same 10% threshold (see Fig. 9). The full plots and numerical results can be found in Sections J.2 and J.3.

5.3 Towards a real use-case in external aerodynamics

We now turn to an external aerodynamics setting, where neural surrogates have direct application in shape optimization for drag and lift behavior. Based on the work of Bonnet et al. (2022), we simulate 2D flows past airfoils using the RANS equations, varying the number of mesh nodes N to construct a multi-fidelity dataset. The surrogate predicts the pressure p and velocity field $\mathbf{u} = [u_x, u_y]$ in the volume, as well as pressure p and wall shear-stress $\boldsymbol{\tau} = [\tau_x, \tau_y]$ on the airfoil surface. We measure dataset generation compute in wall-clock simulation time. Generating all samples at the highest fidelity consumed approximately 13,580 CPU core-hours.

Fig. 8 confirms **C1** and **C2** in this setting. At an average 2% relative L_2 error threshold (in the volumetric fields), multi-fidelity training reduces data generation cost by $2.4\times$ / $2.5\times$ for Transolver / AB-UPT (Fig. 9). This shrinks data generation time from the equivalent of two days (on our 64-core CPU) to under a day. At a moderate budget of $\tilde{B} \approx 200$, AB-UPT cuts the relative L_2 error by 54.1% on the volume fields (drawing $2.9\times$ more samples than the HF baseline) and by 60.3% on the surface fields (Tables 23 and 24). The lift and drag coefficients, which are of direct engineering interest being the typical objectives in shape-optimization pipelines, also follow the multi-fidelity Pareto front

(Fig. 26). This indicates that the data generation savings transfer from the predicted fields to the downstream design metrics used in practice.

5.4 Extrapolating optimal fidelity mixtures across budgets: a post-hoc analysis

The previous sections clearly demonstrate the benefit of multi-fidelity training. A remaining question is how one can, *a priori*, determine the optimal fidelity mixture $\lambda^\star$ for a given training data generation compute budget B. Previous work in neural scaling laws for language modeling (Kaplan et al., 2020; Hoffmann et al., 2022) was able to give a functional form with universal parameters that extrapolated and transferred well due to strongly canonical settings (similar data, architectures and single-epoch training). In scientific ML, however, every PDE, geometry, approximation, discretization, solver, and surrogate architecture reshapes the optimal fidelity mixture landscape making them distinct for each specific case. We therefore propose and test a calibration recipe rather than a universal law, using the simple ansatz

$$\lambda^\star(\tilde{B}) = a + b \log(\tilde{B}) \,. \tag{7}$$

We outline the specifics of the fitting process and details on how to make it robust in Section K.1 as well as the overhead cost of the approach in Section K.2.

To test whether the coefficients calibrated at small budgets extrapolate to substantially larger ones, Table 2 gives a summary of the extrapolation capabilities for our mesh-based experiments (detailed results across all settings can be found in Section K.3). We measure predictive power using regret $r(\tilde{B}) = \mathcal{E}\big(\lambda_{\text{pred}}(\tilde{B}), \tilde{B}\big)/\mathcal{E}\big(\lambda^\star(\tilde{B}), \tilde{B}\big) - 1$, with λ_{pred} being the predicted optimal mixing parameter.

Table 2: Mean held-out regret (%) of the extrapolation recipe per dataset and fidelity axis (mesh experiments) across the first three held-out budgets (ascending, means taken over architectures).

Dataset	**Solver error**			**Discretization error**		
	1st	**2nd**	**3rd**	**1st**	**2nd**	**3rd**
Linear elasticity	4.0	5.2	4.4	6.3	4.5	4.0
Hyperelasticity	2.3	3.4	3.9	9.0	10.0	4.3
Airfoil	–	–	–	0.0	1.7	–

The average regret across all extrapolation budgets is $< 5\%$ on the solver error setups and $\leq 10\%$ on the mesh discretization settings, showing that the recipe yields good extrapolation at small calibration costs. For almost all extrapolated mixtures λ, the error, albeit showing regret, is lower than the corresponding HF-only surrogate (except for some largest budget cases where the two scaling curves already converged (as per claim **C2**)). In the regular grid Poisson discretization setups, where the optimal mixture transitions abruptly to the high-fidelity regime, our ansatz does not capture the trend. We discuss this limitation and report the full results in Section K.3.

6 Conclusion, limitations and future work

Limitations and outlook. Our experiments assume (i) that data generation cost dominates training cost, and (ii) that the models are large enough and trained long enough to absorb the additional data. To probe this, we provide ablations of increasing model size and longer training durations in Sections L.4 and L.5 showing that our findings are robust across setups. Our extrapolative recipe works well in most settings but degrades on some discretization error settings. A complete scaling law analysis would need to jointly vary model size, training compute, dataset generation compute and dataset fidelity mixture across all problems and sources of error, a combinatorial search space that is currently intractable. This opens up two research directions that could improve mixture predictions avoiding this expensive scaling study: (i) a theoretical framework integrating both the approximation behavior of neural surrogates, so far studied in restricted settings and single fidelity data (Kovachki et al., 2021), with the numerical convergence behavior of the underlying simulator, and (ii) active learning (Musekamp et al., 2025) to discover optimal mixtures directly rather than extrapolating them. Moreover, our study covers (statistical-)steady-state problems at moderate scale. Extending the framework to complex 3D geometries and industrial-scale meshes remains open. Considering transient dynamics we see two natural extensions: autoregressive surrogates at a fixed time step can

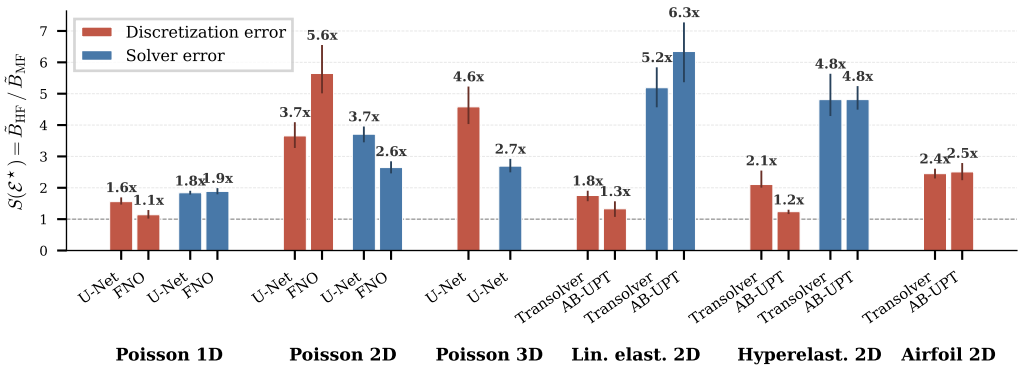


Figure 9: Cost-savings ratio $S(\mathcal{E}^\star)$ of multi-fidelity training relative to high-fidelity training. Relative L_2 error thresholds: $\mathcal{E}^\star = 10\%$ for Poisson, linear elasticity, and hyperelasticity, and $\mathcal{E}^\star = 2\%$ for the airfoil experiment (volume fields). Multi-fidelity training yields cost savings in every one of the 20 setups, up to $6.3\times$ with a mean of $\sim 3.1\times$. Bar heights are bootstrap medians and whiskers denote 95% confidence intervals, all excluding $1\times$. We describe the postprocessing and bootstrapping procedure in Section J.5.

directly use our existing fidelity axes, while the time step itself could form an additional fidelity axis, provided the architecture handles varying step sizes, which may raise challenges similar to those we observed for discretization error on unstructured meshes. As outlined in Section 3.1, the numerical pipeline also introduces *modeling error* (e.g., turbulence closures, boundary layer treatment), which offers additional fidelity axes that our framework could accommodate. In the context of foundation models, this motivates extending our scalar fidelity conditioning to a conditioning vector that captures the full simulation context, and studying multi-fidelity scaling in this richer, but compute intensive setting. Finally, while our approach already yields consistent improvements, more sophisticated distributions, for instance bimodal distributions, could be used for sampling. Specialized fidelity fusion methods beyond our conditioning, as well as training strategies that go beyond naïve dataset mixing, such as fidelity scheduling or pretraining and finetuning, also represent promising directions for future work.

Conclusion. Our empirical results consistently show that, at any finite data generation budget, training on optimal mixtures of simulation fidelities outperforms training on high-fidelity data alone, with the two approaches converging only at very large budgets. This highlights that our multi-fidelity savings are not unique to one setup, but hold across different neural surrogate architectures and error sources and, more importantly, across diverse physical domains encompassing different PDEs, spatial discretizations (finite differences, finite volumes, and finite elements) and numerical solvers. At a matched test error, multi-fidelity training reduces the required data generation budget by an average of $\sim 3.1\times$ across the 20 setups summarized in Fig. 9, with peaks up to $6.3\times$. Concretely, on the airfoil case this turns $\sim$2 days of high-fidelity data generation on a 64-core CPU into $\sim$19 hours.

Beyond raw cost savings, our experiments reveal a practical asymmetry: multi-fidelity over solver error consistently outperforms multi-fidelity over discretization on problems *requiring resolution invariance* and is also simpler to deploy since it changes only a single solver parameter rather than requiring a robust variable-resolution meshing pipeline. We therefore recommend the solver axis as the natural starting point for practitioners.

The largest multi-fidelity gains for the presented datasets arise in the low-to-medium budget regime ($\tilde{B} \approx 100$–500 equivalent high-fidelity samples), exactly where high-fidelity data is scarce and the amortization point determines whether a surrogate is cost-effective at all. At $\tilde{B} \approx 200$ on the airfoil case (AB-UPT), for example, multi-fidelity training reduces the relative lift coefficient error by 54.9% relative to the high-fidelity baseline. As the SciML community moves towards foundation-scale surrogates, simulation budgets increasingly dominate the total compute spend. Established high-fidelity surrogate datasets such as DrivAerML (Ashton et al., 2024), containing ~ 500 high-fidelity samples, sit in the regime where our approach delivers its largest gains. Treating sample fidelity as a first-class design variable rather than a single fixed setting offers a concrete and broadly applicable path to scaling these models at substantially lower data generation cost.

Acknowledgements

The authors thank Gianluca Galletti for the constructive discussions in the initial phase of the project.

The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. We thank the projects FWF AIRI FG 9-N (10.55776/FG9), AI4GreenHeatingGrids (FFG- 899943), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01), FWF Bilateral Artificial Intelligence (10.55776/COE12). We thank NXAI GmbH, Silicon Austria Labs (SAL), Merck Healthcare KGaA, GLS (Univ. Waterloo), TÜV Holding GmbH, Software Competence Center Hagenberg GmbH, dSPACE GmbH, TRUMPF SE + Co. KG. We acknowledge EuroHPC Joint Undertaking for access to MareNostrum5 at BSC, Spain, Vega at IZUM, Slovenia, Karolina at IT4Innovations, Czech Republic, MeluXina at LuxProvide, Luxembourg, and LEONARDO, hosted by CINECA (Italy) and the LEONARDO consortium. The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project b278bb. NHR funding is provided by federal and Bavarian state authorities. The authors also gratefully acknowledge the computational and data resources as well as the support provided by the Leibniz Supercomputing Centre. Felix Koehler and Nils Thuerey acknowledge funding from the Munich Center for Machine Learning (MCML).

References

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, 2024.

Benedikt Alkin, Maurits Bleeker, Richard Kurle, Tobias Kronlachner, Reinhard Sonnleitner, Matthias Dorfer, and Johannes Brandstetter. AB-UPT: scaling neural CFD surrogates for high-fidelity automotive aerodynamics simulations via anchored-branched universal physics transformers. *Trans. Mach. Learn. Res.*, 2025. URL https://openreview.net/forum?id=nwQ8nitlTZ.

Martin S. Alnæs, Anders Logg, Kristian B. Ølgaard, Marie E. Rognes, and Garth N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Trans. Math. Softw.*, 40(2), March 2014. ISSN 0098-3500. doi: 10.1145/2566630. URL https://doi.org/10.1145/2566630.

Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001. doi: 10.1137/S0895479899358194. URL https://doi.org/10.1137/S0895479899358194.

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Türkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda-Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Trans. Mach. Learn. Res.*, 2024. URL https://openreview.net/forum?id=gerNCVqqtR.

Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schonwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, Burkhard Hupertz, and Danielle C. Maddix. Drivaerml: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics. *CoRR*, abs/2408.11969, 2024. doi: 10.48550/ARXIV.2408.11969. URL https://doi.org/10.48550/arXiv.2408.11969.

Neil Ashton, Johannes Brandstetter, and Siddhartha Mishra. Fluid intelligence: A forward look on AI foundation models in computational fluid dynamics. *arXiv preprint arXiv:2511.20455*, 2025.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex: Zero-shot forecasting across long and short horizons with enhanced in-context learning. *CoRR*, abs/2505.23719, 2025. doi: 10.48550/ARXIV.2505.23719. URL https://doi.org/10.48550/arXiv.2505.23719.

Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and Garth N. Wells. Dolfinx: The next generation FEniCS problem solving environment, December 2025. URL https://doi.org/10.5281/zenodo.18101307.

Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1): 235–242, 2000. doi: 10.1093/nar/28.1.235.

Kanishk Bhatia, Felix Koehler, and Nils Thuerey. PRDP: progressively refined differentiable physics. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=9Fh0z1JmPU.

Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, Jayesh K. Gupta, Kit Thambiratnam, Alexander T. Archibald, Chun-Chieh Wu, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris. A foundation model for the earth system. *Nature*, 641(8065): 1180–1187, 2025. doi: 10.1038/S41586-025-09005-Y. URL https://doi.org/10.1038/s41586-025-09005-y.

Matthieu Boileau, Philippe Helluy, Jeremy Pawlus, and Svitlana Vyetrenko. Towards interpretable time series foundation models. *CoRR*, abs/2507.07439, 2025. doi: 10.48550/ARXIV.2507.07439. URL https://doi.org/10.48550/arXiv.2507.07439.

Florent Bonnet, Jocelyn Ahmed Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier-stokes solutions. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/94ab7b23a345f93333eac8748a66c763-Abstract-Datasets_and_Benchmarks.html.

Susanne C Brenner and L Ridgway Scott. *The mathematical theory of finite element methods*. Springer, 2008.

William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000.

N. Carey, L. Zanisi, S. Pamela, V. Gopakumar, J. Omotani, J. Buchanan, J. Brandstetter, Fabian Paischer, Gianluca Galletti, and Paul Setinek. Neural operator surrogate models of plasma edge simulations: feasibility and data efficiency. *Nuclear Fusion*, 65(10):106010, 2025. ISSN 1741-4326. doi: 10.1088/1741-4326/adfdfb. URL http://dx.doi.org/10.1088/1741-4326/adfdfb.

Alexandre Joel Chorin. Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762, 1968.

Naman Choudhary, Vedant Singh, Ameet Talwalkar, Nicholas Matthew Boffi, Mikhail Khodak, and Tanya Marwah. Pre-generating multi-difficulty pde data for few-shot neural pde solvers. *arXiv preprint arXiv:2512.00564*, 2025.

Russell M. Cummings, William H. Mason, Scott A. Morton, and David R. McDaniel. Applied computational aerodynamics: A modern engineering approach. In *Cambridge Aerospace Series*, pages 731–765. Cambridge University Press, 2015. doi: 10.1017/CBO9781107284166.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

J. P. Van Doormaal and G. D. Raithby. Enhancements of the simple method for predicting incompressible fluid flows. *Numerical Heat Transfer*, 7(2):147–163, 1984. doi: 10.1080/01495728408961817. URL https://doi.org/10.1080/01495728408961817.

Joel H Ferziger, Milovan Perić, and Robert L Street. *Computational Methods for Fluid Dynamics*. Springer, 4th edition, 2019.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah M. Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander J. Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. Datacomp: In search of the next generation of multimodal datasets. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/56332d41d55ad7ad8024aac625881be7-Abstract-Datasets_and_Benchmarks.html.

Wenhan Gao, Ruichen Xu, Yuefan Deng, and Yi Liu. Discretization-invariance? on the discretization mismatch errors in neural operators. In *The Thirteenth International Conference on Learning Representations*, 2025.

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. doi: https://doi.org/10.1002/nme.2579. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2579.

Sachin Goyal, Pratyush Maini, Zachary C. Lipton, Aditi Raghunathan, and J. Zico Kolter. Scaling laws for data filtering - data curation cannot be compute agnostic. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 22702–22711. IEEE, 2024. doi: 10.1109/CVPR52733.2024.02142. URL https://doi.org/10.1109/CVPR52733.2024.02142.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *CoRR*, abs/2306.11644, 2023. doi: 10.48550/ARXIV.2306.11644. URL https://doi.org/10.48550/arXiv.2306.11644.

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia de Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. doi: https://doi.org/10.1002/qj.3803. URL https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3803.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022. doi: 10.48550/ARXIV.2203.15556. URL https://doi.org/10.48550/arXiv.2203.15556.

Benjamin Holzschuh, Georg Kohl, Florian Redinger, and Nils Thuerey. P3d: Highly scalable 3d neural surrogates for physics simulations with global context. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8UdCE5nhFl.

Amanda A. Howard, Mauro Perego, George Em Karniadakis, and Panos Stinis. Multifidelity deep operator networks for data-driven and physics-informed problems. *J. Comput. Phys.*, 493: 112462, 2023. doi: 10.1016/J.JCP.2023.112462. URL https://doi.org/10.1016/j.jcp.2023.112462.

Wenjie Hu, Sidun Liu, Peng Qiao, Zhenglun Sun, and Yong Dou. Transolver is a linear transformer: Revisiting physics-attention through the lens of linear attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(1):408–416, Mar. 2026. doi: 10.1609/aaai.v40i1.37003. URL https://ojs.aaai.org/index.php/AAAI/article/view/37003.

Thomas JR Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Courier Corporation, 2012.

Su Jiang and Louis J. Durlofsky. Use of multifidelity training data and transfer learning for efficient construction of subsurface flow surrogate models. *J. Comput. Phys.*, 474:111800, 2023. doi: 10.1016/J.JCP.2022.111800. URL https://doi.org/10.1016/j.jcp.2022.111800.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Ryan Keisler. Forecasting global weather with graph neural networks, 2022. URL https://arxiv.org/abs/2202.07575.

Felix Koehler and Nils Thuerey. Neural emulator superiority: When machine learning for PDEs surpasses its training data. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/f200119a40846e508954abcd61f5f3fd-Abstract-Conference.html.

Nikola B. Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *J. Mach. Learn. Res.*, 22:290:1–290:76, 2021. URL https://jmlr.org/papers/v22/21-0806.html.

Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007. doi: 10.1137/1.9780898717839. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898717839.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need II: phi-1.5 technical report. *CoRR*, abs/2309.05463, 2023. doi: 10.48550/ARXIV.2309.05463. URL https://doi.org/10.48550/arXiv.2309.05463.

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. URL https://openreview.net/forum?id=c8P9NQVtmnO.

LLaMA-3 Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Lu Lu, Raphaël Pestourie, Steven G. Johnson, and Giuseppe Romano. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *CoRR*, abs/2204.06684, 2022. doi: 10.48550/ARXIV.2204.06684. URL https://doi.org/10.48550/arXiv.2204.06684.

Yanfang Lyu, Xiaoyu Zhao, Zhiqiang Gong, Xiao Kang, and Wen Yao. Multi-fidelity prediction of fluid flow based on transfer learning using fourier neural operator. *Physics of Fluids*, 35(7): 077118, 07 2023. ISSN 1070-6631. doi: 10.1063/5.0155555. URL https://doi.org/10.1063/5.0155555.

Florian Menter, M. Kuntz, and RB Langtry. Ten years of industrial experience with the sst turbulence model. *Heat and Mass Transfer*, 4, 01 2003.

Amil Merchant, Simon L. Batzner, Samuel S. Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023. doi: 10.1038/S41586-023-06735-9.

Mistral AI. Mistral large 2 release announcement. https://mistral.ai/news/mistral-large-2407, 2024.

Daniel Musekamp, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert. Active learning for neural PDE solvers. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=x4ZmQaumRg.

Thao Nguyen, Gabriel Ilharco, Mitchell Wortsman, Sewoong Oh, and Ludwig Schmidt. Quality not quantity: On the interaction between dataset design and robustness of CLIP. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/86a8a512b27f49519594ebe89f66d708-Abstract-Conference.html.

Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K. Gupta, and Aditya Grover. Climax: A foundation model for weather and climate. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 25904–25938. PMLR, 2023. URL https://proceedings.mlr.press/v202/nguyen23a.html.

