Geometry Meets Physics: Data-Efficient Pre-Training for Unstructured Neural PDE Solvers

Luis Medrano-Navarro[1] **Giacomo Baldan**[1] **Qiang Liu**[1] **Benjamin Holzschuh**[1]
Jan Hagnberger[2] **Mathias Niepert**[2] **Nils Thuerey**[1]

[1]Technical University of Munich [2]University of Stuttgart
{luis.medrano, nils.thuerey}@tum.de

Abstract

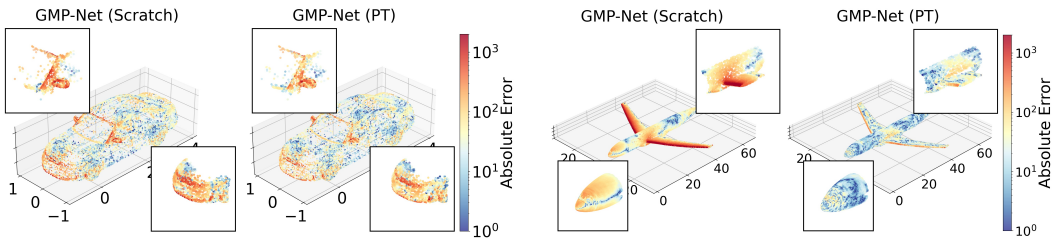
Neural surrogate models for Partial Differential Equations (PDEs) on unstructured 3D geometries are often limited by poor generalization and the high cost of generating large-scale training datasets. Consequently, pre-training on massive datasets of related PDE dynamics has emerged as a critical alternative to enhance the robustness and scalability of these models. However, this strategy is neither compute- nor data-efficient, as it relies on massive pre-computed data that is very costly to generate. In this work, we introduce a disk-data-free pre-training framework tailored to both steady-state and transient regimes. For steady-state problems, we propose a geometry-driven strategy that leverages intrinsic shape descriptors to learn representations of complex 3D domains. For transient problems, we introduce a physics-driven approach based on online generation of synthetic PDE data, enabling scalable pre-training without reliance on expensive datasets. Across multiple experiments, our approach achieves faster convergence, greater data efficiency, and higher accuracy during fine-tuning, particularly under realistic low-data regimes. This methodology provides a practical pathway toward data-efficient neural emulators for large-scale simulations. Code and models are available at `https://github.com/tum-pbs/GMP`.

1 Introduction

Engineering workflows have long depended on high-fidelity numerical solvers to resolve intricate physical systems [1–4]. Nevertheless, their immense computational overhead severely limits their feasibility in time-sensitive settings, iterative design exploration, and closed-loop control. Data-driven surrogate models provide a compelling pathway toward high-throughput inference [5–8]. Yet, practical adoption of these emulators remains critically constrained by an underlying bottleneck: their performance hinges on massive, pre-computed simulation datasets whose generation is itself computationally prohibitive. Motivated by advances in Large Language Models [9, 10, LLMs], foundation models for PDEs have adapted pre-training and fine-tuning paradigms to physical systems [11–15]. Yet, these approaches still rely heavily on extensive offline datasets and predominantly target structured grids without geometric variation. Although recent work on unstructured meshes explores geometric pre-training [16, 17], capturing the coupled dynamics of geometry and physics remains challenging without costly labeled simulations. This motivates our central question: *to what extent can unstructured pre-training be realized efficiently without any precomputed simulation data?*

We categorize unstructured physical problems into two regimes with distinct dynamics: *steady-state* and *transient*. Steady-state problems correspond to equilibrium solutions where physical fields, e.g., surface pressure and shear stress in aerodynamics, are fully dictated by the geometry, boundary conditions and operating conditions. Consequently, pre-training purely on geometric descriptors can yield representations directly transferable to downstream flow prediction. In contrast, transient

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



(a) DrivAerML surface pressure errors

(b) SHIFT-Wing surface pressure errors

Figure 1: Point-wise absolute error comparison between trained from scratch and pre-trained models (PT). Our GMP pre-training shows increased accuracy in critical regions.

Table 1: Comparison of pre-training paradigms across geometric and physical constraints.

Property	**Traditional FM**	**Zhang et al. [17]**	**GeoPT**	**GMP (Ours)**
Physics-label-free	✗	✓	✓	✓
Low pre-training cost	✗	✓	✗	✓
Unique pre-training	✓	✗	✓	✓
No high-detail CAD geometries	✗	✗	✓	✓
Zero disk storage	✗	✗	✗	✓

simulations evolve states autoregressively, with dynamics conditioned jointly on prior physical states in addition to geometric information. This distinction motivates specialized pre-training strategies for each regime to capture structural and physical inductive biases.

To circumvent the prohibitive, often intractable burden of generating massive pre-computed datasets, we develop an entirely storage-free pre-training paradigm that constructs supervisory data on-the-fly. For steady-state regimes, we generate random assemblies of geometric primitives online and train the model to reconstruct essential differential descriptors, including surface normals, curvature, and Vector Distance Fields (VDF). For transient autoregressive settings, we avoid offline trajectory storage by generating solutions to simple, canonical PDEs online during training. Together, these online strategies instill robust geometric and physical inductive biases at reduced computational overhead, bypassing the prohibitive cost of large-scale offline simulation repositories.

Our main contributions are: (**I**) We present *Geometry Meets Physics* (*GMP*), a modular pre-training framework with two distinct variants for steady-state and transient physics simulations on unstructured meshes that bypasses the need to store precomputed PDE solutions and expensive CAD-level geometries. (**II**) We demonstrate its effectiveness to enable faster convergence, data savings and superior accuracy during fine-tuning, particularly in realistic low-data regimes. Additionally, we show that it is inherently architecture-agnostic. (**III**) We propose *GMP-Net*, a modified SOTA architecture specifically designed to efficiently pre-train for both transient and steady-state problems and to boost performance. In summary, Table 1 shows the key structural advantages of GMP. Relative to prior work, GMP requires zero offline data storage, dramatically accelerates pre-training time and supports multiple downstream tasks from a single pre-training stage.

2 Related work

Neural PDE solvers Early efforts in neural physics surrogate modeling mainly employed convolutional neural networks (CNNs) over regular Cartesian grids [18–20]. Concurrently, architectures such as DeepONet [21] and Fourier Neural Operators (FNO) [22] emerged as alternative architectures with varying advantages and disadvantages. Motivated by the success of attention-based architectures in LLMs [23] and computer vision (CV, [24–26]), recent research has adapted Transformers to PDEs, yielding architectures such as FactFormer [27], scOT [13], and PDE-Transformer [28, 29]. Some of these even include physics biases in the form of hard constraints, such as rotational equivariance [30, 31]. Nonetheless, conventional CNNs and vision Transformers remain inherently constrained to Euclidean structured grids. To support irregular computational domains, Graph Neural Networks (GNNs) rapidly gained traction [32–35]. Early GNNs relied primarily on local message passing

and frequently encountered scalability bottlenecks on large-scale 3D meshes comprising millions of vertices. Subsequent frameworks have addressed these constraints by combining graph-based topological structures with attention mechanisms [36], extending Transformers to unstructured point clouds and meshes, including OctFormer [37], OFormer [38], RIGNO [39], UPT [40], and GIST [41]. A persistent challenge is the quadratic complexity $\mathcal{O}(N^2)$ of standard attention, which becomes prohibitive for high-resolution 3D simulations. Recent models deploy diverse efficiency strategies: restricting attention to localized geometric neighborhoods (e.g., GAOT [42], Erwin [43], PTV3 [44]) or the use of subsampled anchor points e.g., AB-UPT [45] and SMART [46]. In parallel, Transolver [47] and its adaptations, such as GeoTransolver [48], address this scaling bottleneck by utilizing physics-informed sliced attention.

Pre-training physics surrogate models While physics foundation models still trail progress in LLMs and CV, several recent frameworks have narrowed this gap [49]. DPOT [15], Poseidon [13], PhysiX [12], and Walrus [14] represent recent examples, scaling up to one billion parameters across domains ranging from astrophysics to plasma physics. Others, such as Yang et al. [50], explore particular applications in aerodynamic design. However, data-driven physics foundation models still require massive precomputed datasets. While frameworks like Tadpole [11] utilize online data generation via GPU-based spectral solvers, they remain restricted to regular grids. Furthermore, the approaches above are typically designed for autoregressive temporal rollouts rather than supporting direct *geometry-to-physics* mappings, such as predicting steady-state aerodynamics around complex geometries. For this latter setting, recent research emphasizes leveraging intrinsic geometric features through specialized pre-training. For instance, the masked autoencoder (MAE) paradigm [51] has been adapted to physics [52], while GeoPT [16] introduced geometric pre-training via lifting. Similarly, others set up pre-train architectures to predict the occupancy of complex shapes without corresponding physics labels [17]. To date, no prior work has presented a modular framework for geometric and physics pre-training, nor demonstrated the effectiveness of storage-free online pre-training for unstructured simulations.

3 Method

This section outlines our pre-training method. As a starting point we consider a unified formulation for steady-state and transient problems. Let a spatial domain $\Omega(\mathcal{S}) \subset \mathbb{R}^n$ be defined by a geometry surface $\mathcal{S}$, and let $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^m$ represent the physical target fields. The governing system obeys

$$\frac{\partial \mathbf{u}}{\partial t} + \mathcal{N}(\mathbf{x}, \mathbf{u}; \mathcal{S}, p, t) = 0, \quad \mathbf{x} \in \Omega(\mathcal{S}), \quad t > 0, \tag{1}$$

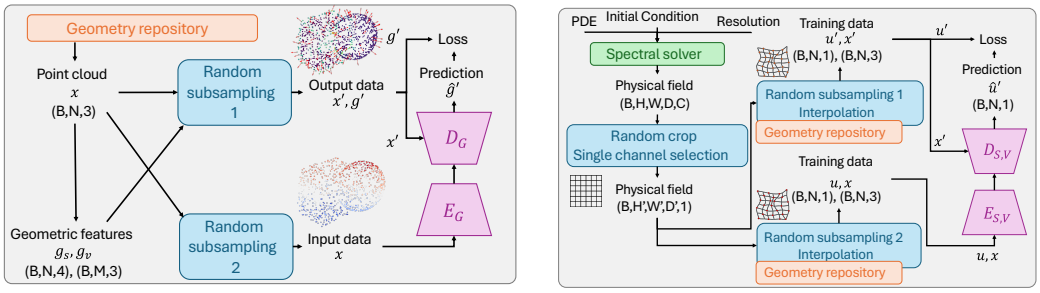
where $\mathcal{N}$ is a differential operator parametrized by the geometry $\mathcal{S}$ and global parameters p. If we discretize $\Omega(\mathcal{S})$ via a point cloud $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^N$, the discretized fields can now be written as $\mathbf{u}_t \in \mathbb{R}^{N \times m}$. We want a surrogate $f_\theta : (\mathbf{x}_t, \mathbf{u}_t, \mathcal{S}, p) \mapsto \mathbf{u}_{t+1}$ for the temporal evolution optimized via error minimization of trajectory data

$$\min_\theta \sum_{j,t} \left| f_\theta(\mathbf{x}_t^{(j)}, \mathbf{u}_t^{(j)}, \mathcal{S}^{(j)}, p^{(j)}) - \mathbf{u}_{t+1}^{(j)} \right|^2, \tag{2}$$

over several available training samples. This formulation seamlessly encompasses steady-state regimes, where solutions $\bar{\mathbf{u}}(\mathcal{S}, p)$ correspond to fixed points satisfying $\mathcal{N}(\mathbf{x}, \bar{\mathbf{u}}; \mathcal{S}, p) = \mathbf{0}$. By decoupling from past temporal states, the model reduces to the direct mapping $f_\theta : (\mathbf{x}, \mathbf{0}, \mathcal{S}, p) \mapsto \bar{\mathbf{u}}$, trained via

$$\min_\theta \sum_{j} \left| f_\theta(\mathbf{x}^{(j)}, \mathbf{0}, \mathcal{S}^{(j)}, p^{(j)}) - \bar{\mathbf{u}}^{(j)} \right|^2. \tag{3}$$

For neural network architectures below, we distinguish geometry representations $\mathcal{S}$, surface physical fields $\mathbf{u}_S$ and volumetric physical fields $\mathbf{u}_V$.