Fabian Paischer, Gianluca Galletti, William Hornsby, Paul Setinek, Lorenzo Zanisi, Naomi Carey, Stanislas Pamela, and Johannes Brandstetter. Gyroswin: 5d surrogates for gyrokinetic plasma turbulence simulations. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 02 - 07, 2025*, 2025.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *CoRR*, abs/2202.11214, 2022. URL https://arxiv.org/abs/2202.11214.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 4172–4182. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00387. URL https://doi.org/10.1109/ICCV51070.2023.00387.

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.

Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter W. Battaglia, Rémi R. Lam, and Matthew Willson. Probabilistic weather forecasting with machine learning. *Nature*, 637 (8044):84–90, 2025. doi: 10.1038/S41586-024-08252-9. URL https://doi.org/10.1038/s41586-024-08252-9.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, Lecture Notes in Computer Science, pages 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://doi.org/10.1007/978-3-319-24574-4_28.

Noveen Sachdeva, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Lichan Hong, Ed H. Chi, James Caverlee, Julian J. McAuley, and Derek Zhiyuan Cheng. How to train data-efficient llms. *CoRR*, abs/2402.09668, 2024. doi: 10.48550/ARXIV.2402.09668. URL https://doi.org/10.48550/arXiv.2402.09668.

Mansi Sakarvadia, Kareem Hegazy, Amin Totounferoush, Kyle Chard, Yaoqing Yang, Ian Foster, and Michael W. Mahoney. The false promise of zero-shot super-resolution in machine-learned operators. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=hkF7ZM7fEp.

Matthew W. Scroggs, Igor A. Baratta, Chris N. Richardson, and Garth N. Wells. Basix: a runtime finite element basis evaluation library. *Journal of Open Source Software*, 7(73):3982, 2022a. doi: 10.21105/joss.03982. URL https://doi.org/10.21105/joss.03982.

Matthew W. Scroggs, Jørgen S. Dokken, Chris N. Richardson, and Garth N. Wells. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Trans. Math. Softw.*, 48(2), May 2022b. ISSN 0098-3500. doi: 10.1145/3524456. URL https://doi.org/10.1145/3524456.

Paul Setinek, Gianluca Galletti, and Johannes Brandstetter. Towards multi-fidelity scaling laws of neural surrogates in CFD. *CoRR*, abs/2511.01830, 2025. doi: 10.48550/ARXIV.2511.01830. URL https://doi.org/10.48550/arXiv.2511.01830.

Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018. URL http://arxiv.org/abs/1803.09820.

Dong H. Song and Daniel M. Tartakovsky. Transfer learning on multi-fidelity data. *CoRR*, abs/2105.00856, 2021. URL https://arxiv.org/abs/2105.00856.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/7b75da9b61eda40fa35453ee5d077df6-Abstract-Conference.html.

Shashank Subramanian, Alexander Kiefer, Arnur Nigmetov, Amir Gholami, Dmitriy Morozov, and Michael W. Mahoney. On neural scaling laws for weather emulation through continual training. *CoRR*, abs/2603.25687, 2026. doi: 10.48550/ARXIV.2603.25687. URL https://doi.org/10.48550/arXiv.2603.25687.

Bane Sullivan and Alexander Kaszynski. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, 4(37):1450, May 2019. doi: 10.21105/joss.01450. URL https://doi.org/10.21105/joss.01450.

Hewei Tang, Qingkai Kong, and Joseph P Morris. Multi-fidelity fourier neural operator for fast modeling of large-scale geological carbon storage. *Journal of Hydrology*, 629:130641, 2024.

Xiao Wang, Siyan Liu, Aristeidis Tsaris, Jong-Youl Choi, Ashwin M. Aji, Ming Fan, Wei Zhang, Junqi Yin, Moetasim Ashfaq, Dan Lu, and Prasanna Balaprakash. ORBIT: oak ridge base foundation model for earth system predictability. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2024, Atlanta, GA, USA, November 17-22, 2024*. IEEE, 2024. doi: 10.1109/SC41406.2024.00007. URL https://doi.org/10.1109/SC41406.2024.00007.

H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620–631, 1998. doi: 10.1063/1.168744.

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.

Han Yang, Chenxi Hu, Yichi Zhou, Xixian Liu, Yu Shi, Jielan Li, Guanzhi Li, Zekun Chen, Shuizhou Chen, Claudio Zeni, Matthew Horton, Robert Pinsler, Andrew Fowler, Daniel Zügner, Tian Xie, Jake Smith, Lixin Sun, Qian Wang, Lingyu Kong, Chang Liu, Hongxia Hao, and Ziheng Lu. Mattersim: A deep learning atomistic model across elements, temperatures and pressures. *arXiv preprint arXiv:2405.04967*, 2024.

Yuejiang Yu, Langwen Huang, Alexandru Calotoiu, and Torsten Hoefler. Scaling laws of global weather models. *CoRR*, abs/2602.22962, 2026. doi: 10.48550/ARXIV.2602.22962. URL https://doi.org/10.48550/arXiv.2602.22962.

Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Zilong Wang, Aliaksandra Shysheya, Jonathan Crabbé, Shoko Ueda, et al. A generative model for inorganic materials design. *Nature*, pages 1–3, 2025.

Dichang Zhang, Zexia Zhang, Christian Santoni, Ali Khosronejad, and Dimitris Samaras. Transfer learning in multi-fidelity surrogate modeling: A wind farm case. In *ICML 2024 AI for Science Workshop*, 2024. URL https://openreview.net/forum?id=yBTDCqNcan.

O. C. Zienkiewicz and J. Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. part 1: The recovery technique. *International Journal for Numerical Methods in Engineering*, 33(7):1331–1364, 1992. doi: https://doi.org/10.1002/nme.1620330702. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620330702.

O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method: Its Basis and Fundamentals*. The Finite Element Method. Butterworth-Heinemann, 2013. ISBN 9780080951355. URL https://books.google.at/books?id=HUsHnUjzEXYC.

Appendix

A Reproducibility Statement

In this appendix, we elaborate on the experimental details of our experiments. Specifically, we extend upon our used formalism for multi-fidelity training in Section C. In Sections D to G we discuss problem formulations for the Poisson, linear elasticity, hyperelasticity, and airfoil experiments, respectively. This includes a description of the numerical methods used in data generation and precisely how the discretization and solver error is augmented in each case, including a discussion of their characteristic behaviors. In Sections H and I, we discuss architecture configurations and training details as well as details on the chosen fidelity menus. In Sections J.1 to J.4 we present all data from our extensive combinatorial studies including isobudget curves and derived scaling behavior. We supplement this with numeric tables that allow for reproducing data generation compute savings or error improvement claims made in the main part. Additionally, we provide ablations in Section L supporting the design choices made throughout this work.

Our full set of experiments exclusively relies on open-source software. Links to the source code are found on the first page.

B AI Disclosure Statement

LLM systems have been involved at various stages of this research. Coding agents (Claude Opus 4.8 to Fable 5.1) were used for workflow orchestration (SLURM job submissions, data aggregation, improvements to visualizations, etc.). Core algorithmic code was written in collaboration with AI agents and completion systems but has always received full human oversight. LLMs (mostly Claude Opus 4.8, 5, and Gemini Pro 3.1) have also been used for light editing of the manuscript. The core text is human written. Some additional citations in the literature review were found and analyzed using LLM tools. The authors read and verified every cited work. Finally, before submission, we gathered feedback on the draft from several AI systems (Claude Opus 4.8, Gemini 3.1 Pro, Grok 3, GPT-5.4, and Mistral Medium 3.5). The authors take full responsibility for all content of this paper.

C Extended Formalism

C.1 Designing the fidelity support

The accuracy oracle. As introduced in Section 3.1, we define the relative L_2 error of our simulator at a certain fidelity via $e(\xi)$, defined in Eq. (2). To be precise, this is an *accuracy oracle*.[3] Asymptotically, the behavior $e(\xi)$ will match the theoretical error convergence of the underlying problem. In infinite precision, we expect

$$\lim_{\xi \to \infty} e(\xi) = 0. \tag{8}$$

Realistically, though, we will have

$$\lim_{\xi \to \infty} e(\xi) \approx \epsilon_{\text{mach}}, \tag{9}$$

in which ϵ_{mach} is the noise floor of the considered floating point accuracy. Denote by ξ_ϵ the fidelity level at which we first reach the machine precision. This gives a first hard ceiling on Ξ, namely $\xi_{\max} \leq \xi_\epsilon$. However we discuss practical rules for the floor and ceiling of Ξ below, which we will see are much tighter.

Choosing the right support. The support Ξ is itself a hyperparameter of the multi-fidelity procedure, and it cannot be optimized away by a subsequent choice of λ. The attainable per-sample error and the minimum per-sample cost are bounded below by Ξ alone,

$$\min_{\xi \in \text{supp}(p)} e(\xi) \;\geq\; e\big(\max(\Xi)\big), \qquad \min_{\xi \in \text{supp}(p)} c(\xi) \;\geq\; c\big(\min(\Xi)\big), \tag{10}$$

[3] In practice, we might realize this via sufficiently high ξ that we consider a good reference. This can be the highest fidelity $\xi_{\max}$. In our experiments, this varies based on feasibility. For more details, see Sections H and I.

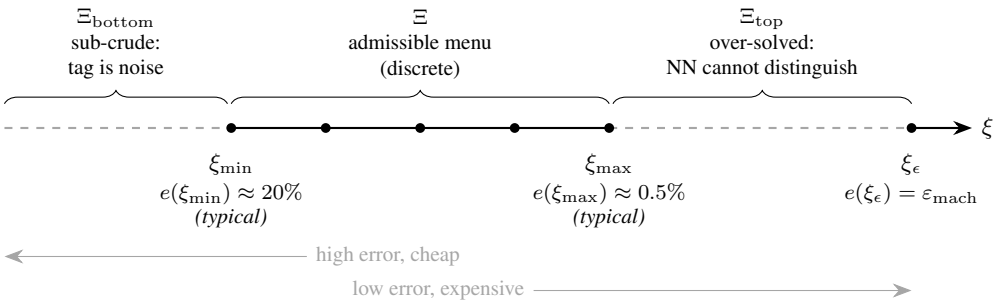


Figure 10: Schematic arrangement of the fidelity menu Ξ on the fidelity axis. The admissible support (solid segment, five illustrative dots) sits between the two excluded tails Ξ_{bottom} (high-error, sub-crude: the fidelity tag carries no signal) and Ξ_{top} (low-error, over-solved relative to the surrogate floor). The machine-precision fidelity ξ_ϵ defines the far end of Ξ_{top}. Concrete error values on the two endpoints are the *typical* targets we use in practice ($e(\xi_{\min}) \approx 20\%$ and $e(\xi_{\max}) \approx 0.5\%$).

so Ξ defines the feasibility envelope within which λ navigates. A mixture experiment trained over a poorly chosen Ξ cannot be rescued by subsequent tuning of λ. We therefore treat choosing the endpoints of Ξ as an important design choice.

It is convenient to name the two regions the menu deliberately excludes. On the high-error side lies

$$\Xi_{\text{bottom}} \;=\; \{\, \xi \,:\, \xi < \xi_{\min} \,\}, \tag{11}$$

where the solver is cruder than our lowest anchor and its outputs are noise-dominated and do not provide meaningful learning signal. On the low-error side lies

$$\Xi_{\text{top}} \;=\; \{\, \xi \,:\, \xi_{\max} < \xi \le \xi_\epsilon \,\}, \tag{12}$$

where the solver is *more* accurate than the surrogate's training floor. Figure 10 illustrates the arrangement schematically. We now discuss how to position Ξ between Ξ_{top} and Ξ_{bottom} in practice. In Section L.6, we investigate the placement of the fidelity menu ablationally.

Top anchor $\xi_{\max}$: slightly below the surrogate floor. Choose $\xi_{\max} := \max(\Xi)$ so that the simulator's error $e(\xi_{\max})$ is well below the test error the surrogate is expected to achieve, but not so far below that additional fidelity yields no learnable signal. If $e(\xi_{\max})$ is too low, this would *not* lower the test error because the surrogate cannot distinguish simulations at $\xi_{\max}$ from those at slightly cheaper fidelities. The probability mass placed there is wasted computation (Bhatia et al., 2025). In practice, this is hard to define a priori without an extensive dataset generation and fidelity sweep. As a proxy, we note that state-of-the-art neural surrogates achieve relative L_2 errors in the range of $10^{-1} - 10^{-3}$ for settings similar to our experiments (Li et al., 2021; Bonnet et al., 2022; Wu et al., 2024; Alkin et al., 2025). Therefore, we target $e(\xi_{\max}) = 0.5\%$ across all our experiments. This places $\xi_{\max}$ far below ξ_ϵ, showing that the surrogate exhausts its capacity long before the solver exhausts its floating-point budget.

Bottom anchor $\xi_{\min}$: crude but informative. Choose $\xi_{\min} := \min(\Xi)$ at the lowest fidelity whose output still carries a measurable signal about the input-to-output map, typically

$$e(\xi_{\min}) \;\in\; [1 \times 10^{-1},\, 3 \times 10^{-1}]. \tag{13}$$

Two failure modes bracket the correct choice. If $e(\xi_{\min})$ is too high, the solver output is noise-dominated and provides no learnable signal. If $e(\xi_{\min})$ is too low, $\xi_{\min}$ is too close to $\xi_{\max}$ in error space. As such, the $\lambda < 0$ regime then yields many almost-high-fidelity samples rather than genuinely cheap ones, eliminating the cost-quality tradeoff that motivates multi-fidelity training in the first place. Empirically, $e(\xi_{\min}) \approx 20\%$ sits comfortably between these regimes across all our experiments.

Axis asymmetry. The two fidelity regimes introduced in Section 3.1 reach the top anchor very differently. For axes with *exponentially linear* error scaling (iterative solvers), over-solving is cheap and the risk is placing $\xi_{\max}$ too far above the surrogate floor, wasting a long tail of the menu on cells the optimizer cannot use. For axes with *algebraic* error scaling (spatial discretization), over-solving is structurally expensive. Here, the largest practically reachable ξ is often the correct $\xi_{\max}$, even when $e(\xi_{\max})$ sits above the surrogate floor.

C.2 Fidelity normalization and clamping

To ensure consistent behavior across different fidelity menus we use the following normalization:

$$\hat{\xi}(\xi) = \frac{e(\xi_{\min}) - e(\xi)}{e(\xi_{\min}) - e(\xi_{\max})} \in [0, 1], \tag{14}$$

where $e(\xi)$ follows the asymptotic convergence of the respective error source (*algebraic* or *exponential*).[4]

C.3 Budgeted sampler

We draw fidelities i.i.d. from the categorical distribution induced by $p(\cdot; \lambda, \Xi)$ until the cumulative cost crosses the budget:

$$M(\lambda, B) = \min \left\{ m \in \mathbb{N} : \sum_{j=1}^{m} c(\xi^{(j)}) \geq B \right\}, \tag{15}$$

$$G(\lambda, B) = \left\{ \xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(M)} \right\}. \tag{16}$$

Both M and G are random (via the fidelity draws); the realized sample count satisfies $\mathbb{E}[M] \approx B / \mathbb{E}_{\xi \sim p}[c(\xi)]$, which grows as λ decreases. In other words, λ allows trading fidelity in the sample for sample counts.

In some scenarios, we use a *greedy backfill strategy*. This means that we try to fill up leftover budget with cheap samples to ensure we approximately meet budget B in each scenario.

Pairing each fidelity with an independently drawn input $\mathbf{f}_h^{(j)} \sim \mathcal{F}_h$ gives the training dataset

$$D(\lambda, B) = \left\{ \left(\mathbf{f}_h^{(j)},\ \mathbf{u}_h^{(j)},\ \xi^{(j)} \right) \right\}_{j=1}^{M}, \qquad \mathbf{u}_h^{(j)} = \mathcal{P}_h\left(\mathbf{f}_h^{(j)}; \xi^{(j)} \right). \tag{17}$$

Crucially, each output function $\mathbf{u}_h^{(j)}$ in the dataset is produced at its corresponding fidelity $\xi^{(j)}$.

C.4 Training objective

For a problem with F output fields (=vector components or channels), let $\mathbf{u}_h^{(f)}$ denote the f-th field of the simulator's ground truth at fidelity ξ (generated as $\mathbf{u}_h = \mathcal{P}_h(\mathbf{f}_h; \xi)$), and $\tilde{\mathbf{u}}_h^{(f)} = \mathcal{S}_\theta(\mathbf{f}_h; \hat{\xi})^{(f)}$ the corresponding surrogate prediction conditioned on the normalized fidelity $\hat{\xi}$.

For grid-based experiments (Section 5.1), we use the standard mean squared error on normalized predictions and ground truth:

$$\zeta\left(\tilde{\mathbf{u}}_h, \mathbf{u}_h \right) \ = \ \frac{1}{N} \left\| \tilde{\mathbf{u}}_h - \mathbf{u}_h \right\|_2^2, \tag{18}$$

where N is the total number of grid points.

For mesh-based experiments (Sections 5.2 and 5.3) the training loss is the field-averaged relative L_2 error:

$$\zeta\left(\tilde{\mathbf{u}}_h, \mathbf{u}_h \right) \ = \ \frac{1}{F} \sum_{f=1}^{F} \frac{\left\| \tilde{\mathbf{u}}_h^{(f)} - \mathbf{u}_h^{(f)} \right\|_2}{\left\| \mathbf{u}_h^{(f)} \right\|_2}. \tag{19}$$

Single-field problems (e.g. structural mechanics, where only the scalar von Mises stress is predicted) reduce to $F = 1$ and recover the standard relative L_2 error.

C.5 Test error

We use the same error as defined in Eq. (19) in physical (denormalized) space.

[4] Note that to estimate $e(\xi)$ we do not need the $\xi_{\max}$ solution of that specific instance. Rather, we can estimate it based on the error convergence relation of the problem at hand.

Choice of test reference. We use $\mathcal{P}_h(\cdot;\xi_{\max})$ as the reference rather than $\lim_{\xi\to\infty}\mathcal{P}_h(\cdot;\xi)$, so that the residual $e(\xi_{\max})$ is absorbed into the definition of "ground truth" and does not contaminate the measurement. The paper's central quantity is therefore *the surrogate's approximation to the best-in-support label*, isolating the effect of the training-data mixture from the fixed fidelity-ceiling bias. Replacing the reference with the $\xi\to\infty$ (practically ξ_ϵ) limit yields a physically stricter metric but introduces a λ-independent floor of size $e(\xi_{\max})$; which would not change the ranking of mixtures. We believe our choice to be practical, since the surrogate generation process naturally inherits some form of errors (Koehler and Thuerey, 2025). In Section L.7 we ablate the influence of changing from $\xi_{\max}$ to $\xi\to\infty$.

D Poisson Experiments

In this section, we explain the details of the Poisson experiment in 1D, 2D, and 3D. We introduce the continuous description with homogeneous Dirichlet boundaries, the discretization via finite difference stencils, and the solution of the arising linear system by a geometric multigrid scheme.

D.1 Overview of the Two Types of Errors

We consider the Poisson equation in $D\in\{1,2,3\}$ dimensions on $\Omega=(0,L)^D$ with homogeneous Dirichlet boundary conditions

$$\Delta u=-f \quad \text{on } \Omega, \qquad u|_{\partial\Omega}=0. \tag{20}$$

In a continuous setting, let Δ^{-1} be the exact solution to the *continuous* problem. Then, we could abstractly write

$$u=\Delta^{-1}(-f). \tag{21}$$

Now we introduce a collection of interior points that make up an equidistant Cartesian grid. Let R_N be the restriction operator that maps a function u into a discrete representation $\mathbf{u}_h$ evaluated on these grid points. Now, let $\mathbf{\Delta}_h$ be the three-point, five-point, or seven-point stencil in 1D, 2D, or 3D, respectively (LeVeque, 2007). It shall correctly respect the boundary condition. Then, $\mathbf{\Delta}_h^{-1}$ is the *exact* solution to the discrete approximation (i.e., the exact solution to the linear system of equations). We know based on the consistency error that

$$\|\mathbf{\Delta}_h^{-1}(R_N(f))-R_N(\Delta^{-1}(f))\|_\infty=\mathcal{O}(1/N^2). \tag{22}$$

This is the *spatial discretization error*. We can realize $\mathbf{\Delta}_h^{-1}$ via the discrete sine transform, a direct linear solver, or a converged iterative solver. Similarly, $R_N\circ\Delta^{-1}$ can be computed analytically, via a spectral solve, or given by a proxy with sufficiently high N.

Now, let Φ be the iterator associated with a reasonable multigrid solver (described in Section D.3) whose repeated application approximates the inverse *discrete* Laplacian, i.e.,

$$\underbrace{(\Phi_h\circ\Phi_h\circ\cdots\circ\Phi_h)}_{k\text{ times}}(\mathbf{f}_h)=\Phi_h^k(\mathbf{f}_h)\approx\mathbf{\Delta}_h^{-1}(\mathbf{f}_h). \tag{23}$$

Its asymptotic behavior is

$$\|\Phi_h^k(R_N(f))-\mathbf{\Delta}_h^{-1}(R_N(f))\|_\infty=\mathcal{O}(\rho^k). \tag{24}$$

In summary, when discretely resolving the Poisson equation on a computer, there are two sources of error. First, there is the spatial discretization error which scales polynomially quadratically with the number of points *per dimension* N. The second source is the solver error, which in our case is based on the iterative truncation error of a multigrid scheme. This one scales exponentially linearly with convergence rate ρ over iterations k.

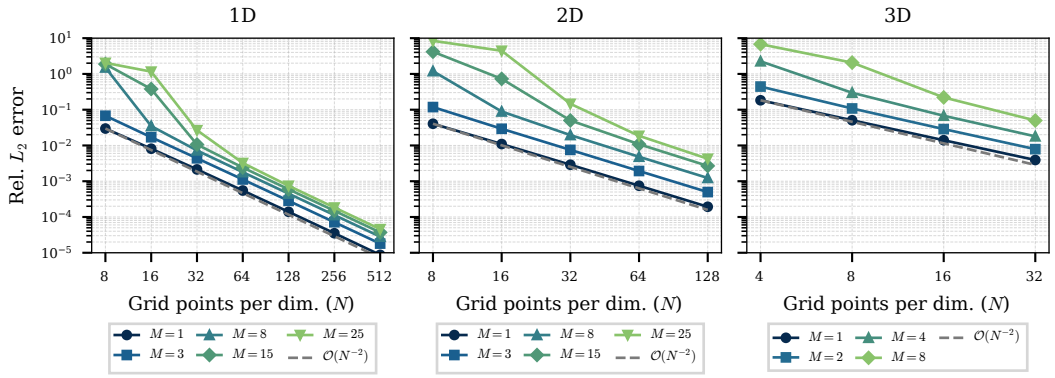


Figure 11: Discretization error convergence for the Poisson problem on a uniform Cartesian grid in 1D, 2D, and 3D. Asymptotically, each configuration reaches the polynomially quadratic decay in the optimality error.

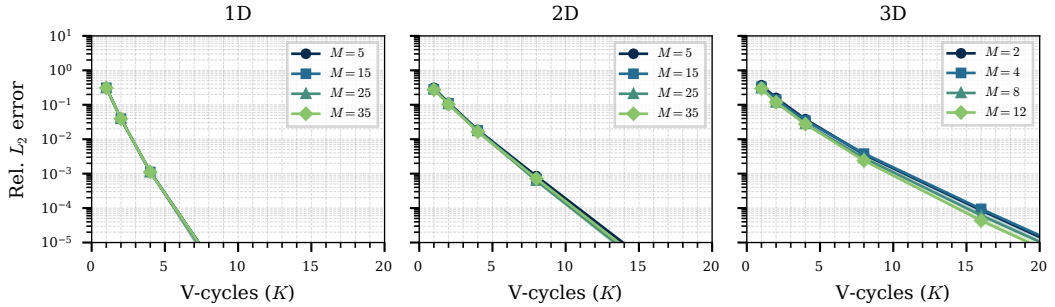


Figure 12: Multigrid V-cycle iterative truncation error for 1D, 2D, and 3D. Each plot uses a fixed resolution of 64 *per dimension* (i.e., 64^D total degrees of freedom). Different colors indicate different numbers of active modes in the right-hand side distribution.

D.2 Distribution of Right-Hand Sides

While the asymptotic behavior of the spatial discretization error and the multigrid truncation error is polynomially quadratically and exponentially linear, respectively, the concrete convergence may be affected by the distribution of right-hand sides $f \sim \mathcal{F}$. Its spectral composition influences the pre-asymptotic regime (which oftentimes is slower than the asymptotic one) and the error offset. For this experiment we look at distributions $\mathcal{F}_h$ that produce "spectrally truncated white noise" with the first M modes being active. In higher dimensions, we use the hypercube of all modes such that their $\|\mathbf{k}\|_1 < M$. As an example, we show the convergence behavior over the spatial discretization error in Fig. 11 over a range of discretization points per dimension N in 1D, 2D, and 3D with different number of active modes. In Fig. 12 we display the iterative convergence of the multigrid scheme.

D.3 Geometric Multigrid V-cycle

The truncation fidelity axis uses $\xi = K$, the number of V-cycles applied to an initial zero guess:

$$\Phi_h^K(\mathbf{f}_h) = \underbrace{(\Phi_h \circ \cdots \circ \Phi_h)}_{K \text{ times}}(\mathbf{0}). \tag{25}$$

A single V-cycle Φ_h is a standard geometric multigrid scheme (Briggs et al., 2000) that solves $A_h \mathbf{u}_h = \mathbf{f}_h$ with $A_h = -\boldsymbol{\Delta}_h$ and homogeneous Dirichlet boundary conditions enforced by zero-padding throughout. One dimension-generic implementation services $D \in \{1, 2, 3\}$, with D inferred from the array shape.

Smoother and transfers. On every level we use red–black Gauss–Seidel with $\nu_1 = \nu_2 = 2$ pre- and post-smoothing sweeps, and $\nu_c = 50$ sweeps at the coarsest level (in lieu of a direct

solver). Restriction uses separable full-weighting (the $[1, 2, 1]/4$ stencil applied along each axis), and prolongation uses multilinear interpolation. Fine and coarse interior grids are non-nested under the "$N + 1$ intervals" convention, so prolongation evaluates the multilinear interpolant at fractional coarse-padded coordinates rather than using straight injection.

Level hierarchy. The hierarchy of J levels is generated by full coarsening, i.e., halving every axis while all current axis lengths are even and ≥ 8. At the $N = 64$ resolution used throughout, this yields $J = 5$ levels of shape $(64, 32, 16, 8, 4)$.

Initial guess and cost model. Each truncation-fidelity sample restarts from $\mathbf{u}_h = \mathbf{0}$, so Φ_h^K is a pure function of $(\mathbf{f}_h, h, K)$ with no warm-start state between samples. Per-call FLOPs depend only on the (fixed) grid shape and on the MG parameters (ν_1, ν_2, ν_c, J), so we take $c(K) = K$ and report budgets in units of V-cycles.

E Linear elasticity experiment

This section describes the linear elasticity simulations, which model a perforated structural steel plate under mechanical loading, often found in various civil and mechanical engineering use cases. We first describe the underlying governing equations and geometrical domain with boundary conditions in Section E.1 and Section E.2 respectively. Finally, we provide a detailed discussion of the errors arising in the numerical data generation pipeline in Section E.4.

E.1 Governing equations

The equations of static linear elasticity for the displacement field $\mathbf{u}$ in the absence of body forces read:

$$\begin{aligned} -\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}) &= \mathbf{0} && \text{in } \Omega \\ \boldsymbol{\sigma}(\mathbf{u}) &= \lambda \operatorname{tr}(\boldsymbol{\varepsilon}(\mathbf{u}))\, \mathbf{I} + 2\mu\, \boldsymbol{\varepsilon}(\mathbf{u}) \\ \boldsymbol{\varepsilon}(\mathbf{u}) &= \frac{1}{2}\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^\top\right), \end{aligned} \tag{26}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor, related to the displacement through Hooke's Law for isotropic materials, and $\boldsymbol{\varepsilon}$ is the linearized strain tensor. Because we model a thin plate, we employ the plane-stress formulation. The Lamé parameters, λ and μ are related to Young's modulus E and Poisson's ratio ν via

$$\mu = \frac{E}{2(1+\nu)}, \qquad \lambda = \frac{2\mu\nu}{1-\nu}. \tag{27}$$

The system is subject to the following boundary conditions:

$$\begin{aligned} \mathbf{u} &= \mathbf{0} && \text{on } \Gamma_D, \\ \boldsymbol{\sigma}(\mathbf{u}) \cdot \mathbf{n} &= \mathbf{t} && \text{on } \Gamma_N, \end{aligned} \tag{28}$$

where Γ_D and Γ_N are the Dirichlet and Neumann boundaries respectively, $\mathbf{n}$ is the outward-pointing unit normal and $\mathbf{t}$ is the prescribed surface traction.

This system is cast into a weak variational form and solved computationally using the FEM (for details, see (Hughes, 2012; Zienkiewicz and Taylor, 2013)).

E.2 Domain and boundary conditions

In order to introduce geometric variability, we define Ω as a 2D plate with randomly placed circular holes. The plate is clamped on the left edge, and a random traction vector is applied to the right edge. Fig. 13 shows a schematic overview of our domain and Table 3 summarizes the sampling ranges for geometric variations and boundary conditions. As material coefficients, we use Young's modulus $E = 210\,\text{GPa}$ and Poisson's ratio $\nu = 0.3$, representative of standard structural steel (e.g., ASTM A36 or S235).

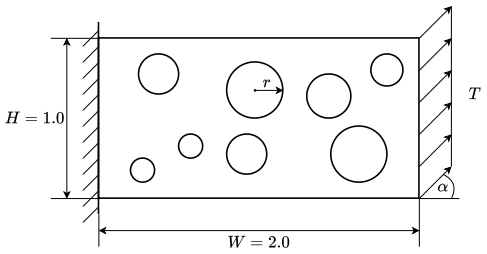


Figure 13: Perforated steel plate, clamped on the left and traction on the right boundary.

Table 3: Sampling ranges for boundary conditions and geometry parameters for the linear elasticity experiment.

Parameter	Symbol	Range
Traction magnitude	T	10^5–10^7 Pa
Traction angle	α	0–2π
Number of holes	–	5–20
Hole radii	r	0.02–0.15

E.3 Simulation and postprocessing

Simulation. We solve the weak form of Eq. (26) using linear (P1) Lagrange elements on triangular meshes, generated with `gmsh` (Geuzaine and Remacle, 2009) via a parametric meshing script with a single continuous parameter controlling mesh fidelity. The simulation uses the FEniCSx finite-element framework (Baratta et al., 2025; Alnæs et al., 2014; Scroggs et al., 2022a,b) with PETSc-based linear solvers. For the discretization error experiments, we eliminate solver error by using a direct LU solver (MUMPS (Amestoy et al., 2001)). For the solver-error experiments, we hold the mesh fixed and use a symmetric Gauss-Seidel iterative scheme to control solver fidelity (Section E.4 below).

Postprocessing. With P1 elements, the displacement field $\mathbf{u}$ lives natively on mesh nodes, but the stress $\boldsymbol{\sigma}(\mathbf{u})$ and strain $\boldsymbol{\varepsilon}(\mathbf{u})$ fields, being derivatives of $\mathbf{u}$, are piecewise constant and discontinuous across elements. For neural surrogate training we require nodal values, so we project these fields onto the continuous P1 space via L_2 projection. This projection acts as a smoothing operation, closely related to superconvergent patch recovery techniques (Zienkiewicz and Zhu, 1992). The von Mises stress, derived from $\boldsymbol{\sigma}$, is similarly projected to P1. Fields are then stored on the mesh nodes in HDF5 format, together with engineered geometric features (signed distance to the domain boundary, surface normals, nodal-equivalent surface tractions on the loaded edge) used as additional surrogate inputs.

E.4 Sources of error

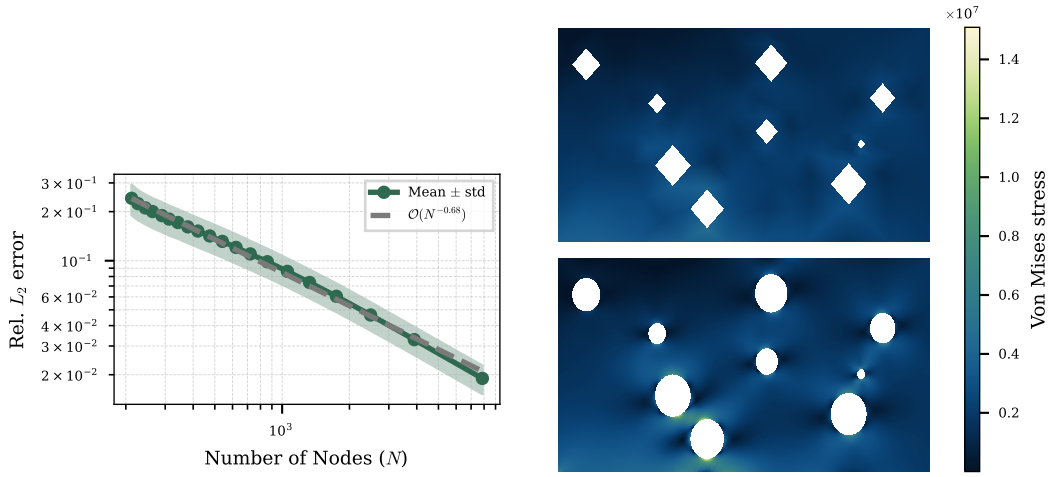
Discretization error. Let h denote a measure of the spatial discretization, i.e. the average edge length and take the definition of error from Eq. (2). For P1 elements (piecewise linear basis) on triangular meshes, the displacement error scales as $e(h) = \mathcal{O}(h^2)$ (Brenner and Scott, 2008, Chapter 5). Since the number of mesh nodes N scales as $N \propto h^{-2}$ in 2D, this gives $e(N) = \mathcal{O}(N^{-1})$. However, this optimal rate only applies to the displacement field $\mathbf{u}$. Theoretically, for the strain $\boldsymbol{\varepsilon}(\mathbf{u})$ and stress $\boldsymbol{\sigma}(\mathbf{u})$ fields, being derivatives of the displacement, the error converges slower: $e(N) = \mathcal{O}(N^{-1/2})$. The smoothing described in Section F.4 partially recovers the lost order of convergence, causing the convergence of the projected von Mises stress field to lie between $\mathcal{O}(N^{-1/2})$ and $\mathcal{O}(N^{-1})$. Our simulation and postprocessing setup leads to an empirical convergence rate of $e(N) = \mathcal{O}(N^{-0.68})$ as shown in Fig. 14a. Fig. 14b shows von Mises stress fields at the lowest and highest discretization fidelities for an example simulation.

Solver error. After spatial discretization of the weak form, the problem reduces to a symmetric positive-definite linear system

$$\mathbf{K}\mathbf{u}_h = \mathbf{f}_h, \tag{29}$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the assembled stiffness matrix and $\mathbf{f}_h$ is the assembled load vector. To study solver error, we hold the mesh fixed and replace the direct LU solver with a symmetric Gauss-Seidel iterative scheme.[5] Starting from a zero initial guess $\mathbf{u}_h = \mathbf{0}$, the relative L_2 error exhibits linear convergence: $e(k) = \mathcal{O}(\rho^k)$, with k being the number of iterations and $\rho \in (0, 1)$ is the spectral radius of the Gauss-Seidel iteration matrix, which depends on the problem and solver settings. Our

[5] Implemented via PETSc's Richardson iteration with symmetric successive over-relaxation (SSOR) at relaxation factor $\omega = 1.0$, mathematically equivalent to symmetric Gauss-Seidel.



(a) Empirical von Mises stress error convergence with number of mesh nodes N. We show the mean and ± 1 std across 10,000 simulations.

(b) Visualization of the von Mises stress field at the lowest and highest mesh resolution for an example simulation.

Figure 14: Discretization convergence for the linear elasticity experiment.

setup results in $\rho \approx 0.99994$. Fig. 15 shows the empirical convergence and Fig. 16 visualizes the von Mises stress field at the lowest and highest solver fidelities.

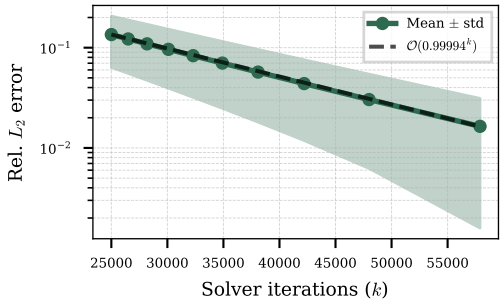


Figure 15: Empirical von Mises stress error convergence with iteration count k for the linear elasticity solver error experiment. We show the mean and ± 1 std across 90,000 simulations.

F Hyperelasticity experiment

This section describes the hyperelasticity simulations, which model a perforated elastomeric membrane under tensile loading, representative of components found in e.g., soft-robotic actuators or biomedical seals. We first describe the underlying governing equations and geometrical domain with boundary conditions in Section F.1 and Section F.2 respectively. We then describe the simulation pipeline in Section F.3. Finally, we provide a detailed discussion of the errors arising in the numerical data generation pipeline in Section F.4.

F.1 Governing equations

We consider a compressible Neo-Hookean hyperelastic material in the static regime, without body forces. Let $\mathbf{u}$ denote the displacement field, $\mathbf{F} = \mathbf{I} + \nabla \mathbf{u}$ the deformation gradient, $\mathbf{C} = \mathbf{F}^\top \mathbf{F}$ the right Cauchy–Green tensor, and $J = \det(\mathbf{F})$ the volume ratio. The strain energy density is given by

$$\psi(\mathbf{F}) = \frac{\mu}{2}\big(\mathrm{tr}(\mathbf{C}) - 2\big) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2, \tag{30}$$

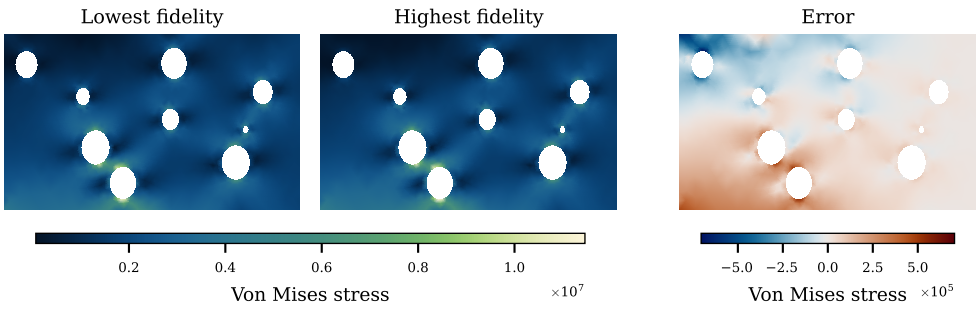


Figure 16: Visualization of the von Mises stress field at the lowest and highest solver fidelities for an example simulation.

where μ and λ are the Lamé parameters, related to Young's modulus E and Poisson's ratio ν via

$$\mu = \frac{E}{2(1+\nu)}, \qquad \lambda = \frac{2\mu\nu}{1-\nu}. \tag{31}$$

The first Piola–Kirchhoff stress tensor is obtained as $\mathbf{P} = \partial\psi/\partial\mathbf{F}$, and the Cauchy stress tensor follows from $\boldsymbol{\sigma} = J^{-1}\mathbf{P}\mathbf{F}^\top$. The equilibrium equation in the reference configuration reads

$$-\nabla \cdot \mathbf{P}(\mathbf{u}) = \mathbf{0} \quad \text{in } \Omega, \tag{32}$$

subject to boundary conditions

$$\begin{aligned} \mathbf{u} &= \mathbf{0} \quad \text{on } \Gamma_D, \\ \mathbf{P}(\mathbf{u}) \cdot \mathbf{n} &= \mathbf{t} \quad \text{on } \Gamma_N. \end{aligned} \tag{33}$$

The problem is inherently nonlinear due to the dependence of $\mathbf{P}$ on $\nabla\mathbf{u}$. A weak form can be derived and solved via the FEM (Hughes, 2012; Zienkiewicz and Taylor, 2013). The resulting nonlinear system is solved iteratively using Newton-type methods, as described in Section F.3.