(a) Geometric pre-training

(b) Physics pre-training

Figure 2: Sketches for our two pre-training strategies. For steady predictions, we use geometric pre-training (left). The model learns to predict geometric features of randomized geometries, like the normals and curvature on the surface, and the Vector Distance Field (VDF) in the surrounding volume. For transient problems, we use physics pre-training (right). The model learns to autoencode online generated solutions to canonical, synthetic PDEs at randomized locations.

3.1 Geometric pre-training

For geometric pre-training we fully rely on online generated geometries and construct randomized assemblies of cubes, spheres and tetrahedra as core primitives. Mathematically, we define shapes $\mathcal{S}(\alpha) \subset \mathbb{R}^3, \quad \alpha \sim \mu(\alpha)$, where $\alpha \in \mathbb{R}^k$ are shape parameters and μ defines their sampling distribution. This construction yields a controlled distribution of geometries with varied, well-defined surface properties. Each surface $\mathcal{S}(\alpha)$ is discretized as a point cloud $\{\mathbf{x}_S^{(i)}\} \subset \mathcal{S}(\alpha)$, $i = 1 \dots N$ and for each we compute a set of geometric descriptor fields $g = \big(n, \kappa, v\big)$. Here, $n : \mathcal{S} \to \mathbb{S}^2$ is the unit normal field, $\kappa : \mathcal{S} \to \mathbb{R}$ denotes the curvature, and $v : \Omega \to \mathbb{R}^3$ is the Vector Distance Field (VDF) defined on a volumetric neighborhood $\Omega \supset \mathcal{S}$. In particular, as κ we use the surface variation (or PCA curvature index [53]). To probe volumetric structure, we sample query locations $\{\mathbf{x}_V^{(j)}\} \subset \Omega$, $j = 1 \dots M$ from a parametrized volumetric deformation within a bounding domain enclosing $\mathcal{S}(\alpha)$. This induces a joint learning problem over surface and volume representations.

For architectures with continuous queryable outputs, we formulate geometric pre-training as a masked operator regression problem: given a subset of locations $\mathbf{x} = (\mathbf{x}_S, \mathbf{x}_V)$, the model predicts descriptor values $g(\mathbf{x}')$ at disjoint query points $\mathbf{x}'$ with $\mathbf{x} \cap \mathbf{x}' = \emptyset$, via $\hat{g}'_\theta = D_\theta(E_\theta(\mathbf{x}), \mathbf{x}')$, where E and D represent the model's encoder and decoder. For architectures with fixed output positions, we drop $\mathbf{x}'$, and pre-training is formulated in the same fixed positions $\hat{g}'_\theta = NN_\theta(\mathbf{x})$. This objective enforces the learning of a continuous latent representation with the model's encoder $z_\theta = E_\theta(\mathbf{x})$ approximating the underlying geometric fields across both $\mathcal{S}$ and Ω. The corresponding learning objective is

$$\mathcal{L} = \omega_\kappa \mathcal{L}_\kappa + \omega_n \mathcal{L}_\mathrm{n} + \omega_V \mathcal{L}_\mathrm{V}, \tag{4}$$

where each component of the loss is $\mathcal{L}_i = ||\hat{g}'_{i,\theta} - g'_i||$, and ω_i denote weights for each loss term. To avoid conflicting directions between the three gradients, we use ConFIG [54] as our optimization algorithm (more details in Appendix A).

The sketch of geometric pre-training is provided in Figure 2a. For the volumetric discretization of $\Omega \supset \mathcal{S}$, we use smooth, and intersection-free Gaussian deformed grids (Figure 3). To do so, we build Gaussian displacements $\mathbf{d}(\mathbf{x})$ (Appendix G) and then $\Omega(\mathbf{x}, \alpha) = \mathbf{x} + \alpha \mathbf{d}(\mathbf{x})$ with $\mathbf{x}$ sampled from regular grids, and $\alpha \leq 1$.

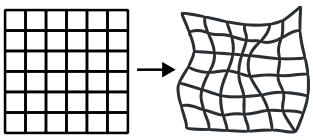


Figure 3: Deformed grid.

While previous approaches have considered engineering-level CAD designs [17] or lifted trajectories [16] requiring large amounts of storage, GMP offers significant advantages in versatility and data efficiency, with zero data storage needed and everything generated on-the-fly. Additionally, we show below that including advanced surface features like normals or curvature proves to be critical to capture complex PDE behavior.

3.2 Physics pre-training

For transient tasks, GMP focuses on masked autoencoding of PDE data that is generated online at training time. Extending the Tadpole [11] approach for structured datasets, we make use of spectral-based solvers for randomized canonical PDE tasks for rapid, online data generation. The models are exclusively pre-trained on static 3D single-channel data and can be effectively fine-tuned for novel multi-channel scenarios or lower dimensions, offering flexibility in channel depth and cross-dimensional transferability. A sketch of the data generation is provided in Figure 2b. The process begins by randomly selecting a PDE type, its physical parameters p, an initial condition IC, and the numerical resolution Res. Then, the spectral solver $\mathcal{P}$ computes the solution trajectory, from which a single snapshot is extracted

$$u_{\text{GRID}} = \mathcal{P}(t, p, IC, \text{Res}). \tag{5}$$

Given the varying resolutions and field counts of the raw simulations, the domain is cropped to a uniform fixed-resolution grid and a single random channel. Then, we select a random geometry x_S and a deformed volume grid x_V (from the geometric pre-training framework), and interpolate the 3D regular data at these coordinates $x = (x_S, x_V)$ to obtain the field values u. This point cloud serves as the model input. For neural operators that support queryable decoders, the sampling process is repeated to generate a different set (x', u') as the target for a masked reconstruction

$$u = \mathcal{I}(u_{\text{GRID}}, x), \qquad u' = \mathcal{I}(u_{\text{GRID}}, x'), \qquad \text{s.t. } x \cap x' = \emptyset, \tag{6}$$

where $\mathcal{I}$ is the interpolation operator. Then, the model's encoder E_θ and decoder D_θ are used to predict $\hat{u}'_\theta = D_\theta(E_\theta(x, u), x')$, and the final reconstruction task is defined as $\mathcal{L} = ||\hat{u}'_\theta - u'||_2^2$.

3.3 Fine-tuning

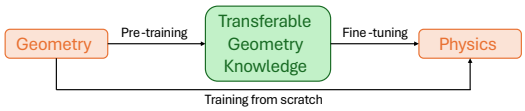


Figure 4: Geometric pre-training sketch.

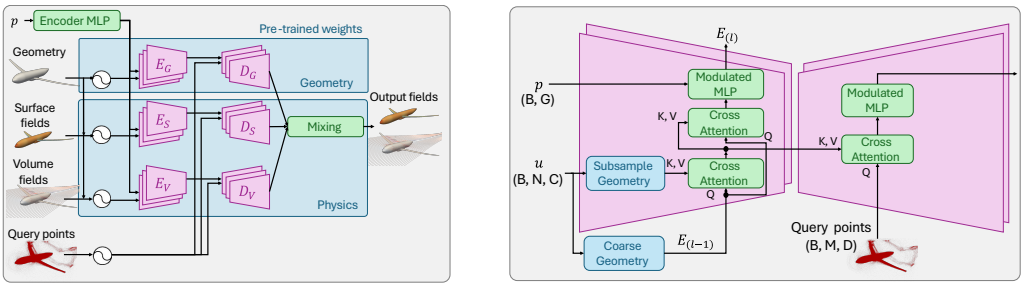
We evaluate pre-trained models against trained-from-scratch baselines across diverse steady-state and transient autoregressive downstream tasks. Note that the description of the pre-training method does not involve details of any particular architecture, making it model-agnostic. During fine-tuning, transient models transition from pre-training masked reconstruction to single-step dynamics. During pre-training, the model learns transferable spatial representations for static physics, and once this is learned, the fine-tuned model learns how to perform dynamics between these learned states.

Steady-state models map directly from geometry to physical target fields. Because in this setup the geometry fully determines the physical outputs, pre-learning to understand geometries helps to transfer knowledge into how fluids flow around these geometries. Both setups utilize full-parameter fine-tuning initialized from their respective pre-trained checkpoints. This process is sketched in Figure 4.

Because downstream tasks often present differing input and output channel dimensions, the first and last projection layers of the models are initialized from scratch. To safeguard pre-trained representations, backbone weights remain frozen during initial warmup epochs while optimizing only the projection layers. We enforce this identical fine-tuning protocol across all pre-trained baselines, including SMART, AB-UPT and Transolver++ [55].

3.4 Architecture

While the pre-training design described so far is agnostic to the specific architecture, assessing the most competitive architectures remains crucial for achieving optimal performance. Hence, we benchmark several modern neural operators: AB-UPT, Erwin, PTV3, Transolver++, GeoTransolver and SMART across three test cases DrivAerML [56], SHIFT-Wing [57], and SHIFT-Crash [58] (Appendix E). Our evaluation shows that SMART consistently achieves the highest standalone accuracy. As such, it presents the most challenging baseline to improve upon via pre-training and is chosen as the central architecture for the experiments below.



(a) General overview

(b) Details of each encoder-decoder block

Figure 5: Sketch of GMP-Net combining three encoders: one processing the general geometry of the object if available, and the other two the autoregressive physical fields. A decoder cross-attends the query positions to the output of the encoders. Finally, we compute a weighted sum using three learnable weights. Each encoder contains cross-attention from a set of anchor points to the rest of the point cloud, and an MLP modulated with external or simulation parameters.

Architectural extensions for transient dynamics Our ultimate goal is to design a modular architecture that excels across both steady-state and transient regimes while maximizing the benefits of pre-training. As the regular SMART architecture cannot perform autoregressive rollouts, we generalize it into a modular triple-encoder framework (Figure 5). A geometric encoder E_G embeds boundary shapes, while dedicated physical encoders E_S and E_V process surface and volumetric physical states u_t, respectively. Each branch applies sinusoidal positional encodings and compresses point clouds via anchor-based cross-attention into latent tokens z_G, z_S, z_V. Decoders then cross-attend target query coordinates x_Q to these latents to predict u_{t+1}, balanced by learned task weights w_i:

$$u_{t+1} = \sum_{i \in \{G,S,V\}} w_i \cdot D_i(z_i, x_Q), \qquad z_G = E_G(x_G, p); \quad z_{S,V} = E_{S,V}(x_{S,V}, u_{S,V}, p). \tag{7}$$

This decoupled formulation allows modular execution: branches are activated selectively based on problem requirements. For steady-state regimes, only the geometric branch (E_G, D_G) is active, naturally recovering the baseline SMART model. For transient problems, the surface (E_S, D_S) and volumetric (E_V, D_V) branches are toggled according to data availability. Crucially, this modularity facilitates decoupled pre-training: the geometric branch is pre-trained using geometric objectives (Section 3.1), while the physical branches are pre-trained online without stored simulation data (Section 3.2). We denote this new extended model as **GMP-Net**.

3.5 Downstream application datasets

To show pre-training impact for real-world scenarios, we evaluate on the following downstream engineering benchmarks unseen during pre-training.

Steady-State Datasets To rigorously assess steady-state operator performance, we evaluate on two primary benchmarks: DrivAerML [56], an open-source automotive benchmark comprising 500 geometric variations of the DrivAer notchback model, and SHIFT-Wing [57], an aerospace testbed featuring thousands of parametrically perturbed NASA Common Research Model (CRM) configurations. While DrivAerML isolates geometric variation under fixed inlet velocities, SHIFT-Wing systematically varies free-stream operating conditions. Representative geometries and flow fields are illustrated in Figure 6. For both datasets, our model predicts surface pressure p_s and wall shear stress τ, as well as volumetric pressure p_v and velocity fields $\mathbf{u}$. Finally, to evaluate transferability to fundamentally distinct physical regimes, we evaluate on SHIFT-Crash [58], an automotive structural mechanics benchmark.

Transient Datasets Given that transient benchmarks on unstructured meshes are relatively scarce, we employ a combination of synthetic experiments and established open datasets. First, we construct a synthetic testbed by interpolating planar Kuramoto–Sivashinsky (KS) solutions onto deformed, irregular grids. While pre-training leverages static 3D fields, this setup verifies whether the architecture generalizes across spatial dimensions and captures unfamiliar temporal dynamics. For

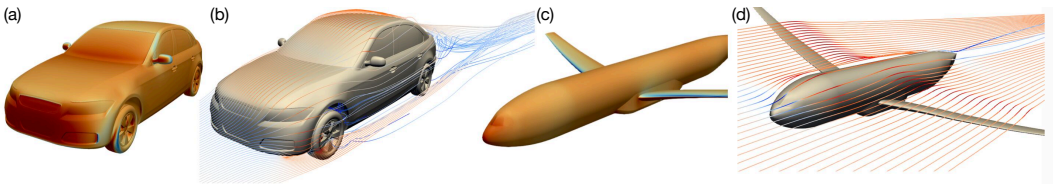


Figure 6: Samples showing surface (pressure in a,c) and volume (velocity streamlines in b,d) fields: DrivAerML (car, left) and SHIFT-Wing (aircraft, right).

Table 2: Overview of the fine-tuning datasets used for downstream tasks evaluation.

Dataset	Time	Dim.	Type	# Samples	# Points (S;V)	# Fields
DrivAerML [56]	Steady	3D	S; V	500	$\sim$ 9M ; 160M	4+4
SHIFT-Wing [57]	Steady	3D	S; V	6000	$\sim$ 3M ; 6M	4+4
KS (ours)	Transient	2D	V	630×30	- ; 4096	1
Ellipse [59]	Transient	2D	V	1200×100	- ; 1024	3
SHIFT-Crash [58]	Both	3D	S	768×12	$\sim$ 400K ; -	4

instance, to solve 2D downstream tasks, coordinates are zero-padded to $(x, y, 0)$. Second, we adopt the 2D flow past an ellipse from DGN [59] to predict time-dependent pressure and velocity fields. Lastly, we evaluate the dynamic response of SHIFT-Crash over the given 12 discrete time steps. For autoregressive rollout experiments, models unroll predictions across the entire temporal sequence; under the steady-state formulation, the model directly maps initial geometries and crash parameters to the final deformed state. Comprehensive benchmark details are provided in Table 2, where *Type* indicates volumetric (V) or surface-based (S) meshes, and the transient sample count denotes total trajectories multiplied by the temporal resolution per rollout.

4 Experiments

4.1 Results for steady-state problems

Accuracy To demonstrate the practical benefits of our pre-training, we adopt a low-data regime analysis. For all datasets, we utilize a subset of 144 samples, reserving a fixed random set of 16 samples for testing, and the remaining 128 samples are used to create nested training sets of varying sizes, allowing us to analyze how accuracy scales with the volume of training data. The results, summarized in Table 3, consistently demonstrate that our pre-training method enhances accuracy across all models in most data regimes. However, the behaviour is different across them. For GMP-Net and Transolver++ we observe a clear and expected trend: the accuracy gain is most pronounced in the sparsest data regimes, with a maximum of 47% improvement for 16 training samples. In addition, GMP-Net shows a more stable improvement across the experiments. For AB-UPT, we observe no measurable gains in highly data-sparse regimes due to severe overfitting: both the scratch and pre-trained variants rapidly converge to nearly identical error levels. With limited training data, model performance appears bounded by an accuracy plateau that both approaches reach almost immediately. Overall, our GMP-Net, derived from SMART, achieves the lowest errors.

Convergence and data saving Figure 7 illustrates the test loss trajectories on DrivAerML for our GMP-Net model. Comparing the number of optimization steps required for the pre-trained model to match the performance of the model trained from scratch highlights a substantial convergence speedup (Figure 9a). Furthermore, data-scaling experiments (Figure 8) show that fine-tuning on only 16 samples outperforms training from scratch with 32 samples, delivering significant sample efficiency gains as summarized in Figure 9b. Finally, evaluating lift and drag coefficients (C_L, C_D) confirms that GMP pre-training improves the agreement of predicted and reference lift coefficients, with the largest gains in the low-data regime (Figure 9c).

Computational efficiency GMP geometric pre-training requires only 4 hours on a single NVIDIA A100, negligible compared to the 80 hours required to train a baseline like SMART on SHIFT-Wing from scratch for 200 epochs. Thus, pre-training can be integrated as a light pre-processing step.

Table 3: Comparison between training from scratch and pre-training (PT) on steady-state tasks: relative L_2 error and pre-training improvement. Tests for different numbers of training samples. A Transolver++ variant pre-trained with GeoPT [16] is also shown for comparison.

	DrivAerML				SHIFT-Wing				SHIFT-Crash			
Training samples	16	32	64	128	16	32	64	128	16	32	64	128
Transolver++ (scratch) ↓	0.685	0.559	0.329	0.180	0.603	0.403	0.235	0.209	0.611	0.601	0.548	0.328
Transolver++ (GeoPT) ↓	0.520	0.443	0.309	0.218	0.308	0.266	0.233	0.204	0.488	0.428	0.410	0.388
Transolver++ (GMP ours) ↓	0.503	0.453	0.302	0.178	0.318	0.265	0.232	0.205	0.530	0.563	0.356	0.322
Rel. Improvement ↑	27%	19%	8%	1%	47%	34%	1%	2%	13%	6%	34%	2%
AB-UPT (scratch) ↓	0.439	0.313	0.244	0.184	0.569	0.456	0.379	0.298	0.396	0.363	0.340	0.326
AB-UPT (GMP ours) ↓	0.450	0.316	0.218	0.155	0.596	0.432	0.347	0.273	0.389	0.362	0.309	0.310
Rel. Improvement ↑	-2%	-1%	10%	16%	-5%	4%	8%	7%	2%	0%	9%	5%
GMP-Net (scratch) ↓	0.412	0.348	0.187	0.099	0.274	0.236	0.234	0.217	0.432	0.398	0.359	0.326
GMP-Net (PT ours) ↓	0.346	0.299	0.178	0.097	0.253	0.224	0.214	0.196	0.395	0.358	0.327	0.300
Rel. Improvement ↑	16%	14%	5%	3%	7%	5%	9%	10%	9%	10%	9%	8%

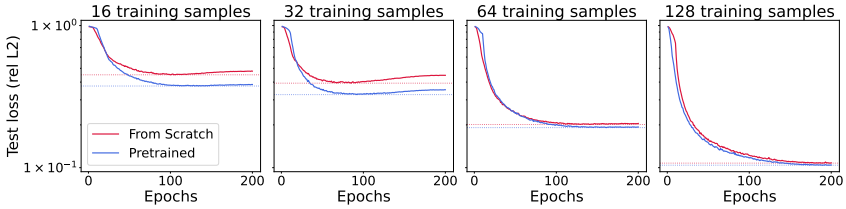


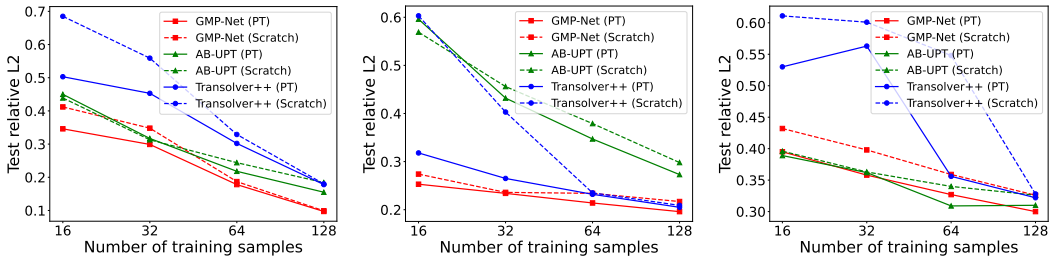
Figure 7: DrivAerML test loss of GMP-Net trained from scratch and with pre-training. Horizontal lines indicate the minimum error. Pre-trained models achieve lower error and faster convergence.