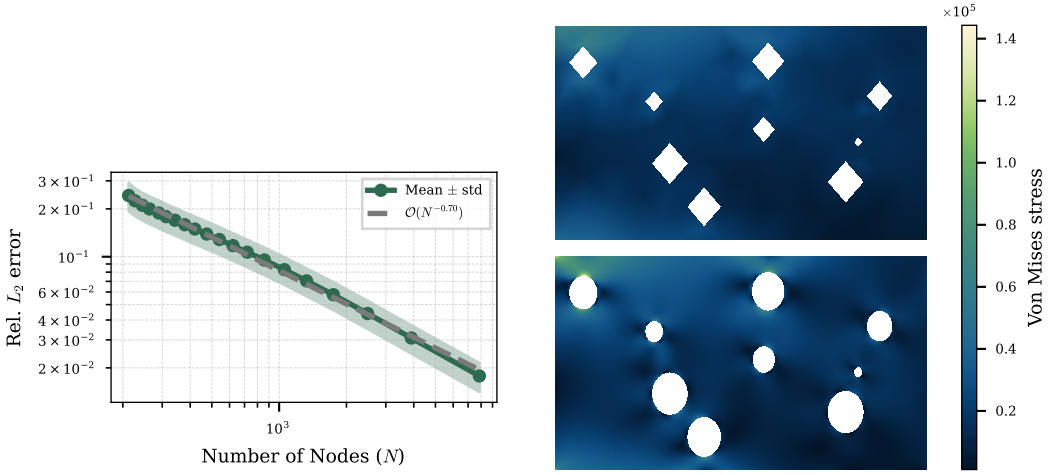
F.2 Domain and boundary conditions

The domain Ω is a 2D rectangular plate with randomly placed circular holes, identical in geometry to the linear elasticity setup shown in Fig. 13. Again, the left boundary is clamped and a randomized traction is applied on the right boundary. Unlike in the linear elasticity case, in this setup compressive loading can cause element inversion due to the geometric nonlinearity. We therefore restrict the traction direction to a $\pm 30^\circ$ cone around the horizontal axis, ensuring mainly tensile loading. Table 4 summarizes the sampling ranges.

As material coefficients, we use $E = 2\,\text{MPa}$ and $\nu = 0.3$, representative of a compressible soft polymer or foam elastomer (e.g., expanded silicone or cellular rubber). With these parameters, the shear modulus is $\mu \approx 0.77\,\text{MPa}$ and the traction-to-shear-modulus ratio ranges from 0.001 to 0.065, placing the problem in a mildly nonlinear regime. The resulting Green–Lagrange strains typically remain below 30%, confirming that the nonlinear constitutive model is exercised beyond the linear regime.

Table 4: Sampling ranges for boundary conditions and geometry parameters in the hyperelasticity experiment.

Parameter	**Symbol**	**Range**
Traction magnitude	T	1–50 kPa
Traction angle	α	$-\pi/6$–$\pi/6$
Number of holes	–	5–20
Hole radii	r_i	0.02–0.15



(a) Empirical von Mises stress error convergence with number of mesh nodes N. We show the mean and ± 1 std across 10,000 simulations.

(b) Visualization of the von Mises stress field at the lowest and highest mesh resolution for an example simulation.

Figure 17: Discretization convergence for the hyperelasticity experiment.

F.3 Simulation and postprocessing

Simulation. We solve the weak form of Eq. (32) using linear (P1) Lagrange elements on triangular meshes, generated with `gmsh` (Geuzaine and Remacle, 2009) via a parametric meshing script with a single continuous parameter controlling mesh fidelity. The meshing setup mimics that of Section E.3. The simulation uses the FEniCSx finite-element framework (Baratta et al., 2025; Alnæs et al., 2014; Scroggs et al., 2022a,b) with PETSc-based nonlinear solvers (SNES). Each linearized step uses a direct LU solver (MUMPS (Amestoy et al., 2001)). For the discretization-error experiments, we use a full Newton–Raphson scheme with relative tolerance 10^{-6}, such that the solver error is negligible relative to the discretization error. For the solver error experiments, we hold the mesh fixed and use a modified Newton (chord) scheme with frozen Jacobian to control solver fidelity (Section F.4 below).

Postprocessing. The postprocessing pipeline mimics the one described in Section E.3.

F.4 Sources of error

Discretization error. As in the linear elasticity case, the spatial discretization uses P1 Lagrange elements on triangular meshes, controlled by a single continuous fidelity parameter that varies the average edge length. The theoretical convergence rates carry over from the linear case: $\mathcal{O}(N^{-1})$ for displacement and $\mathcal{O}(N^{-1/2})$ for stress fields in the L_2 norm (Brenner and Scott, 2008, Chapter 5). The von Mises stress field is postprocessed via L_2 projection onto the continuous P1 space, and the empirical convergence rate lies between these two bounds. Our simulation and postprocessing setup yields an empirical rate of $e(N) = \mathcal{O}(N^{-0.70})$ as shown in Fig. 17a. Fig. 17b shows von Mises stress fields at the lowest and highest discretization fidelities for an example simulation.

Solver error. The nonlinear system arising from the Neo-Hookean model is solved using a modified Newton (chord) method. The Jacobian is computed once at the initial guess $\mathbf{u}_h^{(0)} = \mathbf{0}$ and frozen for all subsequent iterations.[6] This design ensures linear convergence ($\mathcal{O}(\rho^k)$) rather than the quadratic convergence of standard Newton–Raphson. The mesh is held fixed at $N \approx 1{,}600$ mesh nodes across all solver fidelity levels to isolate the solver error axis. Our setup yields an empirical convergence rate of $\rho \approx 0.95$, as shown in Fig. 18. Fig. 19 shows von Mises stress fields at the lowest and highest solver fidelities for an example simulation.

[6] Implemented via PETSc's SNES with `snes_lag_jacobian` and `snes_lag_preconditioner` both set to -2, with a basic line search and a damped chord method.

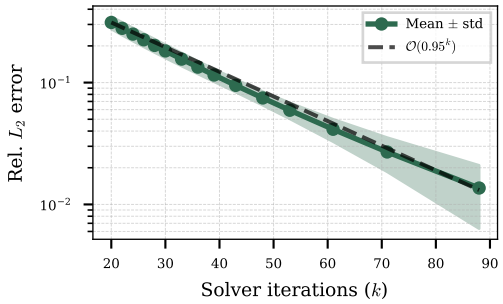


Figure 18: Empirical von Mises stress error convergence with iteration count k for the hyperelasticity solver-error experiment. We show the mean and ± 1 std across 90,000 simulations.

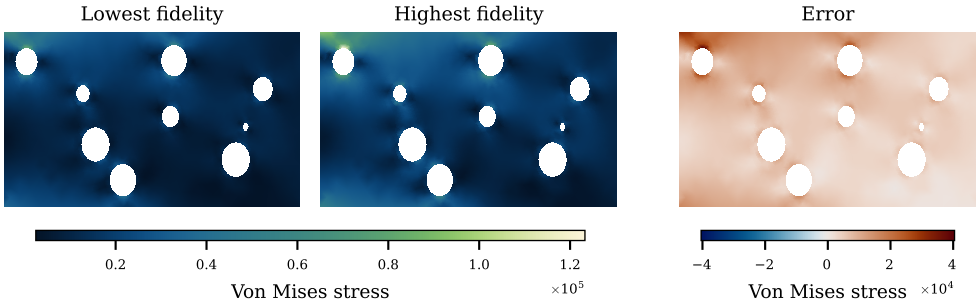


Figure 19: Visualization of the von Mises stress field at the lowest and highest solver fidelities for an example simulation.

G Airfoil experiment

This case setup simulates the airflow around 2D wings, a common setting for aerodynamic shape optimization. We follow the dataset construction pipeline of Bonnet et al. (2022) and extend it to vary the mesh resolution of the simulations. We describe the governing equations in Section G.1, the simulation and postprocessing methodology in Section G.3, the domain and boundary conditions in Section G.2, and the tested error sources in Section G.4.

G.1 Governing equations

We solve the steady-state, incompressible RANS equations; for a detailed derivation, we refer the reader to Bonnet et al. (2022, Appendix F). The incompressibility assumption is justified by the subsonic regime (Mach number < 0.3) of the simulations. Denoting ensemble-averaged quantities by $\bar{\cdot}$, the equations read:

$$\begin{aligned} \nabla \cdot \bar{\mathbf{u}} &= 0, \\ \nabla \cdot (\bar{\mathbf{u}} \otimes \bar{\mathbf{u}}) &= -\nabla \bar{p} + (\nu + \nu_t)\, \nabla^2 \bar{\mathbf{u}}, \end{aligned} \tag{34}$$

where $\bar{\mathbf{u}}$ is the velocity field, $\bar{p}$ the reduced pressure (the effective pressure divided by the constant fluid density ρ), ν the kinematic viscosity, and ν_t the turbulent kinematic viscosity. We use $\rho = 1.184\,\mathrm{kg/m^3}$ and $\nu = 1.56 \times 10^{-5}\,\mathrm{m^2/s}$, representing air at sea level and $298.15\,\mathrm{K}$. The dynamics of ν_t are governed by additional transport equations. For closure, we use the k–ω SST turbulence model of Menter et al. (2003), standard practice in aerodynamics.

The wall shear stress $\boldsymbol{\tau}$ at the airfoil surface is computed from the wall-normal velocity gradient:

$$\boldsymbol{\tau} = \rho(\nu + \nu_t)\, \nabla \bar{\mathbf{u}} \cdot \mathbf{n}\big|_{\partial\Omega_{\text{airfoil}}}, \tag{35}$$

where $\mathbf{n}$ is the outward-pointing wall-normal unit vector. The total aerodynamic force on the airfoil is the integral of pressure and shear contributions over the airfoil surface $\partial\Omega_{\text{airfoil}}$:

$$\mathbf{F} = \oint_{\partial\Omega_{\text{airfoil}}} (-\rho \bar{p}\, \mathbf{n} + \boldsymbol{\tau})\, dS. \tag{36}$$

Decomposing $\mathbf{F}$ into components parallel and perpendicular to the free-stream velocity $\mathbf{U}_\infty$ yields the drag D and lift L, respectively. The dimensionless drag and lift coefficients are then defined as

$$C_d = \frac{D}{q_\infty A}, \qquad C_l = \frac{L}{q_\infty A}, \tag{37}$$

with dynamic pressure $q_\infty = \frac{1}{2}\rho U_\infty^2$ and chord-based reference area $A = 1\,\mathrm{m}^2$.

G.2 Domain and boundary conditions

We model airfoil geometries from NASA's NACA 4- and 5-digit series (Cummings et al., 2015). These airfoil families are parametrized by a small set of integer parameters that determine the maximum camber, its chordwise position, and maximum thickness, allowing systematic exploration of a broad design space. The far-field boundary is placed at 200 chord lengths from the airfoil to minimize boundary-condition effects on the simulation. Table 5 summarizes the variability in geometry and flow conditions in our dataset.

Table 5: Sampling ranges for flow conditions and airfoil geometry parameters in the airfoil experiment. Flow conditions (Reynolds, angle of attack) are sampled uniformly from continuous ranges; airfoil shape parameters are sampled uniformly and rounded to integers, following the NACA 4- and 5-digit families. For 4-digit airfoils, the camber position is set to 0 (uncambered) when the sampled value is below 1.5, eliminating geometries with maximum camber too close to the leading edge, where NACA 4-digit parametrizations become numerically problematic.

Parameter	**Family**	**Range**
Reynolds number	–	$2\text{–}6 \times 10^6$
Angle of attack	–	$-5°\text{–}15°$
Max camber (M)	4-digit	$\{0, 1, \ldots, 7\}$
Camber position (P)	4-digit	$\{0\} \cup \{2, 3, \ldots, 7\}$
Max thickness (XX)	4-digit	$\{5, 6, \ldots, 20\}$
Design lift coeff. (L)	5-digit	$\{0, 1, 2, 3, 4\}$
Camber position (P)	5-digit	$\{3, 4, \ldots, 8\}$
Reflex flag (Q)	5-digit	$\{0, 1\}$
Max thickness (XX)	5-digit	$\{5, 6, \ldots, 20\}$

G.3 Simulation and postprocessing

Simulation. We compute solutions using the steady-state incompressible RANS solver `simpleFoam` from OpenFOAM (Weller et al., 1998).[7] The pressure-velocity coupling uses the SIMPLEC algorithm (Doormaal and Raithby, 1984), with under-relaxation factors applied throughout (0.3 for pressure, 0.7 for momentum and turbulence). Spatial discretization uses Gauss linear schemes for gradients (with cell-based limiters on turbulence quantities for robustness), bounded second-order linearUpwind for convective fluxes, and Gauss linear with limited surface-normal correction for Laplacian terms. Three non-orthogonal correctors are applied per outer iteration to ensure accuracy on the unstructured mesh. Linear systems are solved with PCG/DIC for pressure (absolute tolerance 10^{-6}) and PBiCGStab/DILU for momentum and turbulence quantities (absolute tolerance 10^{-8}). Each simulation is run for 2×10^4 or 4×10^4 outer iterations (depending on the angle of attack), providing a fixed iteration count across all cases and fidelities. All simulations were run on a single AMD EPYC 9655P (96 cores, 192 threads, 4.5 GHz max boost). CPU core hours are reported as wall-clock time multiplied by core count.

Postprocessing. Following the standard finite-volume convention, OpenFOAM stores field values at cell centers. We use `pyvista` (Sullivan and Kaszynski, 2019) to process the raw simulation data stored in VTK format. To use these fields as targets for our neural surrogates, which operate on mesh nodes, we project the cell-centered values to mesh points by averaging the values of all cells incident to each point (`pyvista`'s `vtkCellDataToPointData` filter). This is a standard post-processing step in CFD and introduces a small smoothing effect. The projected fields are stored in HDF5 format.

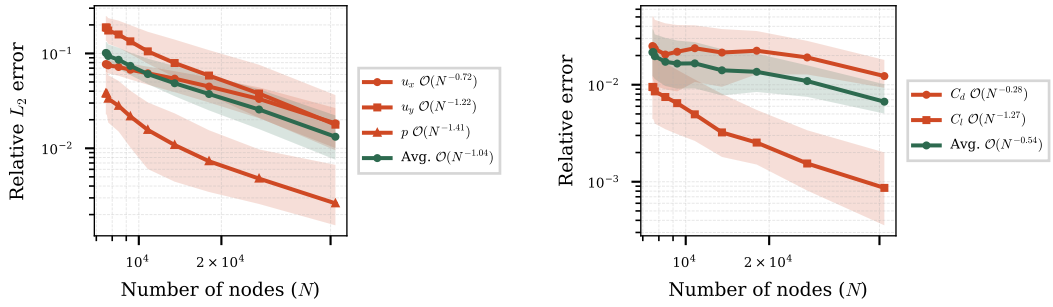
[7] We use OpenFOAM v2506 with Open MPI 4.1.1 for parallel execution.

G.4 Sources of error

Discretization error. We generate meshes ranging from 5,200 to 212,000 mesh nodes, varying the resolution of both the volume and the airfoil-surface mesh.

Fig. 20 displays the resulting empirical discretization error convergence. Fig. 20a shows that each volumetric field exhibits a slightly different error convergence behavior. However, when averaged across fields, the convergence rate is approximately $e(N) = \mathcal{O}(N^{-1.04})$, consistent with the second-order accuracy of the chosen discretization (again, with $N \propto h^{-2}$) (Ferziger et al., 2019, Chapter 4.4).

Fig. 20b shows the convergence of the integrated quantities. While the lift coefficient C_l converges at a rate similar to the field errors, the drag coefficient C_d converges substantially slower.



(a) Volumetric fields error convergence.

(b) Aerodynamic coefficients error convergence.

Figure 20: Discretization error convergence for the airfoil experiment. We show the median (line) and 10th-90th percentiles (shaded band) across 1,627 simulations.

Solver error. Unlike the Poisson and structural mechanics setups, we do not study the solver error setting for the airfoil experiments. The reason is methodological: the SIMPLEC iteration in steady-state RANS does not admit a single, well-defined "solver fidelity" parameter that exhibits the clean exponential convergence as in the other experiments (Sections D to F). Each outer SIMPLEC iteration combines several internal components: a pressure Poisson solve (PCG/DIC), velocity and turbulence transport solves (PBiCGStab/DILU), k–ω SST turbulence-model updates, and three non-orthogonal mesh correctors, each with its own iteration counts and tolerances. The resulting convergence is non-monotonic in the outer iteration count and depends on multiple knobs that interact non-trivially. We therefore run all simulations to a fixed iteration count and study fidelity along the spatial discretization axis only.

H Architecture and training details (grid experiments)

This section specifies the architectures, the fidelity conditioning mechanism, the training objective and schedule, the data pipeline, and the evaluation protocol used for every grid-based Poisson experiment (Section D). The same code path services $D \in \{1, 2, 3\}$ and both fidelity axes (truncation, discretization). The dimension D and the error axis (discretization or truncation) enter only through the array shapes and through the cost oracle $c(\xi)$. An overview of all hyperparameters is given in Table 7. We use a standard FiLM conditioning mechanism (Perez et al., 2018) applied to regular U-Net (Ronneberger et al., 2015) and FNO (Li et al., 2021) architectures.

H.1 Training objective, optimizer, and schedule

Every training run uses the same loss, whitening, optimizer, and schedule; only the data source differs (streaming vs cached, truncation vs discretization axis).

Per-pixel whitening. The dataset of input and output pairs is whitened based on the sample mean and standard deviation. We use a warm-up subsample of $n_w = 8000$ training pairs drawn uniformly without replacement from the budgeted enumeration (or from the full enumeration if $M \leq n_w$).

Loss and optimizer. The training loss is the per-pixel mean-squared error on the whitened output, over mini-batches of size $|\mathcal{B}| = 20$. We optimize with Adam (default $\beta_1 = 0.9$, $\beta_2 = 0.999$) under a warm-up-cosine learning-rate schedule. The learning rate ramps linearly from 0 to $\eta_{\max} = 10^{-4}$ over the first $S_{\text{warm}} = 2000$ steps and then decays as a cosine to 0 over the remaining steps up to the total $S_{\text{total}} = 20000$. We chose this scheduler as we found it to achieve consistent results across configuration cells. For an ablation over the training duration, see Section L.5.

H.2 Data pipeline

For simplicity we never write datasets to disk. Each new seed experiment also generates a new dataset.

Streaming vs cached generation. Two equivalent training paths are supported:

- **Streaming**: each optimizer step regenerates its batch procedurally. No dataset is materialized, and memory is only $\mathcal{O}(|\mathcal{B}|)$. This is the default in 2D and is required in 3D for large M (to avoid writing datasets to disk). Note that although the dataset is procedurally "re-generated" each epoch, we ensure it is pseudo-randomly deterministic. As such, this streaming paradigm does not adversarially create more samples than the current experiment tries to investigate.
- **Cached**: the full M-sample enumeration is pre-materialized once, kept resident on the device (GPU or host RAM if it would otherwise exceed VRAM), and the training loop indexes into the cache. Cheaper per step when the cache fits in memory.

Because both paths call the same underlying generator with the same seed schedule, they produce identical training trajectories up to floating-point non-determinism on the device.

Truncation axis. A sample at fidelity $\xi = K$ is generated by drawing a random windowed Fourier RHS $\mathbf{f}_h$ (see Section D) and applying K V-cycles of the proper multigrid iterator described in Section D.3 to a zero initial guess. All samples live at the target resolution $N_\star$. The fidelity tag ξ is the raw integer V-cycle count.

Discretization axis. A sample at fidelity $\xi = N$ is generated by sampling $\mathbf{f}_h$ at native resolution N, solving $-\Delta_h \mathbf{u}_h = \mathbf{f}_h$ exactly at that resolution (DST direct solver, Section D), and then interpolating both the RHS and the solution from $(N)^D$ to the target grid $(N_\star)^D$ (at which the networks are trained) via multilinear interpolation. Interpolating both sides simultaneously keeps the training pair self-consistent under the target-resolution $(2D+1)$-point Laplacian. Using this setup, all data samples live at $N_\star = 64$ resolution but still contain varying degrees of spatial discretization error. The fidelity tag is the raw native integer N. In Section L.3, we discuss the additional interpolation error this same-resolution approach incurs and how this affects the error improvements of the multi-fidelity approach.

H.3 Fidelity menus, cost model, and swept budget range

This subsection fixes the concrete numerical content of the per-(D, axis) fidelity menus, the cost oracle $c(\xi)$ used by the budgeted sampler, and the normalized-budget range $\tilde{B} = B/c(\xi_{\max})$ that the grid-Poisson sweeps traverse. Menus are chosen following the endpoint-anchoring rule from Section C.1: $e(\xi_{\min})$ lands in $[1 \times 10^{-1}, 3 \times 10^{-1}]$ (crude-but-informative) and $e(\xi_{\max})$ lands near 0.5% (slightly below the surrogate floor).