Comparison against other pre-training methods We compare our pre-training framework against GeoPT on Transolver++. Although official checkpoints are public, we pre-train and fine-tune GeoPT from scratch under a controlled reimplementation, maintaining identical model configurations across all experiments while evaluating different downstream datasets (original published numbers available in Appendix B for completeness). Under identical hardware, GeoPT pre-training requires $\sim$12 hours per epoch (taking $\sim$10 days to reach a test MSE of 0.02), whereas our disk-data-free pre-training converges in just 4 hours. As reported in Table 3 (under the Transolver++ setting), our model matches GeoPT's performance in low-data regimes, except for SHIFT-Crash, and outperforms it as the downstream training set scales. In contrast to our framework, Zhang et al. [17] relies on high-fidelity CAD geometries for pre-training, less practical in workflows where such meshes are proprietary or unavailable. Nonetheless, their reported 10–20% accuracy gains remain comparable to the improvements achieved by our method.

Geometric descriptors ablation We examine the contribution of individual geometric descriptors during pre-training. While prior works rely primarily on VDF [16] or binary occupancy [17], GMP integrates surface normals, curvature, and volumetric VDF. Table 4 evaluates different pre-training variants: surface features only (N and C), volumetric VDF alone, and their combination (all). In addition, we test pre-training using only binary occupancy (second row), the pre-training objective from Zhang et al. [17], but this fails to show any improvement in GMP and is therefore omitted from our final setup. Notably, VDF alone provides the weakest downstream transfer, whereas surface normals and curvature substantially improve representations. This aligns with aerodynamic intuition, as resolving steep boundary-layer gradients necessitates accurate differential surface geometry.

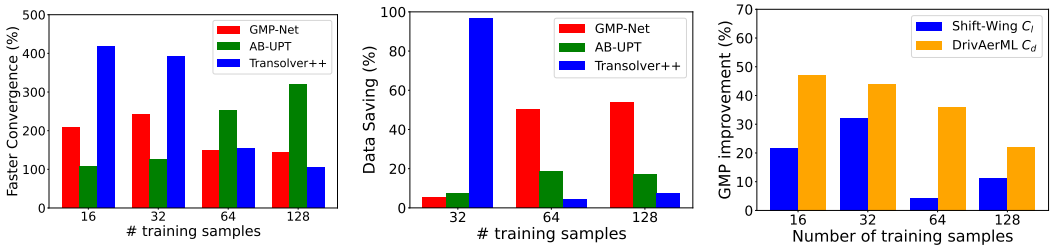
Table 4: Effect of geometric descriptors on GMP pre-training accuracy gains. Mean rel L_2 error.

	DrivAerML average ↓			
Training samples	16	32	64	128
GMP-Net (scratch)	0.412	0.348	0.187	**0.099**
GMP (occupancy)	0.424	0.379	0.243	0.138
GMP (VDF)	0.402	0.333	0.189	0.103
GMP (N and C)	**0.387**	**0.311**	**0.182**	0.101
GMP (all)	**0.346**	**0.299**	**0.178**	**0.097**



(a) DrivAerML. (b) SHIFT-Wing. (c) SHIFT-Crash.

Figure 8: Summary of model performance in steady problems. We show the final relative L_2 error for different fine-tuning dataset sizes.



(a) Convergence (DrivAerML). (b) Data efficiency (SHIFT-Wing). (c) Aerodynamic forces.

Figure 9: Summary of model performance in steady problems. a) Faster convergence analysis shows how much faster pre-trained models converge to the accuracy observed by scratch versions. b) Data efficiency showing how much data can be saved. c) R^2 improvement on the computation of the aerodynamic coefficients thanks to pre-training.

Table 5: NRMSE for transient problems. Comparison of from-scratch models and GMP pre-training.

	2D KS ↓			2D Ellipse ↓			3D SHIFT-Crash ↓		
Timesteps	1	5	10	1	5	10	1	5	10
Erwin	0.143	0.713	1.281	0.016	0.083	0.438	0.039	0.131	0.245
Transolver++	0.203	0.779	1.028	**0.006**	**0.021**	**0.042**	0.020	0.072	0.072
GMP-Net (scratch)	**0.030**	**0.196**	**0.615**	0.008	0.025	0.044	**0.009**	**0.030**	**0.040**
GMP-Net (PT)	**0.024**	**0.148**	**0.479**	**0.006**	**0.020**	**0.033**	**0.008**	**0.026**	**0.034**
Rel. Improvement	20%	24%	22%	25%	20%	23%	11%	13%	15%

4.2 Results for time-dependent problems

Table 5 compares rollout NRMSE against baselines, first showing that GMP-Net (scratch) already achieves the lowest error among the evaluated baselines for 2D KS and 3D SHIFT-Crash, and comparable to Transolver++ in the 2D Ellipse. Most importantly, across all scenarios, pre-training reduces error by 10–25%, demonstrating that inductive biases learned on simple PDEs transfer effectively to more complex systems. This transfer also bridges spatial dimensions and channel configurations, successfully mapping from 3D single-channel pre-training to 2D multi-channel tasks. Qualitative rollouts are illustrated in Figure 10 and Appendix I.

Finally, we analyze the accuracy gains as a function of the amount of training data. While the results in Table 5 use the complete datasets, results in Table 6 use only 16, 32, and 64 training trajectories. We compare the final 1-step test loss (NMSE) averaged across the three datasets, and we observe consistent accuracy gains, with larger absolute gains in the smaller-data regimes, but around 30% relatively in all three cases.

Table 6: 1-step NMSE test loss and GMP pre-training reduction.

# Trajectories	16	32	64
GMP-Net (scratch) $\times 10^{-3}$ ↓	8.8	6.9	5.4
GMP-Net (PT) $\times 10^{-3}$ ↓	6.2	4.8	3.9
Improvement ↑	29%	30%	28%

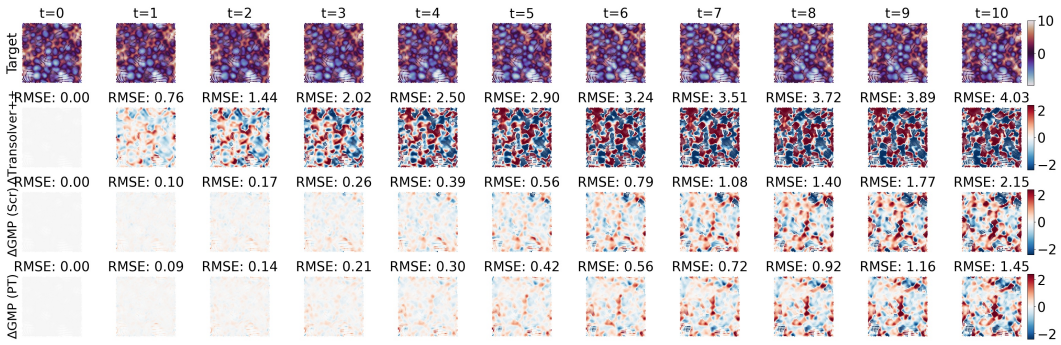


Figure 10: 2D KS equation rollout error comparison. GMP-Net shows state-of-the-art accuracy compared to other baselinse. Physics pre-trained GMP-Net exhibits improved performance over the model trained from scratch.

5 Conclusions, limitations and future work

We have introduced *GMP*, an innovative pre-training framework for unstructured simulations based on online generation of physics data and advanced geometric descriptors. GMP enhances the accuracy of diverse downstream tasks, achieving greater data efficiency and faster convergence. Since the pre-training setup is computationally lightweight, it can be integrated as a standard pre-processing step to improve performance without a substantial increase in total training time. A remaining challenge is mitigating overfitting in highly complex, low-data scenarios. Regarding geometric pre-training, interesting questions remain concerning the selection of geometric descriptors, and in the future we plan to explore whether additional features enhance model robustness and generalization. Importantly, unlike training from scratch, GMP amortizes initial computational costs by accelerating downstream convergence and reducing data requirements, ultimately making the design cycle more efficient. This paves the way for faster and more accurate surrogate models in science and engineering.

Acknowledgments and Disclosure of Funding

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.

We want to thank Michael Emory and the team at Luminary Cloud for the access to the SHIFT datasets. This work would not have been possible without their support and high-fidelity data. Special thanks goes to our colleague Rene Winchenbach for the many helpful discussions we had throughout the development of this project.

References

[1] J.D. Anderson. Computational Fluid Dynamics: The Basics with Applications. McGraw-Hill International Editions: Mechanical Engineering. McGraw-Hill, 1995. ISBN 9780071132107.

[2] Mingming Tong. Review of particle-based computational methods and their application in the computational modelling of welding, casting and additive manufacturing. Metals, 13(8), 2023. ISSN 2075-4701. doi: 10.3390/met13081392.

[3] G. Evans, J. Blackledge, and P. Yardley. Numerical Methods for Partial Differential Equations. Springer Undergraduate Mathematics Series. Springer London, 2012. ISBN 9781447103776. doi: 10.1007/978-1-4471-0377-6.

[4] Joel H. Ferziger, Milovan Perić, and Robert L. Street. Computational Methods for Fluid Dynamics. Springer, 2020. ISBN 978-3-319-99691-2. doi: 10.1007/978-3-319-99693-6.

[5] Mario Lino, Stathi Fotiadis, Anil A. Bharath, and Chris D. Cantwell. Current and emerging deep-learning methods for the simulation of fluid dynamics. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 479(2275):20230058, 2023. doi: 10.1098/rspa.2023.0058.

[6] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. Rev. Mod. Phys., 91: 045002, Dec 2019. doi: 10.1103/RevModPhys.91.045002.

[7] Ricardo Vinuesa and Steven L. Brunton. Emerging trends in machine learning for computational fluid dynamics. Computing in Science and Engg., 24(5):33–41, September 2022. ISSN 1521-9615. doi: 10.1109/MCSE.2023.3264340.

[8] N. Thuerey, B. Holzschuh, P. Holl, G. Kohl, M. Lino, Q. Liu, P. Schnell, and F. Trost. Physics-based deep learning, 2025. URL https://arxiv.org/abs/2109.05237.

[9] Wayne Xin Zhao Kun Zhou Junyi Li Tianyi Tang Zican Dong Yupeng Hou Beichen Zhang Yingqian Min Junjie Zhang Peiyu Liu Xiaolei Wang Yifan Du Chen Yang Yushuo Chen Zhipeng Chen Jinhao Jiang Ruiyang Ren Yifan Li Xinyu Tang Zikang Liu Yiwen Hu Jian-Yun Nie Ji-Rong Wen. A survey of large language models. Frontiers of Computer Science, 20:2012627–, 2026. ISSN 2095-2228. doi: https://doi.org/10.1007/s11704-026-60308-3. URL https://journal.hep.com.cn/fcs/EN/10.1007/s11704-026-60308-3.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[11] Qiang Liu, Felix Koehler, Benjamin Holzschuh, and Nils Thuerey. Tadpole: Autoencoders as foundation models for 3d pdes with online learning, 2026. URL https://arxiv.org/abs/2605.15284.

[12] Tung Nguyen, Arsh Koneru, Shufan Li, and Aditya Grover. PhysiX: A foundation model for physics simulations, 2025. URL https://arxiv.org/abs/2506.17774.

[13] Maximilian Herde, Bogdan Raonic, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=JC1VKK3UXk.

[14] Michael McCabe, Payel Mukhopadhyay, Tanya Marwah, Bruno Régaldo-Saint Blancard, François Rozet, Cristiana Diaconu, Lucas Thibaut Meyer, Kaze W. K. Wong, Mohammad-Hadi Sotoudeh, Alberto Bietti, Irina Espejo Morales, Rio Alexa Fear, Siavash Golkar, Tom Hehir, Keiya Hirashima, Geraud Krawezik, Francois Lanusse, Rudy Morel, Ruben Ohana, Liam Holden Parker, Mariel Pettee, Jeff Shen, Kyunghyun Cho, Miles Cranmer, and Shirley Ho. Walrus: A cross-domain foundation model for continuum dynamics. In Forty-third International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=kcyOjXoUZu.

[15] Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: auto-regressive denoising operator transformer for large-scale PDE pretraining. In Proceedings of the 41st International Conference on Machine Learning, ICML'24. JMLR.org, 2024.

[16] Haixu Wu, Minghao Guo, Zongyi Li, Zhiyang Dou, Mingsheng Long, Kaiming He, and Wojciech Matusik. GeoPT: Scaling physics simulation via lifted geometric pre-training. In ICLR 2026 Workshop on Foundation Models for Science: Real-World Impact and Science-First Design, 2026. URL https://openreview.net/forum?id=N9qIqvanBj.

[17] Zhizhou Zhang, Youjia Wu, Kaixuan Zhang, and Yanjia Wang. From cheap geometry to expensive physics: A physics-agnostic pretraining framework for neural operators. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=iCprPzyrRp.

[18] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for Reynolds-Averaged Navier–Stokes simulations of airfoil flows. AIAA Journal, 58(1):25–36, 2020. doi: 10.2514/1.J058291.

[19] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. Journal of Fluid Mechanics, 870:106–120, 2019. doi: 10.1017/jfm.2019.238.

[20] Rui Wang, K. Kashinath, M. Mustafa, A. Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020.

[21] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence, 3(3):218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5.

[22] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for parametric partial differential equations. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 10012–10022, October 2021.

[26] William Peebles and Saining Xie. Scalable diffusion models with transformers. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pages 4172–4182, 2023. doi: 10.1109/ICCV51070.2023.00387.

[27] Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for PDE surrogate modeling. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=djyn8Q0anK.

[28] Benjamin Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. PDE-transformer: Efficient and versatile transformers for physics simulations. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=3BaJMRaPSx.

[29] Benjamin Holzschuh, Georg Kohl, Florian Redinger, and Nils Thuerey. P3D: Scalable Neural Surrogates for High-Resolution 3D Physics Simulations with Global Context. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=8UdCE5nhFl.

[30] Hao Wei, Björn List, and Nils Thuerey. Revit: Rotational-equivariant vision transformers for neural PDE solvers. In Forty-third International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=fy2IXBBThK.

[31] B. List, M. Lino, and N. Thuerey. Rotational equivariant graph neural networks via local eigenbasis transformations. Physics of Fluids, 37(8):087178, 08 2025. ISSN 1070-6631. doi: 10.1063/5.0279499. URL https://doi.org/10.1063/5.0279499.

[32] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

[33] Mario Lino, Stathi Fotiadis, Anil A. Bharath, and Chris D. Cantwell. Multi-scale rotation-equivariant graph neural networks for unsteady Eulerian fluid dynamics. Physics of Fluids, 34(8):087110, 08 2022. ISSN 1070-6631. doi: 10.1063/5.0097679.

[34] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In Proceedings of the 30th International Conference on Neural Information Processing Systems, page 4509–4517. Curran Associates Inc., 2016. ISBN 9781510838819.

[35] Quercus Hernández, Alberto Badías, Francisco Chinesta, and Elías Cueto. Thermodynamics-Informed Graph Neural Networks. IEEE Transactions on Artificial Intelligence, 5(3):967–976, 2024. doi: 10.1109/TAI.2022.3179681.

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In International Conference on Learning Representations, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[37] Peng-Shuai Wang. Octformer: Octree-based transformers for 3d point clouds. ACM Trans. Graph., 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592131.

[38] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. Transactions on Machine Learning Research, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=EPPqt3uERT.

[39] Sepehr Mousavi, Shizheng Wen, Levi Lingsch, Maximilian Herde, Bogdan Raonic, and Siddhartha Mishra. RIGNO: A graph-based framework for robust and accurate operator learning for PDEs on arbitrary domains. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025. URL https://openreview.net/forum?id=ahJfROJOYt.

[40] Benedikt Alkin, Andreas Fürst, Simon Lucas Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal Physics Transformers: A framework for efficiently scaling neural operators. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=oUXiNX5KRm.

[41] Mattia Rigotti, Nicholas Thumiger, and Thomas Frick. GIST: Gauge-Invariant Spectral Transformers for scalable graph neural operators, 2026. URL https://arxiv.org/abs/2603.16849.

[42] Shizheng Wen, Arsh Kumbhat, Levi Lingsch, Sepehr Mousavi, Yizhou Zhao, Praveen Chandrashekar, and Siddhartha Mishra. Geometry aware operator transformer as an efficient and accurate neural surrogate for PDEs on arbitrary domains. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2026. URL https://openreview.net/forum?id=HXFvNkNt0n.

[43] Maksim Zhdanov, Max Welling, and Jan-Willem van de Meent. Erwin: A tree-based hierarchical transformer for large-scale physical systems. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=MrphqqwnKv.

[44] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point Transformer V3: Simpler, Faster, Stronger. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4840–4851, 2024. doi: 10.1109/CVPR52733.2024.00463.

[45] 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. Transactions on Machine Learning Research, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=nwQ8nitlTZ.

[46] Jan Hagnberger and Mathias Niepert. SMART: Scalable mesh-free aerodynamic simulations from raw geometries using a transformer-based surrogate model. In Forty-third International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=2dzU8sTOO3.

[47] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In Proceedings of the 41st International Conference on Machine Learning, ICML'24. JMLR.org, 2024.

[48] Corey Adams, Rishikesh Ranade, Ram Cherukuri, and Sanjay Choudhry. GeoTransolver: Learning physics on irregular domains using multi-scale geometry aware physics attention transformer, 2025. URL https://arxiv.org/abs/2512.20399.

[49] Neil Ashton, Johannes Brandstetter, and Siddhartha Mishra. Fluid intelligence: A forward look on AI foundation models in computational fluid dynamics, 2025. URL https://arxiv.org/abs/2511.20455.

[50] Yunjia Yang, Babak Gholami, Caglar Guerbuez, Mohammad Rashed, and Nils Thuerey. Toward a foundation-model paradigm for aerodynamic prediction in three-dimensional design. AIAA Journal, 0(0): 1–21, 0. doi: 10.2514/1.J066861. URL https://doi.org/10.2514/1.J066861.

[51] Yatian Pang, Wenxiao Wang, Francis E. H. Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II, page 604–621. Springer-Verlag, 2022. ISBN 978-3-031-20085-4. doi: 10.1007/978-3-031-20086-1_35.

[52] Anthony Zhou and Amir Barati Farimani. Masked autoencoders are PDE learners. Transactions on Machine Learning Research, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=rZNuiFwXVs.

[53] M. Pauly, M. Gross, and L.P. Kobbelt. Efficient simplification of point-sampled surfaces. In IEEE Visualization, 2002. VIS 2002., pages 163–170, 2002. doi: 10.1109/VISUAL.2002.1183771.

[54] Qiang Liu, Mengyu Chu, and Nils Thuerey. ConFIG: Towards conflict-free training of physics informed neural networks. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=APojAzJQiq.

[55] Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=AM7iAh0krx.

[56] Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schonwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, Burkhard Hupertz, and Danielle Maddix. DrivAerML: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics, 2025. URL https://arxiv.org/abs/2408.11969.

[57] Luminary Cloud. SHIFT-Wing: High-fidelity computational fluid dynamics dataset for transonic aerospace external aerodynamics, 2025. URL https://huggingface.co/datasets/luminary-shift/Wing.

[58] Luminary Cloud. SHIFT-Crash: High-fidelity crash simulation dataset for automotive structural mechanics, 2026. URL https://huggingface.co/datasets/luminary-shift/SHIFT-Crash.

[59] Mario Lino, Tobias Pfaff, and Nils Thuerey. Learning distributions of complex fluid simulations with diffusion graph networks. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=uKZdlihDDn.

Technical appendices and supplementary material

These appendices provide additional details on the following topics:

- A: Ablation studies.
- B: Details on the comparison against GeoPT.
- C: Training hyperparameters.
- D: Model and baselines hyperparameters.
- E: Baseline architectures comparison.
- F: Computation of the aerodynamic coefficients.
- G: Intersection-free meshes via Gaussian random fields.
- H: Additional results: regional breakdown.
- I: Additional plots and figures.

A Ablation studies for pre-training

We begin this appendix section by presenting ablation studies on key design choices across the two pre-training regimes.

A.1 Number of points during pre-training

Owing to the discretization-invariant nature of modern unstructured neural operators, pre-training does not strictly require the dense spatial resolutions utilized during downstream fine-tuning, where target meshes routinely scale to millions of vertices. This decoupling enables an advantageous trade-off between computational efficiency (at lower point densities) and representational fidelity (at higher geometric resolutions). Table 7 compares three candidate point cloud resolutions for pre-training, with our default configuration occupying the intermediate operating point. As demonstrated by the empirical trends, scaling the point budget further yields diminishing accuracy gains on downstream tasks while introducing prohibitive computational overhead during the pre-training phase.

Table 7: Performance and fine-tuning relative L_2 error on DrivAerML across different volume and surface point resolutions during pre-training (PT).