Cost oracles. Both axes admit a simple closed-form cost per sample.

- **Truncation.** A sample at fidelity K costs $c(K) = K$ V-cycles of the proper multigrid iterator (Section D.3). All truncation samples live at the common target resolution $N_\star = 64$, so the per-cycle constant $N_\star^D$ is the same across every menu entry and factors out of $\tilde{B}$; we report budgets in units of V-cycles.
- **Discretization.** A sample at fidelity N costs $c(N) = N^D$, i.e. the volume of the native solver grid. This reflects the scaling of both the DST direct solve ($\mathcal{O}(N^D \log N)$ up to

log factors) and of the dominant intermediate array computations. Reported budgets are therefore in units of native grid points.

Concrete menus. Table 6 lists the menu Ξ, the target per-endpoint errors used for menu anchoring, and the normalized-budget range actually swept in our experiments. The boundary hyperparameters $\lambda \to \pm\infty$ are implemented in practice as $\lambda = \pm 20$; on the min-max-clamped softmax this places a near-delta at the corresponding endpoint with a small residual mass leak on the interior cells.

Table 6: Fidelity menus used for the grid-based Poisson experiments. Cost $c(\xi)$ is expressed in V-cycles (K) for truncation or native grid points (N^D) for discretization. $\tilde{B} = B/c(\xi_{\max})$ is the budget expressed in units of one maximum-fidelity sample. All menus use $N_\star = 64$ as the target resolution at which the surrogate is trained and evaluated.

Axis	D	**Menu** Ξ	**Cost** $c(\xi)$	$e(\xi_{\min})$	$e(\xi_{\max})$	$\tilde{B}$ **range**
truncation	1	$\{1, 2, 3\}$	k	$\approx 20\%$	$\approx 1\%$	10–10^4
truncation	2	$\{1, 2, 3, 4, 5, 6, 7\}$	k	$\approx 25\%$	$\approx 0.5\%$	10–10^4
truncation	3	$\{2, 3, 4, 5, 6, 7, 8\}$	k	$\approx 13\%$	$\approx 0.4\%$	10–10^4
discretization	1	$\{20, 28, 36, 48\}$	N^1	$\approx 25\%$	$\approx 0.5\%$	10–10^4
discretization	2	$\{20, 32, 48, 64, 96\}$	N^2	$\approx 30\%$	$\approx 0.5\%$	10–10^4
discretization	3	$\{16, 24, 32, 48, 64, 96\}$	N^3	$\approx 20\%$	$\approx 0.5\%$	10–10^4

H.4 Hyperparameter summary

We summarize all hyperparameters in Table 7.

Table 7: Architecture, optimization, and data hyperparameters shared across every grid-based Poisson run. Fidelity menus per (D, axis) are listed in Table 6.

Group	Symbol / flag	Value
Shared backbone / conditioning		
Target resolution	$N_\star$	64 (per axis, all D)
Conditioning dim	d_{cond}	256
Sinusoidal frequencies	–	16, log-spaced in $[0.1, 10]$
Conditioning MLP	–	2 hidden layers, SiLU
FiLM-U-Net		
Levels	L	4
Base channels	C_0	64
Channel multiplier	–	2 (so $(64, 128, 256, 512)$)
Blocks per level	–	2
Conv kernel	–	3 (padding 1)
Norm	–	GroupNorm, $\min(8, C)$ groups
Activation	–	SiLU
Down / up transition	–	strided-2 conv / NN-upsample + conv
Output head	–	1×1 conv
FiLM-FNO		
Backbone blocks	B	4
Hidden channels	H	64
Modes per axis	K_{FNO}	12
Dirichlet pad ratio	r_{pad}	1.5
Activation	–	GELU
Optimization		
Optimizer	–	Adam (β_1=0.9, β_2=0.999)
Peak learning rate	$\eta_{\max}$	10^{-4}
Warm-up steps	S_{warm}	2000
Total steps	S_{total}	20000
Schedule	–	warm-up + cosine decay to 0
Batch size	$\lvert\mathcal{B}\rvert$	20
Loss	–	MSE on whitened output
Whitening warm-up	n_w	8000 samples
RHS distribution		
Modes per axis	M	15
Domain extent	L	1.0
Window	–	outer product of 1D Hann
Normalization	–	$\max \lvert\mathbf{f}_h\rvert = 1$ per sample
Test		
Test samples	–	100
Test seed	–	42
Evaluation fidelity	–	$\xi_{\max}$

I Architecture and training details (mesh experiments)

This section specifies the architectures, fidelity conditioning mechanism, training objective and schedule, data pipeline, and evaluation protocol used for every mesh-based experiment (linear elasticity in Section E, hyperelasticity in Section F, and external aerodynamics in Section G). The solid-mechanics experiments (linear elasticity, hyperelasticity) share a common set of hyperparameters listed in Table 11, while the airfoil experiments use larger model variants and adapted training schedules listed in Table 12.

We use two transformer based surrogate architectures designed for operator learning on irregular meshes: Transolver (Wu et al., 2024) and AB-UPT (Alkin et al., 2025). Both architectures consume the normalized fidelity $\hat{\xi}$ via a sinusoidal encoding. With J frequency pairs, the encoding is constructed as:

$$\gamma(\hat{\xi}) = \big[\sin(\omega_0\hat{\xi}), \ldots, \sin(\omega_{J-1}\hat{\xi}), \cos(\omega_0\hat{\xi}), \ldots, \cos(\omega_{J-1}\hat{\xi})\big], \qquad \omega_j = \tfrac{2\pi}{\ell_{\min}}\, 10^{-4j/J}, \quad (38)$$

i.e. wavelengths geometrically spaced over $[\ell_{\min}, 10^4\ell_{\min})$ with fixed dynamic range $\Lambda = 10^4$. We use $\ell_{\min} = 2\pi/1000$ for our main experiments. For an ablation on $\ell_{\min}$, see Section L.2. This is followed by an MLP, with the resulting conditioning vector then injected into the transformer blocks via adaLN modulation (Peebles and Xie, 2023).

I.1 Data normalization

All input and target fields are normalized with per-channel statistics (mean and standard deviation) computed on the training set, and applied as standard z-score normalization. Spatial coordinates are min-max scaled to $[-1000, 1000]$ along each axis, using the bounding box of the whole training set. Statistics are computed once per training run from the budgeted dataset, stored alongside the model checkpoint, and reused at inference time.

I.2 Training objective, optimizer, and schedule

Loss. The training loss is the relative L_2 error (Eq. (19)), averaged over the F output fields. For single-field problems (von Mises stress in elasticity and hyperelasticity), $F = 1$ recovers the standard relative L_2 error. For airfoil simulations, $F = 5$ covers velocity $\mathbf{u} = (u_x, u_y)$, pressure p, and wall shear stress $\boldsymbol{\tau} = (\tau_x, \tau_y)$. Here, the loss is calculated on the volume and on the surface separately and then averaged with equal weighting as in (Bonnet et al., 2022).

Optimizer and schedule. We use AdamW under a one-cycle learning-rate schedule (Smith, 2018) with cosine annealing. Peak learning rate, total step count, weight decay, and warmup fraction are chosen per experiment family/architecture (see Tables 11 and 12).

I.3 Data pipeline

Mesh datasets are pre-generated and stored on disk in HDF5 format (Sections E.3, F.3 and G.3), with each file containing nodal coordinates, mesh connectivity, field values at each node, the fidelity tag, and per-sample geometric features. Unlike the grid experiments, mesh-based samples are not procedurally regenerated each epoch, since the high cost of the underlying FEM and CFD simulations would make this prohibitive.

Node subsampling. For the structural mechanics experiments, mesh sizes range from 212 to 29,408, and for the airfoil experiments from 5,200 to 212,000 nodes per sample, exceeding the available GPU memory when loaded at full resolution. We therefore subsample to 16,000 for the plate experiments and to 32,000 for the airfoil experiments for Transolver training. For training AB-UPT, this is handled by anchor point sampling (for detailed numbers, see Tables 11 and 12). Evaluation is always performed at full resolution.

I.4 Fidelity menus, cost model, and swept budget range (linear elasticity)

This subsection fixes the concrete numerical content of the linear elasticity fidelity menus, the cost oracle $c(\xi)$ used by the budgeted sampler, and the normalized-budget range $\tilde{B} = B/c(\xi_{\max})$ that the linear elasticity sweeps traverse. Menus are chosen following the endpoint-anchoring rule from Section C.1.

Cost oracles. Both axes admit a simple closed-form cost per sample.

- **Solver.** A sample at fidelity K costs $c(K) = K$ Gauss-Seidel iterations (Section E.4). The mesh is held fixed across all solver fidelity levels, so the per-iteration constant factors out of $\tilde{B}$; we report budgets in units of solver iterations.
- **Discretization.** A sample at fidelity N costs $c(N) = N$, i.e. the number of mesh nodes. This reflects the dominant scaling of the direct LU solve and of the assembled stiffness matrix. Reported budgets are therefore in units of mesh nodes.

Concrete menus. Table 8 lists the menu Ξ (reduced form), the target per-endpoint errors used for menu anchoring, the cost of the most expensive sample, and the normalized-budget range actually swept in our experiments. The boundary hyperparameters $\lambda \to \pm\infty$ are implemented in practice as $\lambda = \pm 20$. Note that for the discretization axis, the menu values are continuous (mesh sizes from a parametric meshing script), and we report the rounded mean nodal counts at each fidelity level.

Table 8: Fidelity menus used for the linear elasticity experiments. Per-endpoint errors $e(\xi_{\min}), e(\xi_{\max})$ are motivated by the target anchors from the menu-selection rule. Cost $c(\xi)$ is expressed in solver iterations (K) for the solver axis or mesh nodes (N) for the discretization axis. $\tilde{B} = B/c(\xi_{\max})$ is the budget expressed in units of one maximum-fidelity sample.

Axis	**Menu** Ξ ($\lvert\Xi\rvert$)	**Cost** $c(\xi)$	$e(\xi_{\min})$	$e(\xi_{\max})$	$\tilde{B}$ **range**
solver	{25,000, …, 262,144} (11)	k	$\approx 15\%$	$\approx 0.5\%$	100–6,400
discretization	{212, …, 29,408} (21)	N	$\approx 25\%$	$\approx 1\%$	100–3,200

I.5 Fidelity menus, cost model, and swept budget range (hyperelasticity)

The hyperelasticity setup uses the same cost oracles and discretization-axis menu as linear elasticity (Section I.4), with a different solver-axis menu reflecting the different convergence behavior of the modified Newton (chord) method.

Cost oracles. The cost oracles are identical to the linear elasticity setup: $c(K) = K$ for the solver axis (where K is the modified Newton iteration count) and $c(N) = N$ for the discretization axis (where N is the number of mesh nodes).

Concrete menus. Table 9 lists the menu Ξ (reduced form), the target per-endpoint errors used for menu anchoring, the cost of the most expensive sample, and the normalized-budget range actually swept in our experiments. The discretization menu is identical to that of linear elasticity (Table 8); we repeat its endpoints here for completeness. The boundary hyperparameters $\lambda \to \pm\infty$ are implemented in practice as $\lambda = \pm 20$.

Table 9: Fidelity menus used for the hyperelasticity experiments. The discretization menu is identical to that of linear elasticity (Table 8). Cost $c(\xi)$ is expressed in modified Newton iterations (K) for the solver axis or mesh nodes (N) for the discretization axis.

Axis	**Menu** Ξ ($\lvert\Xi\rvert$)	**Cost** $c(\xi)$	$e(\xi_{\min})$	$e(\xi_{\max})$	$\tilde{B}$ **range**
solver	{20, …, 190} (16)	k	$\approx 30\%$	$\approx 0.5\%$	100–6,400
discretization	{212, …, 29,408} (21)	N	$\approx 25\%$	$\approx 1\%$	100–3,200

I.6 Fidelity menus, cost model, and swept budget range (airfoil)

The airfoil setup admits only a discretization error fidelity axis (Section G.4 discusses why the SIMPLEC iteration does not afford a clean solver error axis). Unlike the synthetic unit-cost models used for the elasticity setups, here we can measure data generation cost directly in CPU core-hours via the wall-clock simulation time of each sample.

Cost oracle. A sample at fidelity $\xi = N$ (mesh node count) costs $c(N) =$ measured CPU core-hours. Because cost depends on mesh anisotropy, boundary-layer treatment, and convergence behavior in non-trivial ways, we report the empirical mean cost at each fidelity level rather than a closed-form scaling, mimicking real-world simulation pipelines. Reported budgets are therefore in real CPU core-hours.

Concrete menu. Table 10 lists the menu Ξ (reduced form), the target per-endpoint errors used for menu anchoring, and the normalized-budget range swept in our experiments. The menu spans 11 continuous meshing-fidelity levels, ranging from $\sim$7,600 to $\sim$180,000 mesh nodes per sample, with corresponding mean simulation times from $\sim$0.9 to $\sim$8.3 CPU core-hours. The boundary hyperparameters $\lambda \to \pm\infty$ are implemented in practice as $\lambda = \pm 20$.

Table 10: Fidelity menu used for the airfoil experiments. The cost oracle is the empirically measured wall-clock simulation time, in CPU core-hours; we report the mean across samples at each fidelity level. $e(\xi_{\min})$ and $e(\xi_{\max})$ are the volume-field-averaged relative L_2 errors at the menu endpoints.

Axis	**Menu** Ξ ($\lvert\Xi\rvert$)	**Cost** $c(\xi)$	$e(\xi_{\min})$	$e(\xi_{\max})$	$\tilde{B}$ **range**
discretization	{7,600, …, 180,000} (11)	CPU core-hours	$\approx 10\%$	$\approx 0.5\%$	50–800

I.7 Hyperparameter summary (solid mechanics)

Table 11: Architecture, optimization, and data hyperparameters shared across the linear elasticity and hyperelasticity experiments. The same configuration applies to both discretization error and solver error experiments.

Group	Value
Transolver	
Layers	8
Hidden dim	128
Slice dim	128
Attention heads	4
MLP expansion	4
Conditioning dim	16
Dropout	0.1
AB-UPT	
Hidden dim	96
Attention heads	4
Geometry depth	1
Physics blocks	[perceiver, self, cross, self]
Domain anchor points	1024 per domain
Decoder blocks per domain	2
Supernodes	256
Supernode neighbors	32
Cross-attention chunk size	16,000
Fidelity embedding dim	48
Feature embedding dim	48
Optimization	
Optimizer	AdamW (β_1=0.9, β_2=0.999)
Weight decay	5×10^{-4}
Peak learning rate	2×10^{-3}
Schedule	one-cycle cosine annealing
Warm-up fraction	0.2
Total steps	125,000
Train batch size	16
Loss	per-channel relative L_2
Gradient clipping	enabled
Test	
Evaluation fidelity	$\hat{\xi} = 1$ (corresponds to $\xi_{\max}$)
Test samples	1,000 discretization, 4,500 solver

I.8 Hyperparameter summary (airfoil)

Table 12: Architecture, optimization, and data hyperparameters for the airfoil experiments.

Group	Value
Transolver (large)	
Layers	12
Hidden dim	256
Slice dim	256
Attention heads	8
MLP expansion	4
Conditioning dim	16
Dropout	0
AB-UPT (large)	
Hidden dim	192
Attention heads	3
Geometry depth	1
Physics blocks	[perceiver, self, cross, self, cross, self, cross, self]
Domain anchor points	4096 volume, all surface points
Decoder blocks per domain	2
Supernodes	256
Supernode neighbors	32
Cross-attention chunk size	16,384
Fidelity embedding dim	96
Feature embedding dim	96
Dropout	0
Optimization	
Optimizer	AdamW (β_1=0.9, β_2=0.999)
Weight decay	10^{-8}
Peak learning rate	3×10^{-4}
Schedule	one-cycle cosine annealing
Warm-up fraction	0.05
Total steps	100,000
Train batch size	2
Node subsampling	32,000 per sample (training only, Transolver)
Loss	per-channel relative L_2 (volume + surface)
Gradient clipping	enabled
Test	
Evaluation fidelity	$\hat{\xi} = 1$ (corresponds to $\xi_{\max}$)
Test samples	163

J Additional results

This section reports detailed results across all experiments. We show scaling curves for multi-fidelity vs. high-fidelity training together with their respective numerical tables, as well as the needed computational resources required to produce them.

J.1 Poisson experiments

In Table 13, we list the compute resources used for all Poisson experiments in terms of GPU-hours. Note that since the Poisson experiment procedurally generates all training data during each cell run, we do not explicitly report data generation cost. In total, we consumed $\approx$ 1500 A100-equivalent GPU-hours ($\approx$ 62 A100-equivalent GPU-days). This is dominated by the cost of the 3D experiments. We expect to have spent another equivalent (untracked) amount in exploratory experiments. Only negligible storage usage was required due to procedural data generation. Large-scale array studies allowed us to explore the experimental hypergrid virtually in parallel.

Table 13: Compute resources used to produce all experiments related to the Poisson equation in 1D, 2D, and 3D (5,450 distinct training runs across the $(\text{axis}, \text{dim}, \text{arch}, \lambda, B, \text{seed})$ grid). The training recipe is fixed across all cells, so per-cell wallclock depends only on (dim, arch, cluster). A 1g.10gb MIG slice uses one seventh of an A100 resource. Cluster systems used: (i) FAU: Alex system at Friedrich-Alexander Universtität Nürnberg, (ii) LRZ: Leibniz Rechenzentrum AI resources MCML partition, (iii) Leonardo: Leonardo (CINECA) Boost partition

Cluster	Hardware	Dim	Cells	Seeds	Runs	h / run	GPU-h
FAU	A100 80 GB	3D	42	1	42	1–3	42–126
LRZ	A100 80 GB (1g.10gb)	1D	648	6	3,888	0.04–0.1	156–389
LRZ	A100 80 GB (1g.10gb)	2D	$\approx$ 218	6	1,310	0.3–0.6	393–786
Leonardo	A100 64 GB (slower PCIe)	3D	$\approx$ 70	3	210	5.5–6	1,155–1,260
Total			$\approx$ 978		**5,450**		$\approx$ 1,746–2,561
Full-A100							$\approx$ 1,275–1,554

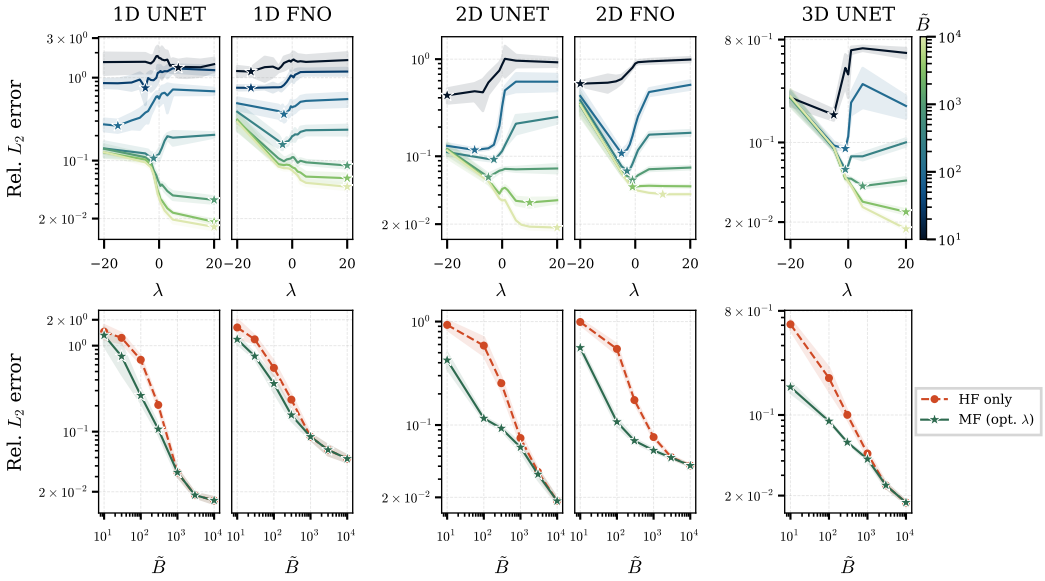


Figure 21: Poisson experiments of multi-fidelity training with discretization error. **Top:** Iso-budget curves over mixture hyper-parameter λ in terms of test error. Minimum loss for each budget is highlighted with $\star$. **Bottom:** Compute-optimal scaling behavior using λ^* at each budget.

Table 14: Summary of the Poisson experiment results (truncation/solver error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline ($\lambda \to \infty$). M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline.