Vol. Points	Surf. Points	PT Time (h)	Rel L_2 Error FT (64 samples)
1024	512	2.5	0.194
2048	**1024**	**4.0**	**0.178**
4096	2048	9.0	0.175

A.2 Loss weighting during pre-training

During geometric pre-training, the model jointly predicts three distinct descriptors: local curvature κ, surface normals $\mathbf{n}$, and the vector distance function (VDF). A central question is how to properly balance these heterogeneous objective terms. We evaluate two distinct multi-task weighting strategies:

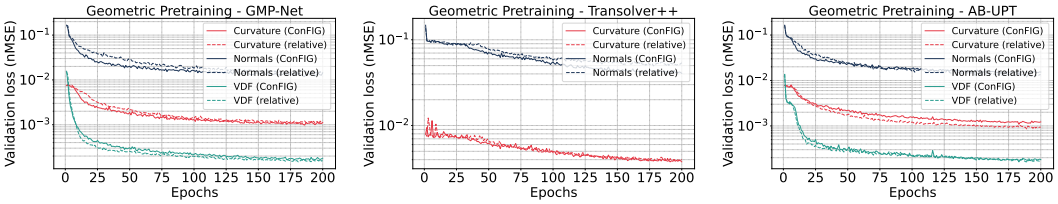
- **Dynamic Relative Weighting:** Rebalances the objective dynamically according to the running magnitude of each loss term, preventing dominant tasks from starving weaker gradients:

$$\mathcal{L} = w_\kappa^{(t)} \mathcal{L}_\kappa + w_n^{(t)} \mathcal{L}_n + w_V^{(t)} \mathcal{L}_V, \tag{8}$$

  where normalized weights are computed at iteration t as

$$w_k^{(t)} = K \cdot \frac{\tilde{w}_k^{(t)}}{\sum_{j \in \mathcal{T}} \tilde{w}_j^{(t)}}, \qquad \tilde{w}_k^{(t)} = \frac{1}{\bar{\mathcal{L}}_k^{(t)} + \epsilon}, \tag{9}$$

  with $\mathcal{T} = \{\kappa, n, V\}$, task count $K = |\mathcal{T}| = 3$, numerical stabilizer $\epsilon > 0$, and $\bar{\mathcal{L}}_k^{(t)}$ denoting the exponential moving average (EMA) of the k-th loss component.
- **Gradient Conflict resolution (ConFIG):** Simple scalarization approaches may suffer from destructive gradient interference across competing objectives. Rather than collapsing the multi-task objectives into a single scalar loss, we leverage the ConFIG algorithm [54], which performs gradient surgery at each optimization step to project task gradients onto a unified, conflict-free descent direction.



(a) GMP-Net. (b) Transolver++. (c) AB-UPT.

Figure 11: Geometry pre-training.

Figure 11 illustrates the evolution of each loss component throughout geometric pre-training, showing that ConFIG generally achieves superior convergence across all objectives. A critical question is whether these optimization gains translate into downstream fine-tuning performance. As detailed in Table 8 , the representations pre-trained with ConFIG consistently yield the lowest prediction errors upon downstream fine-tuning, validating that mitigating gradient conflict during multi-task pre-training produces more transferable geometric features.

Table 8: Performance evaluation (Relative L_2 error) on DrivAerML (64 samples).

Setting	**Transolver++**	**AB-UPT**	**GMP-Net**
Scratch	0.329	0.244	0.187
GMP relative	0.325	0.238	0.185
GMP ConFIG	0.302	0.218	0.178
Max. Imp	8%	17%	6%

A.3 Family of PDEs

Following the design philosophy of the Tadpole framework for regular grids [11], our physics pre-training strategy relies on representation learning via spatial autoencoding rather than learning multi-channel temporal trajectories. Consequently, the primary objective of PDE selection is not to match downstream dynamics, but to expose the spatial autoencoder to a comprehensive set of spatial features, including smooth diffusive gradients, sharp shock interfaces, and multi-scale turbulent vortices.

To achieve complete spectral coverage, the chosen suite of initial conditions (IC) and pre-training equations, e.g., diffusion, advection, wave equations, reaction-diffusion... was curated to cover the full spectrum of spatial wavenumbers. Sampling single-channel spatial crops on-the-fly ensures the representation remains system-agnostic and decoupled from any specific state vector or time-stepping scheme, while preserving power spectral density across high and low frequencies. Figure 12 shows the spectrum of the solutions to different PDEs with different IC, showing a very varied spectral behavior. Samples are cropped to 32^3 regions for our experiments.

A.4 Physics pre-training results

Because training samples are synthesized dynamically on-the-fly at each iteration, the optimization pipeline operates without a static validation set. Quantitatively, the self-supervised masked reconstruction objective demonstrates high fidelity, reaching a normalized mean squared error (NMSE) on the order of 10^{-4}. This applies for the pre-training autoencoding objective on single snapshots, without any dynamics. The model is later fine-tuned to learn dynamic evolution between consecutive time steps, naturally a more difficult task.

A.5 Model size

The majority of existing unstructured models [43, 42, 45, 55, 46] reported in literature utilize relatively modest parameter budgets, typically ranging between 1–5 M parameters. In contrast, models designed for structured physics data frequently scale to several million [28, 29] or even billions of parameters [14]. To investigate the scaling laws within our framework, Figures 13 and 14 illustrate the performance of two representative architectures, Erwin and SMART, when trained on the DrivAerML dataset. For this analysis, we evaluate three variants of SMART containing 11 M (small), 39 M (medium), and 79 M (large) parameters, alongside three versions of Erwin with 2 M (tiny), 8 M (small), and 30 M (medium) parameters. Our observations indicate distinct scaling behaviors: while Erwin appears more challenging to scale—primarily due to a heightened susceptibility to overfitting at higher capacities—SMART demonstrates consistent scaling behavior. However, it

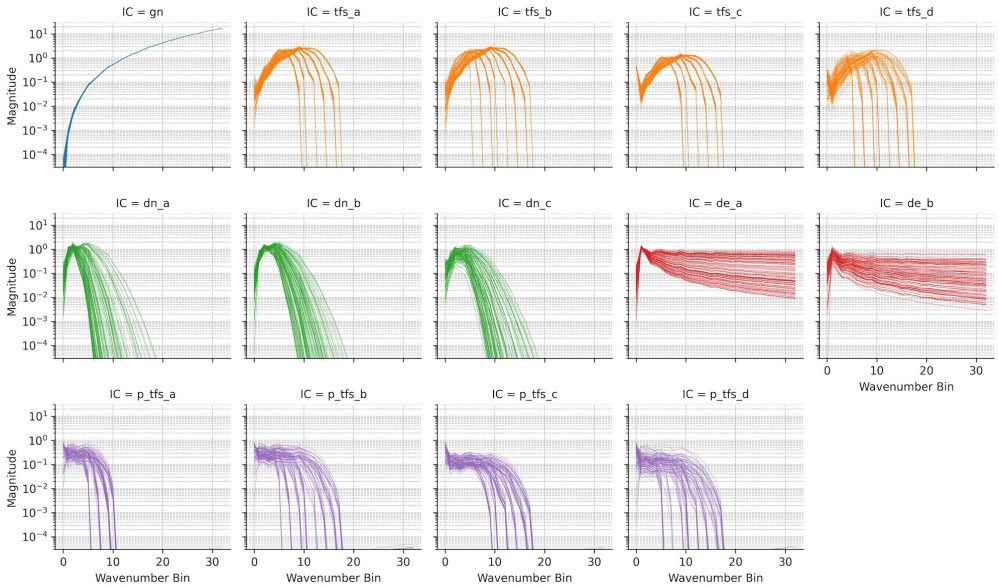


Figure 12: Spectra of different pre-training solutions. Source: Liu et al. [11].

should be noted that the performance gains for SMART diminish as capacity increases, yielding only marginal improvements at the highest parameter counts.

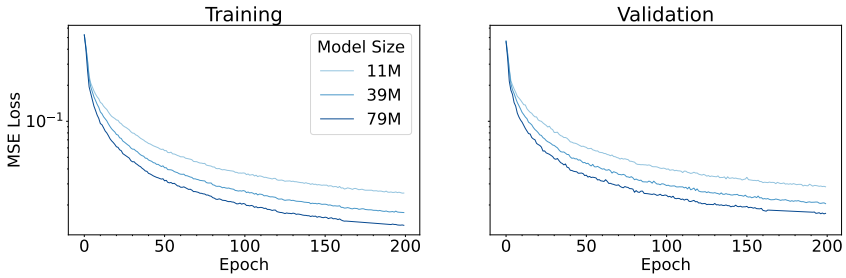


Figure 13: Scaling of the size of SMART.

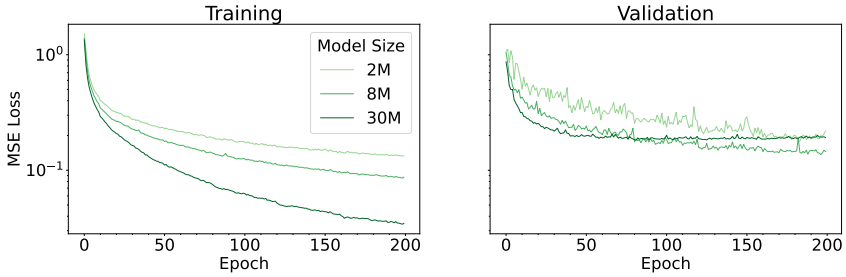
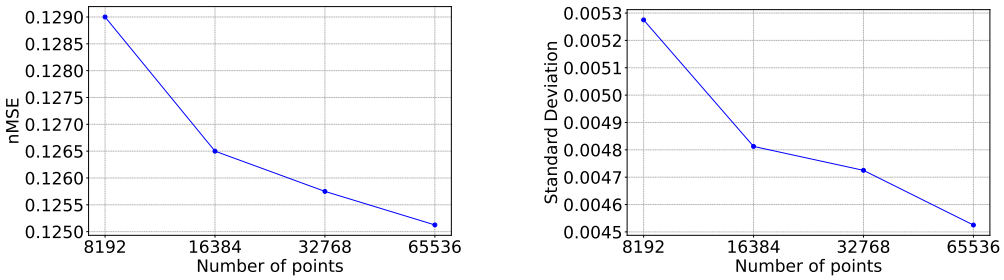


Figure 14: Scaling of the size of Erwin.

A.6 Number of points during fine-tuning

In addition, we conduct an ablation study regarding the density of the training point clouds. Current unstructured architectures often encounter out-of-memory (OOM) constraints when processing millions of points simultaneously; consequently, they are typically trained on subsampled representations. To investigate the sensitivity of model performance to this subsampling factor, Figure 15 illustrates the test nMSE evaluated on the entire samples as a function of the training point count. While the results follow the expected downward trend in error, the performance gain for the SMART model is relatively modest, with only a 3% improvement when increasing the training resolution from 2^{13} to 2^{16} points. However, a more significant impact is observed in the point-wise error distribution: the standard deviation across points decreases by 14% over the same range. This suggests that increasing the training point density significantly enhances local stability and reduces the occurrence of high-error outliers.



(a) Mean.

(b) Standard deviation.

Figure 15: DrivAerML final test nMSE for different sizes of the point cloud.

B More on the comparison against other pre-training strategies

Although GeoPT [16] suggests that static geometric pre-training alone yields negligible downstream gains, our findings demonstrate that static pre-training can substantially improve fine-tuning accuracy without relying on complex synthetic dynamics or high-fidelity CAD models as in Zhang et al. [17]. This performance gain is driven by three key components:

- **Enriched geometric descriptors:** Complementing signed distance fields (SDF) or binary occupancy with explicit surface normals and curvature significantly improves representation quality and downstream transfer.
- **Conflict-free optimization via ConFIG:** As demonstrated in our ablation study, scalarized multi-task loss balancing fails to fully exploit the geometric objectives due to conflicting gradient directions. ConFIG resolves this negative transfer across pre-training objectives.
- **Staged fine-tuning via backbone freezing:** Adapting the model to downstream dimensionalities requires training task-specific input and output projection layers from scratch. Implementing an initial warmup phase—where the pre-trained backbone remains frozen—prevents large early gradient updates from corrupting the learned feature space (representational collapse), preserving transfer performance.

Table 3 presents a direct comparison against GeoPT as a representative geometric pre-training baseline. To ensure a strictly controlled and equitable evaluation, we executed both the GeoPT pre-training phase and subsequent fine-tuning within our experimental framework, enforcing identical data partitions, backbone architectures, and optimization hyperparameters. Because the official GeoPT repository omits the end-to-end pre-training pipeline required to reproduce their published benchmarks out of the box, we re-implemented their pre-training methodology adhering strictly to the specifications detailed in their manuscript. While minor numerical discrepancies are unavoidable in the absence of an end-to-end public release, this unified setup ensures a faithful, apples-to-apples comparison. For completeness, Table 9 compiles the original numerical results reported by the GeoPT authors. These results are not directly comparable to ours. Although we both consider three similar downstream scenarios (car aerodynamics, aircraft aerodynamics, and automotive crash tests), only DrivAerML uses exactly the same dataset, while the other two are different. Besides, model sizes, train/test data splits, the particular output fields predicted, and other hyperparameters may also have a major influence on the final results. Despite these differences, the comparison still provides an informative external reference point.

Table 9: GeoPT comparison of relative L_2 error between Transolver (scratch) and GeoPT at 200 epochs, alongside relative improvement across varying sample sizes. Source: Wu et al. [16].

	DrivAerML		NASA CRM		Car Crash	
Training samples	32	64	32	64	32	64
Transolver (scratch) ↓	0.166	0.110	0.242	0.136	0.315	0.214
Transolver (GeoPT) ↓	0.126	0.090	0.203	0.113	0.235	0.192
Rel. Improvement ↑	24%	18%	16%	17%	25%	10%

C Training hyperparameters

We first detail the physics pre-training hyperparameters in Table 10, where H', W', and D' denote the final crop dimensions of the regular grid data. Since the underlying PDE data is generated at varying resolutions, this cropping procedure ensures a uniform input format across the pipeline. To ensure a broad range of spectral behaviors, we employ the diverse PDE types and initial conditions provided in the Tadpole framework [11]. These data generation details are particularly salient, as our online generation process represents a significant departure from the conventional disk-based training pipelines described in subsequent tables. For instance, since data is constantly being generated, there is no fixed definition of an epoch, and we simply assign it to 1000 samples.

Additionally, Table 11 provides the hyperparameters for geometric pre-training, e.g., the number of points used during pre-training: 1024 on the surface, and 2048 in the volume. These values are much smaller than those in the fine-tuning datasets (DrivAerML and SHIFT-Wing), but pre-training still captures relevant geometric information.

Table 12 covers the three transient benchmarks: the 2D Kuramoto-Sivashinsky (KS) equation, 2D flow around an ellipse, and 3D SHIFT-Crash, and Tables 13, 14 and 15 summarize the settings for the steady-state cases. For brevity, model-specific configurations are omitted from these summary tables and are instead detailed in the following section of the Appendix.

Table 10: Hyperparameters for the physics pre-training.

Category	Hyperparameter	Value / Description
Data generation	H', W', D'	32
	PDEs	Diffusion, Hyper-diffusion, Burgers, Korteweg-de Vries (KdV), Kuramoto-Sivashinsky (KS), Fisher-KPP, Swift-Hohenberg
	Initial conditions	Gaussian noise, Truncated Fourier series, Diffused noise, Decayed Energy, Poisson
	Resolution	64, 128, 256, 384
	# surface points	1024
	# volume points	2048
	# data GPUs	1
Optimization	Optimizer	AdamW
	Learning Rate	1e-4
	LR schedule	cosine
	Batch Size	32
	Weight Decay	10^{-5}
Training	Epochs	200
	Samples per epoch	1000
	Training time	42 hours
	Precision	Single (FP32)
	Hardware	2 x NVIDIA RTX A5000 Ampere (24 GB)

Table 11: Hyperparameters for the geometric pre-training.

Category	Hyperparameter	Value / Description
Data	# surface points	1024
	# volume points	2048
Optimization	Optimizer	AdamW
	Learning Rate	1e-4
	LR schedule	cosine
	Batch Size	8
	Weight Decay	10^{-5}
Training	Epochs	200
	Samples per epoch	1000
	Training time	4 hours
	Precision	Single (FP32)
	Hardware	1 x NVIDIA A100

Table 12: Hyperparameters for autoregressive training.

Category	Hyperparameter	Value / Description
Data	Training/Test split	KS: 580/50 trajectories Ellipse: 1000/200 trajectories Crash: 600/100 trajectories
Optimization	Optimizer	AdamW
	Max learning Rate	1e-4
	Lr. schedule	Cosine
	Batch Size	8
Training	Epochs	20
	Weights warmup	1 epoch
	Training time	8 hours (for GMP-Net)
	Precision	Single (FP32)
	Hardware	1 x NVIDIA RTX A5000 Ampere (24 GB)

Table 13: Hyperparameters for steady DrivAerML.

Category	Hyperparameter	Value / Description
Data	Training/Test split	(Varying size from 16 to 128)
	# points	32768 (S) + 32768 (V)
Optimization	Optimizer	LION
	Learning Rate	1e-4
	LR schedule	Cosine
	Batch Size	1
Training	Epochs	200
	Training time	12 hours (max for GMP-Net)
	Precision	Single (FP32)
	Hardware	1 x NVIDIA L40S

Table 14: Hyperparameters for steady SHIFT-Wing.

Category	Hyperparameter	Value / Description
Data	Training/Test split	(Varying size from 16 to 128)
	# points	32768 (S) + 32768 (V)
Optimization	Optimizer	LION
	Learning Rate	1e-4
	LR schedule	Cosine
	Batch Size	1
Training	Epochs	50
	Training time	20 hours (max for GMP-Net)
	Precision	Single (FP32)
	Hardware	1 x NVIDIA A100

Table 15: Hyperparameters for steady SHIFT-Crash.

Category	Hyperparameter	Value / Description
Data	Training/Test split	(Varying size from 16 to 128)
	# points	32768
Optimization	Optimizer	LION
	Learning Rate	1e-4
	LR schedule	Cosine
	Batch Size	1
Training	Epochs	50
	Training time	16 hours (max for GMP-Net)
	Precision	Single (FP32)
	Hardware	1 x NVIDIA H100

D Baselines hyperparameters

This section provides a comprehensive overview of the configurations for all models and baselines evaluated in this work, as detailed in Tables 16, 17, 18, and 19. The disclosure of these hyperparameters is intended to ensure full transparency and to provide the necessary technical specifications to facilitate the exact reproduction of our empirical results.

Table 16: GMP-Net/SMART model configuration for steady problems.

Hyperparameter	Value / Description
spatial dim	3
Surface channels	4
Volume channels	4
Parameter channels	16 (DrivAerML) 1 (SHIFT-Wing) 6 (SHIFT-Crash)
Latent dim	256
Latent geometry points	4096
Subsampled geometry points	16384
Num encoder/decoder blocks	8
Num heads	8
pos scale factor	1000
dropout	0.0

Table 17: GMP-Net model configuration for transient problems.

Hyperparameter	Value / Description
spatial dim	2 (KS) / 2 (Ellipse) / 3 (Crash)
Input surface channels	0 (KS) / 0 (Ellipse) / 1 (Crash)
Input volume channels	1 (KS) / 3 (Ellipse) / 0 (Crash)
Output surface channels	0 (KS) / 0 (Ellipse) / 1 (Crash)
Output volume channels	1 (KS) / 3 (Ellipse) / 0 (Crash)
Parameter channels	1
Latent dim	512
Latent geometry points	4096
Subsampled geometry points	4096
Num encoder/decoder blocks	1
Num heads	16
pos scale factor	1000
dropout	0.0

Table 18: AB-UPT model configuration.

Hyperparameter	Value / Description
ndim	3
input dim	3
output dim surface	4
output dim volume	4
dim	192
geometry depth	3
num heads	8
blocks	pscscs
num volume blocks	8
num surface blocks	8
radius	0.25
supernode fraction	0.125
num anchors	8192

Table 19: Transolver++ model configuration.

Hyperparameter	Value / Description
space dim	3
n layers	6
n hidden	512
n heads	16
dropout	0.0
activation	gelu
mlp ratio	1.0
fun dim	0
out dim	4
slice num	64
condition size	16 (DrivAerML)
	1 (SHIFT-Wing)
	6 (SHIFT-Crash)

E Base architecture selection

Although our physics and geometric pre-training setups could be applied independently to any backbone architecture, it is essential to evaluate these strategies using the most competitive architectures available. Establishing robust pre-training frameworks for high-performing architectures represents a more rigorous and relevant challenge than improving lower-performing baselines that naturally possess more headroom for improvement. Below, we compare several recent models [45, 46, 43, 55] to establish an architectural basis. To ensure a fair comparison, all models are configured with approximately 10M parameters. One key distinction must be noted: while SMART [46] and AB-UPT [45] are designed to process surface and volume data simultaneously, other models do not inherently distinguish between the two. Consequently, we train those models separately for surface and volume tasks with a halved parameter count to maintain a consistent total. To evaluate these models, we utilize some of the most established high-fidelity datasets in the community: DrivAerML [56], SHIFT-Wing [57] and SHIFT-Crash [58]. As our primary metric, we use the relative L_2 error. All models are trained by randomly subsampling the full point cloud to $2^{15} = 32,768$ points for surface and volume. As SMART consistently demonstrates superior accuracy across both datasets for both surface and volume data, we focus the analysis on this architecture. Table 20 shows the exact averaged results. Finally, Tables 21, 22 and 23 show the autoregressive results from the main manuscript with the corresponding standard deviations.

Table 20: Test relative L_2 error (%) averaged across surface and volume fields for DrivAerML, SHIFT-Wing, and SHIFT-Crash benchmarks.