Dim	Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. $\mathbf{L_2}$ error**		
				MF	**HF**	**Δ(%)**	**MF**	**HF**	**Δ(%)**
1D	U-Net	10	-20.00	30	10	+200.0	1.113	1.325	-16.0
		30	-20.00	90	30	+200.0	0.803	1.123	-28.5
		100	-7.00	292	100	+192.0	0.295	0.770	-61.6
		300	-3.50	751	300	+150.3	0.081	0.214	-62.2
		1,000	-1.00	1,781	1,000	+78.1	0.028	0.036	-20.4
		3,000	0.75	3,999	3,000	+33.3	0.018	0.018	-3.4
		10,000	20.00	10,000	10,000	+0.0	0.015	0.015	+0.0
	FNO	10	-20.00	30	10	+200.0	0.967	1.549	-37.6
		30	-3.00	72	30	+140.0	0.766	1.231	-37.8
		100	-2.00	208	100	+108.0	0.318	0.517	-38.6
		300	-3.50	751	300	+150.3	0.131	0.226	-42.0
		1,000	-2.50	2,256	1,000	+125.6	0.060	0.088	-31.5
		3,000	-2.00	6,310	3,000	+110.3	0.037	0.044	-17.3
		10,000	3.50	10,687	10,000	+6.9	0.030	0.031	-2.4
2D	U-Net	10	-20.00	67	10	+570.0	0.711	0.968	-26.5
		100	-10.00	567	101	+461.4	0.163	0.566	-71.2
		300	-5.00	1,195	302	+295.7	0.077	0.263	-70.8
		1,000	-5.00	3,988	1,005	+296.8	0.032	0.075	-57.6
		3,000	-2.00	7,551	3,016	+150.4	0.020	0.031	-35.7
		10,000	20.00	10,053	10,053	+0.0	0.018	0.018	+0.0
	FNO	10	-20.00	67	10	+570.0	0.813	0.991	-17.9
		100	-5.00	402	101	+298.0	0.196	0.549	-64.4
		300	-5.00	1,195	302	+295.7	0.090	0.173	-47.7
		1,000	-5.00	3,988	1,005	+296.8	0.054	0.078	-31.4
		3,000	-0.50	5,711	3,016	+89.4	0.045	0.050	-8.8
		10,000	0.00	17,475	10,053	+73.8	0.042	0.042	-0.1
3D	U-Net	10	-20.00	40	10	+300.0	0.396	0.649	-39.1
		100	-5.00	293	101	+190.1	0.115	0.170	-32.3
		300	-5.00	860	302	+184.8	0.058	0.109	-46.9
		1,000	-5.00	2,898	1,005	+188.4	0.029	0.047	-38.1
		3,000	-1.00	5,495	3,014	+82.3	0.020	0.024	-16.8
		10,000	5.00	11,031	10,047	+9.8	0.017	0.017	-0.3

Table 15: Summary of the Poisson experiment results (discretization error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline ($\lambda \to \infty$). M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline.

Dim	Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. $\mathbf{L_2}$ error**		
				MF	**HF**	**Δ(%)**	**MF**	**HF**	**Δ(%)**
1D	U-Net	10	7.00	10	10	+0.0	1.321	1.454	-9.2
		30	-5.00	65	30	+116.7	0.752	1.232	-39.0
		100	-15.00	239	100	+139.0	0.263	0.685	-61.5
		300	-2.00	539	300	+79.7	0.107	0.204	-47.7
		1,000	20.00	1,000	1,000	+0.0	0.034	0.034	+0.0
		3,000	20.00	3,000	3,000	+0.0	0.018	0.018	+0.0
		10,000	20.00	10,001	10,001	+0.0	0.016	0.016	+0.0
	FNO	10	-15.00	24	10	+140.0	1.184	1.628	-27.2
		30	-15.00	72	30	+140.0	0.755	1.185	-36.3
		100	-3.00	194	100	+94.0	0.362	0.551	-34.3
		300	-3.50	603	300	+101.0	0.156	0.236	-33.8
		1,000	20.00	1,000	1,000	+0.0	0.087	0.087	+0.0
		3,000	20.00	3,000	3,000	+0.0	0.061	0.061	+0.0
		10,000	20.00	10,001	10,001	+0.0	0.048	0.048	+0.0
2D	U-Net	10	-20.00	214	10	+2040.0	0.424	0.928	-54.4
		100	-10.00	1,668	100	+1568.0	0.116	0.586	-80.2
		300	-3.00	2,020	300	+573.3	0.093	0.254	-63.3
		1,000	-5.00	10,100	1,000	+910.0	0.061	0.075	-18.5
		3,000	10.00	3,029	3,000	+1.0	0.033	0.035	-5.2
		10,000	20.00	10,002	10,002	+0.0	0.018	0.018	+0.0
	FNO	10	-20.00	214	10	+2040.0	0.559	0.991	-43.6
		100	-5.00	1,004	100	+904.0	0.107	0.544	-80.3
		300	-3.00	2,020	300	+573.3	0.071	0.174	-59.5
		1,000	-1.00	3,727	1,000	+272.7	0.057	0.076	-25.6
		3,000	-1.00	11,176	3,000	+272.5	0.048	0.049	-1.3
		10,000	10.00	10,100	10,002	+1.0	0.041	0.041	-0.2
3D	U-Net	10	-5.00	375	10	+3650.0	0.175	0.611	-71.4
		100	-1.00	627	100	+527.0	0.088	0.209	-57.7
		300	-1.00	1,899	300	+533.0	0.058	0.100	-42.2
		1,000	5.00	1,151	1,000	+15.1	0.041	0.046	-10.6
		3,000	20.00	3,001	3,001	+0.0	0.025	0.025	+0.0
		10,000	20.00	10,004	10,004	+0.0	0.017	0.017	+0.0

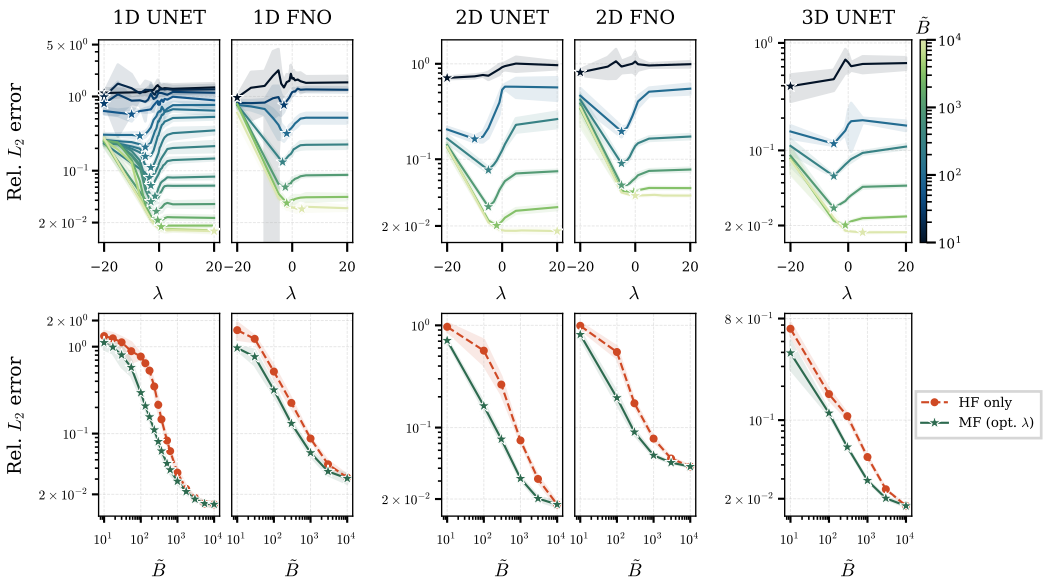


Figure 22: Poisson experiments of multi-fidelity training with truncation error. **Top:** Iso-budget curves over mixture hyper-parameter λ in terms of test error. Minimum loss for each budget is highlighted with $\star$. **Bottom:** Compute-optimal scaling behavior using λ^* at each budget.

J.2 Hyperelasticity experiments

Table 16: Compute resources used to produce all hyperelasticity experiments, reported as actual measured wall-clock GPU-hours per (architecture, cluster) with the mean per-run time. Partially, runs of one sweep were distributed across sites. The *ablations* rows cover the figure ablations on the hyperelasticity solver axis (no-conditioning, $\ell_{\min}$, binary-vs-spectrum). Cluster systems used: (i) local: in-house GPU servers (NVIDIA H100 80 GB, A100 80 GB, A100 40 GB), (ii) Karolina: IT4Innovations (A100 40 GB), (iii) Vega: IZUM (A100 40 GB), (iv) MeluXina: LuxProvide (A100 40 GB).

Axis	Arch	Cluster	Hardware	Runs	h / run	GPU-h
solver	Transolver	local	A100 80 GB	147	∼2.3	∼340
	Transolver	local	A100 40 GB	21	∼2.7	∼58
	AB-UPT	local	H100 80 GB	168	∼3.0	∼506
discretization	Transolver	Vega	A100 40 GB	147	∼11.0	∼1,614
	AB-UPT	Vega	A100 40 GB	115	∼6.6	∼760
	AB-UPT	MeluXina	A100 40 GB	32	∼6.3	∼201
ablations	Transolver	local	A100 80 GB	24	∼4.8	∼115
	Transolver	local	H100 80 GB	57	∼1.8	∼96
Total				**711**		∼3,690

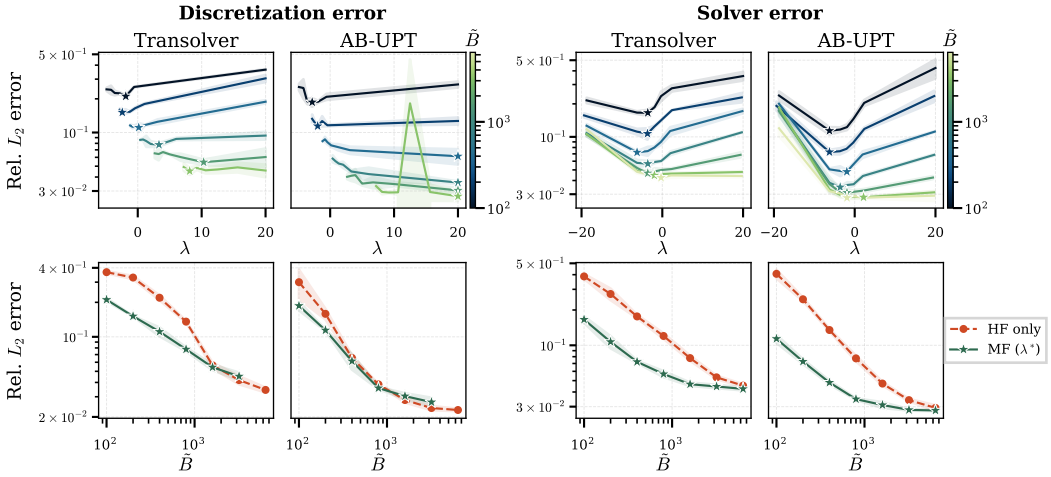


Figure 23: Hyperelasticity experiments. **Top:** Iso-budget curves over mixture hyper-parameter λ in terms of test error. Minimum loss for each budget is highlighted with $\star$. **Bottom:** Compute-optimal scaling behavior using λ^* at each budget.

Table 17: Summary of the hyperelasticity experiment results (solver error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline.

Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. $\mathbf{L_2}$ error**		
			MF	**HF**	**$\Delta(\%)$**	**MF**	**HF**	**$\Delta(\%)$**
AB-UPT	100	-6.25	791	100	+691.0	0.114	0.407	-72.0
	200	-6.25	1,582	200	+691.0	0.073	0.247	-70.5
	400	-1.95	2,104	400	+426.0	0.048	0.135	-64.3
	800	-3.59	5,303	800	+562.9	0.035	0.077	-55.1
	1,600	-1.91	8,425	1,600	+426.6	0.031	0.047	-34.3
	3,200	2.19	8,386	3,200	+162.1	0.028	0.034	-17.4
	6,400	-1.91	33,713	6,400	+426.8	0.028	0.029	-4.8
Transolver	100	-3.59	660	100	+560.0	0.166	0.387	-57.1
	200	-3.67	1,325	200	+562.5	0.107	0.274	-60.9
	400	-6.25	3,188	400	+697.0	0.072	0.176	-59.0
	800	-3.59	5,303	800	+562.9	0.057	0.120	-52.6
	1,600	-3.59	10,577	1,600	+561.1	0.047	0.078	-40.1
	3,200	-1.95	16,916	3,200	+428.6	0.045	0.053	-16.4
	6,400	-0.31	25,287	6,400	+295.1	0.043	0.045	-6.2

Table 18: Summary of the hyperelasticity experiment results (discretization error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline. Optima in the saturated high-fidelity regime are reported as $\lambda^* \geq 15$ (HF-equivalent).

Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. $\mathbf{L_2}$ error**		
			MF	**HF**	**Δ(%)**	**MF**	**HF**	**Δ(%)**
AB-UPT	100	-2.81	4,049	104	+3793.3	0.187	0.300	-37.8
	200	-1.95	5,416	206	+2529.1	0.114	0.159	-28.1
	400	≥ 15	583	411	+41.8	0.061	0.065	-6.2
	800	≥ 15	1,140	810	+40.7	0.036	0.038	-7.1
	1,600	≥ 15	2,279	1,605	+42.0	0.030	0.028	+8.2
	3,200	≥ 15	4,498	3,204	+40.4	0.027	0.024	+12.6
Transolver	100	-1.88	2,757	104	+2551.0	0.211	0.367	-42.5
	200	-2.42	6,710	206	+3157.3	0.151	0.329	-54.0
	400	0.16	4,273	411	+939.7	0.111	0.219	-49.4
	800	3.32	3,411	810	+321.1	0.078	0.136	-42.6
	1,600	10.31	3,195	1,605	+99.1	0.054	0.056	-3.8
	3,200	8.12	7,397	3,204	+130.9	0.045	0.042	+7.9

J.3 Linear elasticity experiments

Table 19: Compute resources used to produce all linear elasticity experiments, reported as actual measured wall-clock GPU-hours per (architecture, cluster) with the mean per-run time. Partially, runs of one sweep were distributed across sites. Cluster systems used: (i) local: in-house GPU servers (NVIDIA H100 80 GB, A100 80 GB, A100 40 GB), (ii) Karolina: IT4Innovations (A100 40 GB), (iii) Vega: IZUM (A100 40 GB), (iv) MeluXina: LuxProvide (A100 40 GB).

Axis	**Arch**	**Cluster**	**Hardware**	**Runs**	**h / run**	**GPU-h**
solver	Transolver	local	A100 80 GB	147	∼2.1	∼312
	Transolver	local	A100 40 GB	21	∼2.8	∼59
	AB-UPT	local	H100 80 GB	168	∼2.9	∼491
discretization	Transolver	local	A100 80 GB	45	∼7.9	∼354
	Transolver	local	A100 40 GB	56	∼14.7	∼822
	Transolver	Vega	A100 40 GB	46	∼13.9	∼639
	AB-UPT	local	H100 80 GB	42	∼4.3	∼179
	AB-UPT	Vega	A100 40 GB	105	∼6.5	∼687
Total				**630**		∼3,544

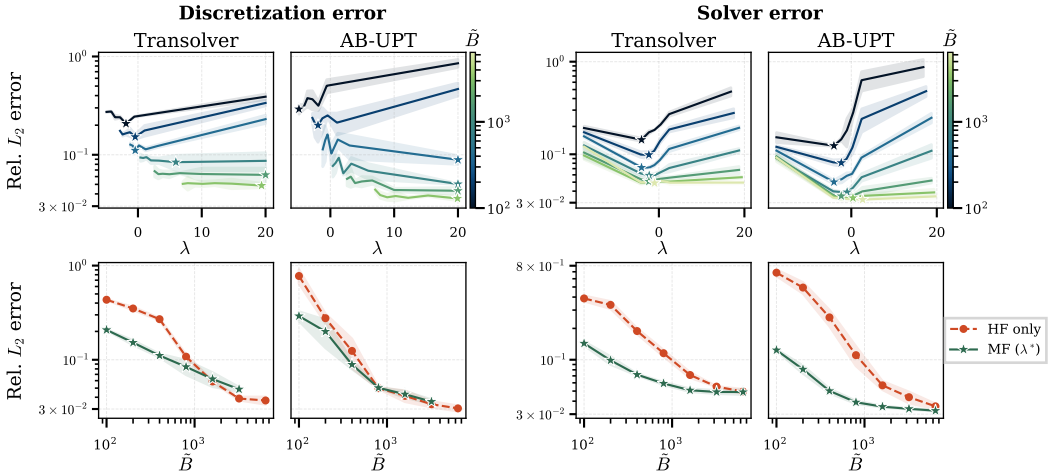


Figure 24: Linear elasticity experiments. **Top:** Iso-budget curves over mixture hyper-parameter λ in terms of test error. Minimum loss for each budget is highlighted with $\star$. **Bottom:** Compute-optimal scaling behavior using λ^* at each budget.

Table 20: Summary of the linear elasticity experiment results (solver error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline.

Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. $\mathbf{L_2}$ error**		
			MF	**HF**	**Δ(%)**	**MF**	**HF**	**Δ(%)**
AB-UPT	100	-4.06	879	100	+779.0	0.124	0.686	-82.0
	200	-2.34	1,455	200	+627.5	0.081	0.494	-83.6
	400	-4.06	3,542	400	+785.5	0.050	0.255	-80.3
	800	-0.98	4,598	800	+474.7	0.039	0.111	-64.7
	1,600	-2.34	11,663	1,600	+628.9	0.035	0.057	-37.6
	3,200	0.39	13,464	3,200	+320.8	0.034	0.044	-22.7
	6,400	2.66	16,953	6,400	+164.9	0.032	0.036	-9.1
Transolver	100	-4.06	879	100	+779.0	0.144	0.388	-63.0
	200	-2.34	1,455	200	+627.5	0.099	0.336	-70.7
	400	-4.06	3,542	400	+785.5	0.072	0.189	-61.9
	800	-2.27	5,812	800	+626.5	0.059	0.116	-48.8
	1,600	-2.34	11,663	1,600	+628.9	0.051	0.071	-28.6
	3,200	-1.04	18,433	3,200	+476.0	0.049	0.055	-11.2
	6,400	-1.02	36,747	6,400	+474.2	0.049	0.049	-1.1

Table 21: Summary of the linear elasticity experiment results (discretization error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline. Optima in the saturated high-fidelity regime are reported as $\lambda^* \geq 15$ (HF-equivalent).

Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. $\mathbf{L_2}$ error**		
			MF	**HF**	**$\Delta(\%)$**	**MF**	**HF**	**$\Delta(\%)$**
AB-UPT	100	-4.92	8,000	104	+7592.3	0.291	0.776	-62.4
	200	-1.93	5,399	206	+2520.9	0.200	0.276	-27.8
	400	≥ 15	565	411	+37.5	0.089	0.124	-28.4
	800	≥ 15	1,114	810	+37.5	0.050	0.050	+0.9
	1,600	≥ 15	2,229	1,605	+38.9	0.043	0.041	+4.1
	3,200	≥ 15	4,401	3,204	+37.4	0.036	0.034	+6.6
Transolver	100	-1.88	2,756	104	+2550.0	0.208	0.434	-52.2
	200	-0.43	2,847	206	+1282.0	0.152	0.350	-56.6
	400	-0.43	5,479	411	+1233.1	0.111	0.270	-58.9
	800	5.94	2,238	810	+176.3	0.084	0.108	-22.0
	1,600	≥ 15	2,229	1,605	+38.9	0.062	0.059	+6.2
	3,200	≥ 15	4,461	3,204	+39.2	0.049	0.039	+25.6

J.4 Airfoil experiments

Table 22: Compute resources used to produce all airfoil experiments, reported as actual measured wall-clock GPU-hours per (architecture, cluster) with the mean per-run time. Partially, runs of one sweep were distributed across sites. Fifteen duplicated AB-UPT HF-only runs on Vega (∼328 GPU-h), superseded by the Karolina re-runs, are excluded. Cluster systems used: (i) local: in-house GPU servers (NVIDIA H100 80 GB, A100 80 GB, A100 40 GB), (ii) Karolina: IT4Innovations (A100 40 GB), (iii) Vega: IZUM (A100 40 GB), (iv) MeluXina: LuxProvide (A100 40 GB).

Arch	**Cluster**	**Hardware**	**Runs**	**h / run**	**GPU-h**
Transolver	local	A100 80 GB	42	∼8.9	∼374
Transolver	Karolina	A100 40 GB	78	∼13.1	∼1,023
AB-UPT	local	H100 80 GB	21	∼12.8	∼269
AB-UPT	Karolina	A100 40 GB	99	∼19.9	∼1,974
Total			**240**		∼3,640

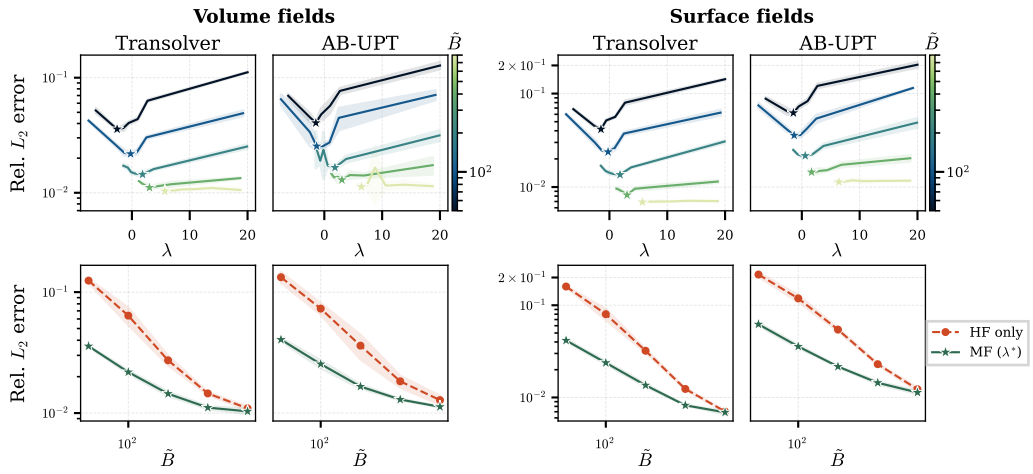


Figure 25: Airfoil experiments. **Top:** Iso-budget curves over mixture hyper-parameter λ in terms of volume and surface test error. Minimum loss for each budget is highlighted with $\star$. **Bottom:** Compute-optimal scaling behavior using λ^* at each budget.

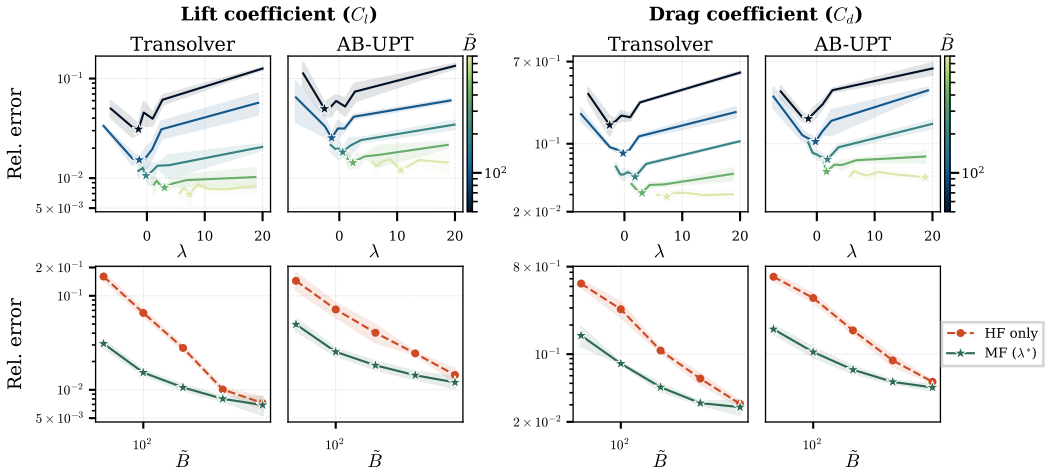


Figure 26: Airfoil experiments. **Top:** Iso-budget curves over mixture hyper-parameter λ in terms of test error of aerodynamic coefficients. Minimum loss for each budget is highlighted with $\star$. **Bottom:** Compute-optimal scaling behavior using λ^* at each budget.

Table 23: Summary of the airfoil experiment results (volume field error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline.

Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. L_2 error**		
			MF	**HF**	**$\Delta(\%)$**	**MF**	**HF**	**$\Delta(\%)$**
AB-UPT	50	-1.46	316	52	+507.7	0.040	0.133	-69.5
	100	-1.33	634	101	+527.7	0.025	0.073	-65.2
	200	1.84	584	200	+192.0	0.017	0.036	-54.1
	400	3.05	872	397	+119.6	0.013	0.018	-29.3
	800	6.41	1,236	798	+54.9	0.011	0.013	-12.0
Transolver	50	-2.56	385	52	+640.4	0.036	0.125	-71.3
	100	-0.23	503	101	+398.0	0.022	0.064	-65.9
	200	1.84	584	200	+192.0	0.014	0.027	-47.1
	400	3.05	872	397	+119.6	0.011	0.015	-23.6
	800	5.70	1,301	798	+63.0	0.010	0.011	-5.6

Table 24: Summary of the airfoil experiment results (surface field error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline.

Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. $\mathbf{L_2}$ error**		
			MF	**HF**	**Δ(%)**	**MF**	**HF**	**Δ(%)**
AB-UPT	50	-1.46	316	52	+507.7	0.062	0.215	-71.1
	100	-1.33	634	101	+527.7	0.036	0.119	-69.9
	200	0.59	764	200	+282.0	0.022	0.054	-60.3
	400	1.68	1,155	397	+190.9	0.014	0.023	-37.2
	800	6.41	1,236	798	+54.9	0.011	0.012	-7.7
Transolver	50	-1.46	316	52	+507.7	0.042	0.160	-74.0
	100	-0.23	503	101	+398.0	0.024	0.080	-70.3
	200	1.84	584	200	+192.0	0.014	0.032	-57.8
	400	3.05	872	397	+119.6	0.008	0.012	-33.6
	800	5.70	1,301	798	+63.0	0.007	0.007	-2.2

Table 25: Summary of the airfoil experiment results (lift coefficient C_l error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline.

Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. error**		
			MF	**HF**	**Δ(%)**	**MF**	**HF**	**Δ(%)**
AB-UPT	50	-2.56	385	52	+640.4	0.050	0.144	-65.6
	100	-1.33	634	101	+527.7	0.025	0.072	-64.6
	200	0.59	764	200	+282.0	0.018	0.040	-54.9
	400	2.36	1,013	397	+155.2	0.014	0.024	-41.4
	800	10.62	1,024	798	+28.3	0.012	0.014	-16.6
Transolver	50	-1.46	316	52	+507.7	0.031	0.160	-80.7
	100	-1.33	634	101	+527.7	0.015	0.066	-76.8
	200	-0.16	940	200	+370.0	0.011	0.028	-62.0
	400	3.05	872	397	+119.6	0.008	0.010	-20.4
	800	7.34	1,159	798	+45.2	0.007	0.007	-4.8

Table 26: Summary of the airfoil experiment results (drag coefficient C_d error). Multi-fidelity (MF) uses the optimal λ^*; High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples; $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline. Optima in the saturated high-fidelity regime are reported as $\lambda^* \geq 15$ (HF-equivalent).

Arch	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. error**		
			MF	**HF**	**Δ(%)**	**MF**	**HF**	**Δ(%)**
AB-UPT	50	-1.46	316	52	+507.7	0.181	0.629	-71.2
	100	-0.23	503	101	+398.0	0.105	0.380	-72.3
	200	1.84	584	200	+192.0	0.069	0.176	-60.8
	400	1.68	1,155	397	+190.9	0.052	0.086	-40.0
	800	≥ 15	880	798	+10.3	0.045	0.052	-12.6
Transolver	50	-2.56	385	52	+640.4	0.156	0.536	-70.9
	100	-0.23	503	101	+398.0	0.080	0.291	-72.5
	200	1.84	584	200	+192.0	0.046	0.109	-58.2
	400	3.05	872	397	+119.6	0.031	0.056	-44.2
	800	7.34	1,159	798	+45.2	0.028	0.031	-7.4

J.5 Bootstrapping approach

The savings ratio $S(\mathcal{E}^\star) = \tilde{B}_{\mathrm{HF}}/\tilde{B}_{\mathrm{MF}}$ in Fig. 9 compares the budgets at which high-fidelity-only training and multi-fidelity training at the optimal mixture reach the specified error threshold $\mathcal{E}^\star$. To compute it, we average the test error of each $(\tilde{B}, \lambda)$ cell over its training seeds, take the minimum over λ at each budget to obtain the respective multi-fidelity curve, and average the high-fidelity runs per budget for the baseline. Both curves are interpolated linearly in log-log space between the two budgets surrounding $\mathcal{E}^\star$ to find the crossing points.

The confidence intervals come from a seed bootstrap. In each of $2{,}000$ draws we resample the seeds of every cell with replacement, recompute the optimal multi-fidelity and high-fidelity curves, and recompute the ratio as described above. Bar heights are the medians of the resulting distribution and whiskers span the 2.5th to 97.5th percentiles. The intervals reflect training randomness only, since the sampled dataset of each cell is fixed.

K Extrapolating lambda values

This section outlines the detailed procedure used for fitting the coefficients from Eq. (7) in Section K.1, calculations on the associated overhead cost in Section K.2, as well as detailed results across all architectures and datasets in Section K.3.

K.1 Fitting the functional form

Following Approach 2 proposed by Hoffmann et al. (2022), we sweep λ for a fixed set of budgets $\tilde{B}$. We then fit a parabola around each iso budget's local minimum and record the optimal lambda value, which is in turn used to fit the variables a and b of Eq. (7).

Our approach aims to use a few low calibration budgets and extrapolate to optimal data mixtures at higher budgets. Below, we provide a detailed recipe for the fitting procedure with safeguards important to achieve robust extrapolation:

1. **Number and spacing of calibration budgets:** We use at least three calibration budgets, which are $\geq 2\times$ apart from each other. Since a two-point linear fit is fully determined, we found it too sensitive to potential outliers in the calibration budget range and recommend three calibration points.
2. **Local parabola fit:** For each calibration budget, we take the λ at minimum error (seed mean) together with its two neighboring grid values and fit a 3-point parabola: its vertex, clipped to the interval spanned by the three points, is the estimate $\lambda^\star(\tilde{B})$. If the triple is not convex, we use the grid minimum itself. If the minimum lies at an edge of the swept grid, no vertex is computed and the budget is not used as a fit point (see Steps 3 and 5). To obtain weights, we repeat this procedure on 300 bootstrap draws (resampling the seeds within each λ cell), discarding draws whose minimum falls on a grid edge, and record the standard deviation of the vertices.
3. **Avoidance of pre-asymptotic regime:** An optimum pinned at the low-fidelity extreme of the sweep (smallest λ), or even drifting toward lower optimal λ with a larger budget, signals a pre-asymptotic regime. An example of this can be seen in Fig. 21 and Fig. 22 for the 1D U-Net, where the three smallest budgets show pre-asymptotic behavior. In this case, we slide the calibration budget window to higher budgets, essentially doubling all three calibration budgets. Repeat this if necessary.
4. **Fitting the functional form:** To obtain a robust fit of the constants a and b from Eq. (7), use a Weighted Least Squares (WLS) procedure using the vertices from Step 2 and $w = 1/\max(\mathrm{std}, 0.25)^2$.
5. **Saturation censoring:** Since λ enters the sampling density exponentially (Eq. (3)), beyond a menu dependent threshold, a "mixture" focuses sampling almost exclusively on high-fidelity samples. We define $\lambda_{\text{sat}} = 15$. This bound is used twice: (i) if the extrapolated prediction is larger than λ_{sat}, allocate the entire generation budget to high-fidelity samples, and (ii) if a calibration budget's vertex from Step 2 lies above λ_{sat}, or its minimum lies at the upper edge

of the grid with $\lambda_{\max} \geq \lambda_{\text{sat}}$, compute at larger budgets is allocated to high-fidelity samples only.

K.2 Overhead cost

The calibration sweep incurs some overhead data generation cost. Below, we quantify this overhead.

Naïvely, the calibration sweep of our setup with 7 mixtures at each of the 3 calibration budgets costs $7 \cdot (B_1 + B_2 + B_3)$. Assuming an always doubling data cost across budgets $(B_1, B_2, B_3, B_4, ...)$, the overhead of the first deployment budget $B_4 = 2B_3$ would be $\sim 613\%$. However, this naïve approach leads to a substantial overestimation, since samples can be reused: assuming all plans draw inputs from the same distribution, a sample generated at fidelity level ξ can be part of any dataset containing that level. The sweep therefore only needs, for each fidelity level, the maximum number of samples over all calibration plans, not their sum. Since the calibration budgets grow geometrically, in most cases this maximum is set by the largest budget, so the smaller budgets add little. By the same argument, the calibration samples can be folded into the deployment dataset, so only samples the deployment cannot absorb count as overhead.

As an example of this, on the linear-elasticity solver problem, this overhead amounts to $\sim 51\%$ of the deployment budget when deploying once at $2B_3$, shrinking to $\sim 9\%$ at our largest tested budget of $16B_3$. For comparison, sweeping λ directly at the deployment budget, with the same sample reuse, costs an additional 170%.

K.3 Detailed results

Table 27 displays the results across all solver error setups, split by dataset and architecture. Since training at arbitrary λ is infeasible, predictions are evaluated at the nearest swept grid cell. Reported regrets therefore include a quantization component that depends on the λ grid spacing and local smoothness of the error landscape. The table shows that our functional fit extrapolates reliably. Regrets stay below 14% in every cell and do not grow with extrapolation depth, tested at budgets up to $16\times$ the calibration ceiling.

Results on the discretization axis (Table 28) exhibit two distinct behaviors. First, for the linear elasticity and hyperelasticity AB-UPT settings the calibration sweeps already place their minima at the high-fidelity boundary of the grid, above λ_{sat}. The saturation censoring (step 5. above) is therefore triggered and HF-only extrapolation is adopted. Second, all Poisson discretization settings undergo an abrupt transition of optimal λ value above the calibration window (Table 15). The empirical minimum drifts slowly within the calibration range and then abruptly jumps to the high-fidelity regions of the sweeps. Our log-linear Ansatz cannot capture this behavior and hence the largest regrets concentrate at the transition budgets (e.g. 48.9% for Poisson-1D FNO, $70.3\%/99.2\%$ for Poisson-2D U-Net).

Notably, the recipe partially recovers beyond the transition. Once the extrapolated prediction crosses λ_{sat}, the saturation clause causes HF-only deployment, which is optimal in the post-transition regime (as indicated by zero regret at the deepest holdout budgets of Poisson 1D).

L Ablations

L.1 Binary vs. spectrum formulation of fidelity

Section 4.1 shows that the spectrum formulation outperforms the binary baseline across all mixtures on the hyperelasticity problem with discretization error, where mesh sizes span two orders of magnitude (Section F). To test the limits of this finding, we repeat the experiment but hold the mesh fixed and instead vary the number of solver iterations. The fidelities are therefore much closer in the sense that they all share the same discretization.

Fig. 27 shows the result at two data generation budgets ($\tilde{B} = 200$ and $\tilde{B} = 1{,}600$). Here, the picture is less clear-cut than in Fig. 4. At both budgets, neither curve dominates the other across the entire mixing range. The spectrum formulation (green line) never peaks as high as the binary formulation (black dashed line), making it more robust to the choice of mixing ratio. At the optima, however, the binary formulation achieves marginally lower errors. We attribute this trade-off to the narrower

Table 27: Per setting recipe results on the **solver error** axis: regret at each holdout budget (ascending).

Dataset	Arch	Calibration $\tilde{B}$	Regret at holdout $\tilde{B}$ (%)			
			1st	2nd	3rd	4th
Poisson 1D	FNO	10, 30, 100	13.6	10.7	8.1	5.9
	U-Net	100, 233, 520	8.7	4.1	2.4	3.3
Poisson 2D	FNO	100, 300, 1000	0.1	0.3	–	–
	U-Net	100, 300, 1000	12.9	3.1	–	–
Poisson 3D	U-Net	100, 300, 1000	0.0	3.1	–	–
Linear elasticity	AB-UPT	100, 200, 400	0.0	0.6	5.2	0.0
	Transolver	100, 200, 400	8.0	9.9	3.6	0.9
Hyperelasticity	AB-UPT	100, 200, 400	4.7	1.7	0.0	0.3
	Transolver	200, 400, 800	0.0	5.1	7.8	–

Table 28: Per-setting recipe results on the **discretization-error** axis: regret at each held-out budget (ladder rungs above the calibration window, in ascending order).

Dataset	Arch	Calibration $\tilde{B}$	Regret at held-out $\tilde{B}$ (%)			
			1st	2nd	3rd	4th
Poisson 1D	FNO	10, 30, 100	48.9	9.2	0.0	0.0
	U-Net	30, 100, 300	27.7	28.9	0.0	–
Poisson 2D	FNO	100, 300, 1000	1.2	9.1	–	–
	U-Net	100, 300, 1000	70.3	99.2	–	–
Poisson 3D	U-Net	10, 100, 300	0.0	22.8	55.6	–
Linear elasticity	AB-UPT	200, 400, 800	0.0	0.0	–	–
	Transolver	100, 200, 400	12.6	9.0	4.0	–
Hyperelasticity	AB-UPT	100, 200, 400	7.7	0.0	0.0	–
	Transolver	100, 200, 400	10.3	20.0	8.6	–
Airfoil	AB-UPT	50, 100, 200	0.0	3.4	–	–
	Transolver	50, 100, 200	0.0	0.0	–	–

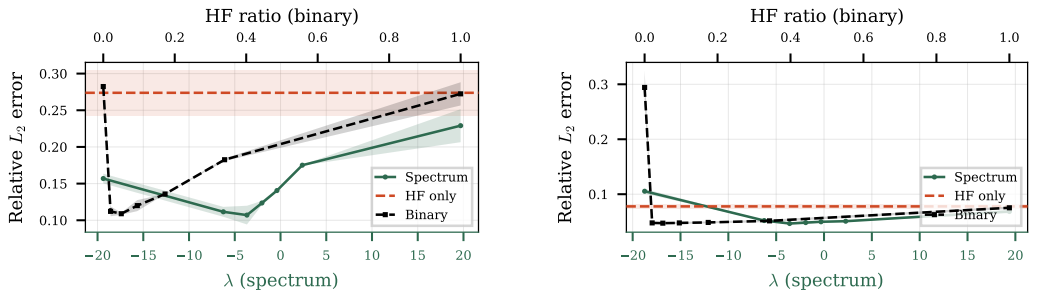
fidelity range in this setting: because all samples share the same discretization, the two endpoints of the binary formulation are much closer.

This raises a natural question: *what is the right family of mixing distributions for multi-fidelity training?* We suspect that no single distribution dominates across all settings. Our framework allows for arbitrary parametrizable distributions over the discrete fidelity menu Ξ, including bimodal distributions that recover the binary formulation as an edge case. However, adding parameters to the distribution enlarges the search space of mixing hyperparameters. We therefore opted for the expressive but simple softmax parametrization and leave a systematic study of alternative mixture distributions for future work (Section 6).

L.2 On interpolation smoothness

We found that the surrogates' error smoothness at interpolation fidelities (unseen during training but in support $\hat{\xi}$) is mainly dependent on two factors, namely the number of samples in adjacent fidelity regions as well as the conditioning resolution of the sinusoidal fidelity encoding. The ablation below demonstrates these findings for a Transolver model trained on the hyperelasticity dataset using different training configurations.

Fig. 28 visualizes the error smoothness at interpolation fidelities for models trained with different budgets $\tilde{B}$ and optimal fidelity mixtures $\lambda^\star$. Models trained at $\tilde{B} = 400$ show prediction errors at



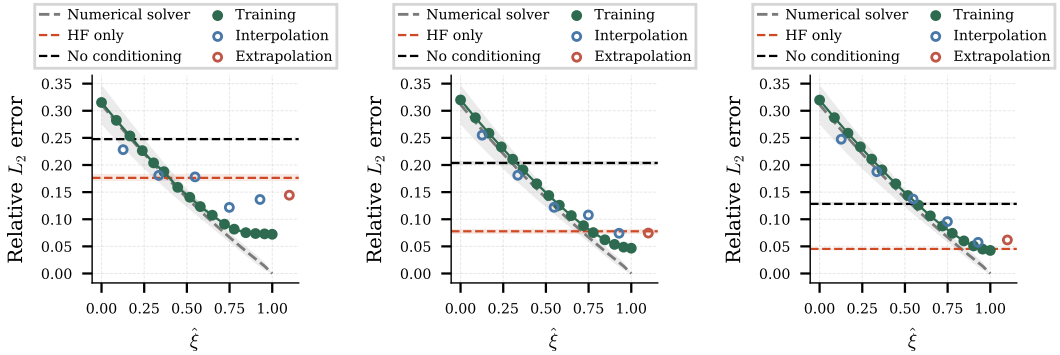
(a) $\tilde{B} = 200$. (b) $\tilde{B} = 1{,}600$.

Figure 27: Spectrum vs. binary fidelity formulation at fixed data generation budgets $\tilde{B}$ for a Transolver model on the hyperelasticity dataset (solver error).

interpolation fidelities that deviate significantly from the respective errors of the numerical solver, especially for higher fidelity levels $\hat{\xi} > 0.5$ (Fig. 28a). This can be attributed to the fact that the overall sample count at this budget is low and the optimal fidelity mix is skewed to lower fidelities. Fig. 28b already displays a smoother interpolation for models trained at $\tilde{B} = 1600$, while still exhibiting some deviations at higher fidelities. In contrast, Fig. 28c clearly shows the benefits of a balanced mix ($\lambda^\star \approx 0$, uniform distribution over fidelities) with extensive training samples at $\tilde{B} = 6400$, leading to a smooth interpolation throughout the entire fidelity support.

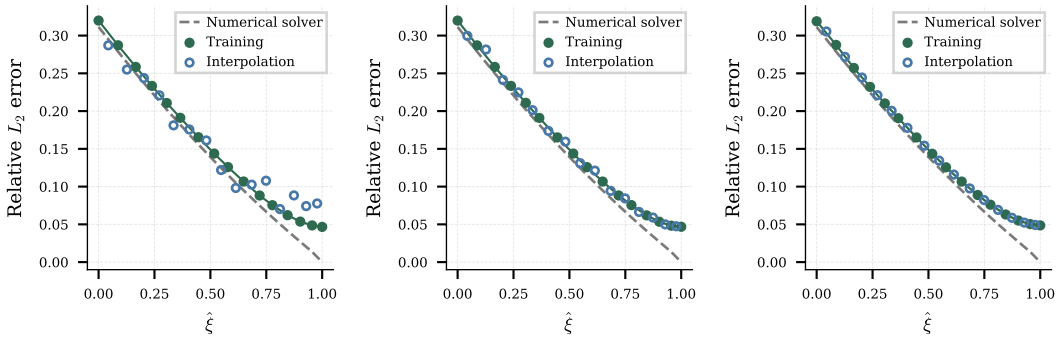
Fig. 29 shows the error smoothness with respect to the sinusoidal encoding resolution of the conditioning signal. We ablate $\ell_{\min} \in \{2\pi, 2\pi/50, 2\pi/1000\}$ (Eq. (38)) at a budget of $\tilde{B} = 1600$ and the corresponding optimal fidelity mixture. The resulting errors at interpolation fidelities become increasingly smooth as the encoding resolution decreases (larger $\ell_{\min}$).

The finest resolution $\ell_{\min} = 2\pi/1000$ achieves the lowest test error at $\hat{\xi} = 1$ with an increasing performance difference at lower budgets. Since the primary goal of our multi-fidelity training is to approximate the highest-fidelity simulator, our main experiments use this setting, trading interpolation smoothness for test performance. If smooth interpolation across fidelities is desired, a resolution calibrated to the spacing of the training fidelities ($\ell_{\min} = 2\pi/50$ here) retains it at a small increase in test error at moderate budgets.



(a) $\tilde{B} = 400$, $\lambda^\star = -6.25$. (b) $\tilde{B} = 1{,}600$, $\lambda^\star = -3.59$. (c) $\tilde{B} = 6{,}400$, $\lambda^\star = -0.31$.

Figure 28: Influence of training budget and fidelity mixture on interpolation smoothness. Showing Transolver models trained on the optimal fidelity mixtures $\lambda^\star$ at different budgets $\tilde{B}$ on the hyperelasticity dataset (solver error axis). (b) repeats Fig. 5 for comparison.



(a) $\ell_{\min} = 2\pi/1000$. (b) $\ell_{\min} = 2\pi/50$. (c) $\ell_{\min} = 2\pi$.

Figure 29: Influence of conditioning resolution on interpolation smoothness. Showing Transolver models trained with $\tilde{B} = 1{,}600$ and $\lambda^\star = -3.59$ across different encoding resolutions $\ell_{\min}$.

L.3 On interpolation errors for the grid experiments

In the Poisson experiments, as described in Section D, the surrogates operate on one fixed resolution per dimension of $N^* = 64$. This design choice was taken in order to benchmark a standard U-Net architecture, which is by default not resolution invariant. Additionally, we saw that models not being fully resolution invariant likely is the issue of reduced MF advantages in mesh experiments. However, to have samples at a consistent resolution carry *discretization error*, we had to interpolate both input $\mathbf{f}_h$ and output $\mathbf{u}_h$ to the operating resolution. For this, we employed a multi-linear interpolation scheme. This incurs a non-negligible error, which we analyze with this ablation.

Multi-linear interpolation is second-order accurate, the same order as the discretization itself (Section D). It therefore does not change the asymptotic convergence of the total error, which retains the expected second-order rate (as visible in Fig. 11). It can, however, change the error constant, and with it the value of low-fidelity samples for training.

To quantify this effect, we reran the 1D U-Net discretization experiment using error-free interpolation with the discrete sine transform. This applies here because the states are bandlimited. In Table 29 we report errors using both approaches and the corresponding error improvements of using the optimal mixture λ^* versus a high-fidelity only run.

In regimes of low budget ($10 < \tilde{B} \leq 1{,}000$), optimal MF-errors are lower on the error-free interpolation branch. This seems aligned with different λ^* being found at many budgets. The HF-only error does not significantly change between the two branches. Correspondingly, the MF advantage is higher under DST interpolation.

Overall, removing the interpolation error increases rather than decreases the MF gain. The multi-linear interpolation used in the main experiments therefore biases the results against MF training, and the MF gains we report for the grid experiments can be seen as conservative.

L.4 Model size ablation

We conduct a small ablation to investigate the effect of model parameter size on multi-fidelity gains. In Table 30, we show variations to the U-Net architecture for the 2D Poisson task over the solver error axis.

The error floor drops monotonically from 4% to 1.4% as capacity grows. While the larger models seem to reach the 10% error threshold training on multiple fidelities with a smaller budget (likely due to 10% being further from their error floor), the achieved multi-fidelity savings remain at about 3×. Note that training duration was kept constant in update steps (wall-clock and training FLOPs grow with model size). We also refer to Section L.5 for an ablation of the number of training steps.

Table 29: Interpolation ablation on the 1D U-Net Poisson experiment (discretization error). We compare the multi-linear prolongation of the main experiments with prolongation via the discrete sine transform (DST). Both branches use the same mixtures $\lambda \in \{-20, -5, -1, 0, 1, 5, 20\}$ and 6 seeds, so the multi-linear values can differ slightly from Table 15, which uses a finer λ grid. Multi-fidelity (MF) uses the optimal λ^*. High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples. $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline. Optima in the saturated high-fidelity regime are reported as $\lambda^* \geq 15$ (HF-equivalent).

Prolongation	$\tilde{B}$	λ^*	$\mathbf{M_{train}}$			**Rel. $\mathbf{L_2}$ error**		
			MF	**HF**	$\mathbf{\Delta(\%)}$	**MF**	**HF**	$\mathbf{\Delta(\%)}$
Multi-linear	10	5.00	11	10	+10.0	1.422	1.454	-2.2
	100	-20.00	240	100	+140.0	0.273	0.685	-60.1
	300	-1.00	488	300	+62.7	0.114	0.204	-44.3
	1,000	≥ 15	1,000	1,000	+0.0	0.034	0.034	+0.0
	3,000	≥ 15	3,000	3,000	+0.0	0.018	0.018	+0.0
	10,000	≥ 15	10,001	10,001	+0.0	0.016	0.016	+0.0
DST	10	≥ 15	10	10	+0.0	1.378	1.378	+0.0
	100	-20.00	240	100	+140.0	0.255	0.754	-66.1
	300	-5.00	644	300	+114.7	0.070	0.200	-64.9
	1,000	-1.00	1,623	1,000	+62.3	0.027	0.036	-23.9
	3,000	≥ 15	3,000	3,000	+0.0	0.018	0.018	+0.0
	10,000	≥ 15	10,001	10,001	+0.0	0.016	0.016	+0.0

Table 30: Varying the parameter size of the surrogate architecture via the base channels of the U-Net (2D Poisson, solver error). NN floor is the HF-only error at the highest budget. $\tilde{B}_{10\%}$ is the budget at which the MF (λ^*) and the HF-only scaling curves reach a relative L_2 error of 10%, and the saving is their ratio, as in Fig. 9 (95% seed-bootstrap confidence interval in brackets). All widths use the mixtures $\lambda \in \{-20, -5, -1, 0, 1, 5, 20\}$ and 6 seeds. With the finer λ grid of the main experiment, the saving of the 64-channel model is 3.7×.

Base ch.	**Params**	**NN floor**	$\tilde{B}_{10\%}$**, MF**(λ^*)	$\tilde{B}_{10\%}$**, HF-only**	**Saving @ 10%**
16	0.69M	4.0%	260	715	2.8× [2.5, 3.0]
32	2.74M	2.5%	236	745	3.2× [2.8, 3.5]
64 (main)	10.92M	1.8%	224	758	3.4× [3.2, 3.6]
128	43.65M	1.4%	207	654	3.2× [2.7, 3.5]

L.5 Training duration ablation

For comparability across λ sweeps and thus different dataset sizes, we hold the number of update steps within one study fixed. To ablate whether multi-fidelity gains only occur due to under- or over-training, we repeat the experiments from the 1D U-Net Poisson study over the solver error axis at budget $\tilde{B} = 300$, varying the update steps (the learning rate scheduler was modified accordingly). In Table 31, we present the achieved test errors of the optimal MF network, the corresponding HF-only network, and the relative change.

Under longer training durations, the MF network consistently improves on the test error, reducing the error from 18.1% at 1,000 steps to 5.9% at 80,000 (a reduction of 3.07×). In contrast, the HF-only surrogate reduces the error by only 1.09×, indicating that the corresponding HF-only dataset is too small.

Given these findings and insights from neural scaling laws (Hoffmann et al., 2022), this would hint that compute-optimal (in the sense of training compute and not our data generation compute) MF models should be smaller but train for more epochs. This, together with the fact that $\lambda^\star$ decreases with increasing training steps, underscores the importance of a full scaling study across all four axes

Table 31: Training duration ablation for the 1D U-Net Poisson experiment (solver error) at $\tilde{B} = 300$. All step counts use the mixtures $\lambda \in \{-20, -5, -1, 0, 1, 5, 20\}$ and 6 seeds, so the 20,000-step row can differ from Table 14, which uses a finer λ grid. Multi-fidelity (MF) uses the optimal λ^*. High-fidelity (HF) is the high-fidelity-only baseline. M_{train} denotes the number of training samples. $\Delta(\%)$ denotes the relative change in number of training samples and error compared to the high-fidelity baseline.

Steps	λ^*	M_{train}			Rel. L_2 error		
		MF	HF	$\Delta(\%)$	MF	HF	$\Delta(\%)$
1,000	-1.00	534	300	+78.0	0.181	0.246	-26.6
2,500	-1.00	534	300	+78.0	0.153	0.239	-36.0
5,000	-1.00	534	300	+78.0	0.139	0.227	-38.7
10,000	-5.00	826	300	+175.3	0.124	0.222	-44.2
20,000 (main)	-5.00	826	300	+175.3	0.094	0.214	-56.2
40,000	-5.00	826	300	+175.3	0.069	0.212	-67.2
80,000	-5.00	826	300	+175.3	0.059	0.227	-74.1

that we identified as future work. Nevertheless, the savings we report in our main study at 20,000 steps are therefore likely underestimates of the potential of the method.

L.6 Ablating fidelity menu endpoints

In Section C.1, we presented the heuristic we use to select ξ_{min} and ξ_{max}. To ablate this, we run the 2D solver-error axis U-Net experiment with a modified high-fidelity endpoint ξ_{max}. Since the numerical fidelity is induced by the number of V-cycles K (Section D.3), we varied $K_{\text{max}} \in \{3, 5, 6, 7, 9\}$ (7 was used in the main experiments) across the full $(\tilde{B}, \lambda)$ sweep. To make menus with different tops comparable, we (i) evaluate every menu against the same menu-independent reference (the DST-exact discrete solution, the $K \to \infty$ limit (Section L.7)), and (ii) normalize all budgets by a single fixed cost anchor ($K = 7$), so equal budget means equal total solver compute. Table 32 displays errors at λ^* and the relative change $\Delta(\%)$ compared to an HF-only run within the menu of the respective $K_{\max}$.

Table 32: Endpoint ablation for the 2D Poisson U-Net experiment (solver error). Each entry reports the mean relative L_2 error (%) at λ^* and, in parentheses, the relative change $\Delta(\%)$ in error compared to HF-only training within the same menu. Budgets $\tilde{B}_c$ are normalized by a fixed cost anchor ($K = 7$), and all menus are evaluated against the same DST-exact reference. All menus use the mixtures $\lambda \in \{-20, -5, -1, 0, 1, 5, 20\}$ and 6 seeds, so the $K_{\max} = 7$ column differs from Table 14, which uses a finer λ grid and the menu top as reference.

$\tilde{B}_c$	$K_{\max} = 3$	$K_{\max} = 5$	$K_{\max} = 6$	$K_{\max} = 7$ (main)	$K_{\max} = 9$
100	14.94 (-62.4)	15.97 (-68.7)	18.00 (-68.3)	20.64 (-62.5)	18.98 (-72.5)
300	7.78 (-22.8)	6.90 (-53.7)	7.36 (-64.2)	7.75 (-70.9)	8.65 (-78.2)
1,000	4.67 (-15.0)	3.21 (-46.3)	3.11 (-53.8)	3.19 (-57.5)	3.43 (-60.1)
3,000	4.14 (-0.4)	2.12 (-14.6)	2.07 (-25.7)	2.12 (-32.5)	2.11 (-45.3)
10,000	4.06 (-0.8)	1.84 (+0.0)	1.76 (-1.6)	1.77 (+0.0)	1.77 (-3.9)

First, our choice is near-optimal and insensitive to over-specification: The compute-optimal error saturates for $K_{\max} \geq 6$ at $\approx 1.8\%$, matching the network's capacity floor (Section L.4), and moving the high-fidelity endpoint from $K = 7$ to $K = 9$ changes it by $<1\%$ relative. Our heuristic ($K = 7$, placed just below the network floor) therefore lies on the compute-optimal envelope, and drawing the endpoint finer buys essentially nothing.

Second, a finer endpoint matters only in the data-limited regime. At the smallest budget $\tilde{B}_c = 100$, a cruder top is preferable, since more samples per unit compute outweigh their lower individual accuracy. The compute-optimal top then rises with budget ($K = 3 \to K = 5 \to K = 6$ as $\tilde{B}_c$ grows), and from $\tilde{B}_c = 3{,}000$ on $K = 6$, 7 and 9 lie within $\approx 3\%$ of each other. An over-specified

endpoint thus costs accuracy only at low budget, and moderately so (e.g., 8.65% vs. 6.90% at $\tilde{B}_c = 300$ for $K = 9$), whereas an under-specified one caps the achievable error at its own accuracy (≈4.1% for $K = 3$ vs. ≈1.8% at $\tilde{B}_c = 10{,}000$).

Third, our choice is conservative for the central claim, and mixing helps at every endpoint wherever data is the binding constraint. For $\tilde{B}_c = 300$ to 3,000, the error reduction of multi-fidelity training grows monotonically with endpoint fineness (e.g., -78.2% at $K = 9$ vs. -70.9% at $K = 7$ at $\tilde{B}_c = 300$). Notably, the reduction grows with fineness even where the achievable error is nearly identical, because a high-fidelity sample costs K: a finer endpoint makes the single-fidelity baseline more expensive and hence more data-starved at fixed budget (at $\tilde{B}_c = 300$ the pure-HF dataset holds ≈3× fewer samples at $K = 9$ than at $K = 3$, giving 39.7% vs. 10.1% single-fidelity error), which the mixture recovers. Conversely, even at the cheapest endpoint ($K = 3$) multi-fidelity training still reduces the error wherever data is the binding constraint (-62.4% at $\tilde{B}_c = 100$ and -22.8% at $\tilde{B}_c = 300$). Multi-fidelity mixing thus pays off regardless of where the high-fidelity line is drawn, and a finer line only raises the stakes of going single-fidelity.

In summary, we find that our heuristic endpoint choice lies within the compute-optimal regime and is robust to over-specification. A further refinement, which we leave to future work, is to move beyond the monotone softmax mixture (which for any λ concentrates toward an endpoint of the menu) to a family whose mode can lie in the interior (e.g., a truncated Gaussian over log-fidelity with a directly tunable mode). Such a family could emphasize intermediate fidelities directly, decoupling the compute-optimal allocation from the placement of the endpoints.

L.7 On the test reference fidelity

In Section C.5, we argued that we use $\mathcal{P}_h(\mathbf{f}_h, \xi_{\max})$ and not $\lim_{\xi\to\infty} \mathcal{P}_h(\mathbf{f}_h, \xi)$ as the reference data for testing the surrogates. To put it differently, we measure the networks against the *best in-support label*. This was chosen pragmatically to maintain consistency, since some of our mesh-based experiments do not allow us to converge the discretization error all the way to machine precision. Nevertheless, it may raise the question of how this could affect multi-fidelity gains.

Table 33 showcases the result of the 2D U-Net Poisson experiment on the solver error axis, where we additionally evaluate all trained models against the fully converged reference. Across all budgets $\tilde{B}$, the errors against the converged reference are higher, but the relative difference remains at most ≈0.5%. As expected, this small difference between the two settings aligns with our desiderata of choosing the highest point of the fidelity menu Ξ to have $e(\xi_{\max}) \approx 0.5\%$, which was outlined in Section C.1 and lies below the network's error floor (Section L.4). Since this residual error equally applies to the HF-only models, the relative change $\Delta(\%)$ shifts by at most 0.2 percentage points.

Table 33: Results for evaluating the surrogates not only against their best in-support label at $\xi_{\max}$, but also against the converged reference $\xi \to \infty$ for the 2D U-Net Poisson experiment on the solver error axis ($K_{\max} = 7$ runs of Table 32). Errors are mean relative L_2 errors at λ^* over 6 seeds; rel. diff. is computed from unrounded values. $\Delta(\%)$ denotes the relative change in error at λ^* compared to the high-fidelity baseline within the menu.

	Error at λ^* (%)			**Δ(%)**	
$\tilde{B}$	**menu top**	**conv. ref**	**rel. diff.**	**menu top**	**conv. ref**
100	20.56	20.64	+0.42%	-62.7	-62.5
300	7.73	7.75	+0.25%	-71.0	-70.9
1,000	3.18	3.19	+0.25%	-57.6	-57.5
3,000	2.12	2.12	+0.08%	-32.4	-32.5
10,000	1.77	1.77	+0.51%	+0.0	+0.0

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We formalize the multi-fidelity training in Section 3.2 (and further extend upon in Section C). Our literature review in Section 2 confirms that this is the first work truly training on multi-fidelity (not just binary) mixtures. Ablations on the fact that surrogates learn the fidelity and can be steered with their conditioning are found in Section 4. Our full set of experiments is described in Section 5 (with consistent multi-fidelity compute savings) and complete results across all axes are presented in Section J.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in Section 6.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: We present an extensive empirical investigation, but no theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: We provide more details in our reproducibility statement in Section A. Our full set of experiments exclusively relies on open-source software. The code is public and links are on the first page.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: Our code is public including instructions how to reproduce all results (see first page for links).

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Full experimental descriptions are given in Sections H and I.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We rerun all our experiments with at least three different random seeds (six for the 1D and 2D Poisson experiments) and report aggregated statistics.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We report the consumed GPU resources for each experiment in Tables 13, 16, 19 and 22.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that our work conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: Our work is foundational and does not have direct societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Our work does not use data or models that have a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: When building upon the work of others (simulation software, numerical methods, neural architectures, etc.), we cite them appropriately.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The released code and datasets are documented and licensed accordingly.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: There are no crowdsourcing experiments or research with human subjects in the paper.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: There are no crowdsourcing experiments or research with human subjects in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are not used as a component of the core methodology of this paper.

    Guidelines:

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