Model	DrivAerML		SHIFT-Wing		SHIFT-Crash*
	Surface	Volume	Surface	Volume	Surface
PTV3	31.33	44.31	20.87	46.76	35.25
Erwin	28.52	26.67	11.52	38.67	42.15
Transolver++	11.32	13.04	7.67	24.04	23.97
GeoTransolver	10.33	10.90	7.50	24.90	23.39
AB-UPT	10.76	8.83	10.07	25.11	39.65
SMART	7.93	7.01	7.95	9.36	20.56

Table 21: 2D KS equation. Mean RMSE for timesteps 1, 5, and 10.

Model	1	5	10
Erwin	0.143 ± 0.011	0.713 ± 0.043	1.281 ± 0.073
Transolver++	0.203 ± 0.014	0.779 ± 0.052	1.028 ± 0.035
GMP-Net (scratch)	0.030 ± 0.003	0.196 ± 0.039	0.615 ± 0.060
GMP-Net (PT)	0.024 ± 0.002	0.148 ± 0.030	0.479 ± 0.073

Table 22: 2D flow around an ellipse. Mean RMSE for timesteps 1, 5, and 10.

Model	1	5	10
Erwin	0.016 ± 0.002	0.083 ± 0.034	0.438 ± 0.244
Transolver++	0.006 ± 0.001	0.021 ± 0.003	0.042 ± 0.005
GMP-Net (scratch)	0.008 ± 0.001	0.025 ± 0.043	0.044 ± 0.009
GMP-Net (PT)	0.006 ± 0.001	0.020 ± 0.004	0.033 ± 0.011

Table 23: Deformation and von Mises stress average of SHIFT-Crash. Mean RMSE for timesteps 1, 5, and 10.

Model	1	5	10
Erwin	0.039 ± 0.001	0.131 ± 0.034	0.245 ± 0.021
Transolver++	0.020 ± 0.016	0.072 ± 0.039	0.072 ± 0.021
GMP-Net (scratch)	0.009 ± 0.008	0.030 ± 0.017	0.040 ± 0.024
GMP-Net (PT)	0.008 ± 0.007	0.026 ± 0.018	0.034 ± 0.013

To continue the arguments on the selection of the backbone architecture, a summary of computational efficiency is provided in Table 24. This is tested with subsampled DrivAerML samples on an NVIDIA L40S GPU. While Erwin and Transolver++ exhibit significantly better efficiency, the superior accuracy demonstrated by SMART makes it the most compelling candidate for our pre-training framework.

Table 24: Model complexity and computational efficiency comparison (batch size 1).

Model	Size (M) ↓	VRAM (GB) ↓	Throughput (it/s) ↑
Erwin [43]	**6.8**	**2.15**	**2.45**
Transolver++ [55]	6.9	**3.50**	**3.27**
AB-UPT [45]	11.4	12.79	0.47
SMART [46]	11.4	9.58	0.70

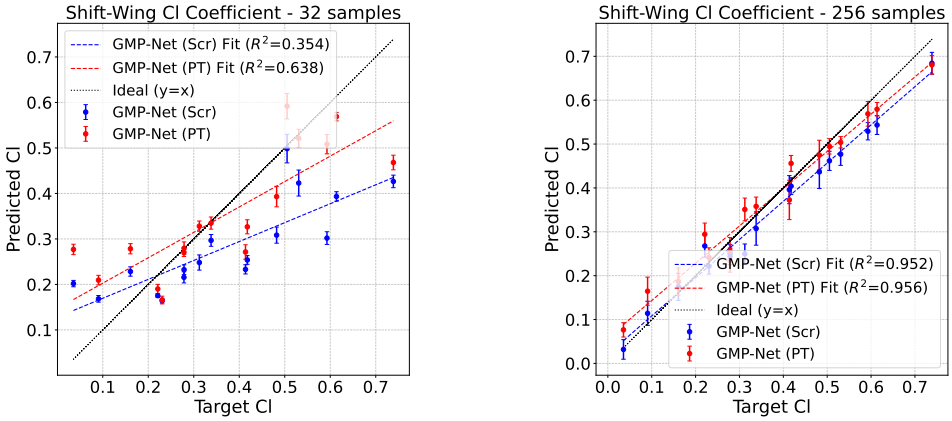
F Aerodynamic forces calculation

For real-world applications, the calculation of the aerodynamic forces acting on an object is the ultimate goal. The total aerodynamic force $\mathbf{F}$ is driven by pressure differences and viscous stress. Together, the total force is computed via a surface integral that is numerically solved with a simple quadrature rule

$$\mathbf{F} = \int_S (-p\mathbf{n} + \tau) dS \approx \sum_i (-p_i \mathbf{n}_i + \tau_i) S_i \tag{10}$$

where we sum over all the surface cells, p_i and τ_i are the pressure and wall shear stress at the center of the i-th cell, $\mathbf{n}_i$ its normal and S_i its area. Then, the drag is simply the horizontal force along the moving direction, and lift the vertical component. For aerospace applications, we must additionally account for the angle of attack.

Figure 16 shows the lift coefficient for SHIFT-Wing test samples when using 32 and 256 training samples. It provides a comparison between the trained from scratch GMP-Net, and our pre-trained results. While training with only 32 samples is too few to accurately predict the aerodynamic forces of test samples, both cases clearly show increased accuracy of our GMP pre-training.



(a) C_l for 32 samples.

(b) C_l for 256 samples.

Figure 16: Lift coefficient computation for SHIFT-Wing test samples when using 32 and 256 training samples. While the 32 case seems to be too sparse to accurately predict the aerodynamic forces of test samples, both cases clearly show increased accuracy of our GMP pre-training.

G Intersection-free meshes via Gaussian random fields

As analytic mesh generation component if $\mathcal{G}$ in Section 3.1 we employ intersection-free deformations generated by frequency controlled Gaussian random fields. The following section details this step in two dimensions for brevity, it naturally extends to three spatial dimensions.

The construction follows three key steps: (1) Generate smooth Gaussian random fields (u, v). (2) Normalize them to satisfy a global Lipschitz bound ($|D\mathbf{d}| < 1$). (3) Apply the deformation to the grid nodes. This yields a random but globally valid deformation, combining smoothness (from the Gaussian field), randomness (from spectral sampling), and strict geometric guarantees (from the Lipschitz constraint).

We consider a regular Cartesian grid over a rectangular domain $\Omega = [0, L_x] \times [0, L_y]$, with grid points $(x_i, y_j), \quad i = 1, \ldots, N, ; j = 1, \ldots, M$. Our goal is to construct a random, smooth deformation

$$\Phi : \Omega \to \mathbb{R}^2, \quad \Phi(x, y) = (x, y) + \mathbf{d}(x, y), \tag{11}$$

such that the resulting mesh is non-overlapping and orientation-preserving, i.e.

$$\det D\Phi(x, y) > 0 \quad \forall (x, y) \in \Omega. \tag{12}$$

We define the displacement field $\mathbf{d}(x, y) = \big(u(x, y)\, v(x, y)\big)$, where u and v are independent realizations of smooth Gaussian random fields. They are constructed via a spectral representation:

$$u(x, y) = \mathcal{F}^{-1}\left(\hat{\eta}(k_x, k_y), \sqrt{S(k_x, k_y)}\right), \tag{13}$$

where: $\hat{\eta}$ is complex white noise in Fourier space, and $S(k_x, k_y)$ is a power spectrum of the form $S(k_x, k_y) = \exp\big(-\ell^2(k_x^2 + k_y^2)\big)$, which enforces smoothness with correlation length ℓ. The same construction is used for $v(x, y)$. The deformation gradient (Jacobian) is given by

$$D\Phi(x, y) = I + D\mathbf{d}(x, y) \begin{pmatrix} 1 + \partial_x u & \partial_y u \\ \partial_x v & 1 + \partial_y v \end{pmatrix}. \tag{14}$$

with the determinant $\det D\Phi = (1 + \partial_x u)(1 + \partial_y v) - (\partial_y u)(\partial_x v)$. Then a necessary and sufficient condition for no local inversion is $\det D\Phi(x, y) > 0 \quad \forall (x, y)$. Instead of directly enforcing positivity of the determinant, we impose a stronger but simpler condition:

$$|D\mathbf{d}(x, y)|_2 \leq L < 1 \quad \forall (x, y). \tag{15}$$

This implies that all eigenvalues of $D\mathbf{d}$ satisfy $|\lambda_i| < 1$, $\det D\Phi > 0$, and thus Φ is locally invertible everywhere. This condition ensures global injectivity on Ω, i.e. no self-intersections.

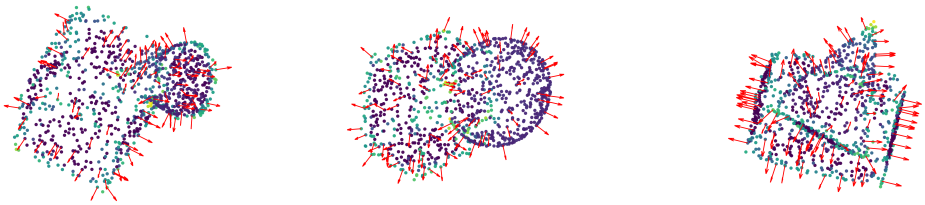
Given a sampled displacement field, we estimate the maximum L via a Frobenius norm and then rescale the field:

$$\tilde{\mathbf{d}}(x, y) = \frac{L_{\text{target}}}{L_{\max} + \varepsilon} \mathbf{d}(x, y). \tag{16}$$

This ensures that the final deformation $\Phi(x, y, \alpha) = (x, y) + \alpha\tilde{\mathbf{d}}(x, y)$ is intersection free for $\alpha \leq 1$. Each quadrilateral cell is defined by four neighboring vertices. To verify validity, we compute the signed area via a triangle decomposition and ensure that $A > 0$ everywhere. Thus, we realize our shape family $\mathcal{S}$ from Section 3.1 via $\Phi(x, y, \alpha)$ with $\alpha \sim [0.5, 1.0]$. We did not find it necessary to vary the spectral content of the deformation fields, and use a constant $\ell = 0.1$. We experimented with various correlation lengths and amplitudes, observing minimal performance variation, and selected a heuristic range bounding point shifts within one cell width to prevent extreme distortions.

Furthermore, our fine-tuning dataset for the 2D KS equation contains out-of-distribution samples with $\alpha = 1.4$. Thus, evaluation on the KS test set (which includes larger deformations) confirms that model performance generalizes robustly without degradation, justifying that more ablation of these parameters are not necessary.

Finally, Figure 17 shows examples of our pre-training shapes, generated as assemblies of geometry primitives such as polyhedra, spheres, and tetrahedra.



(a) Sample 1 (b) Sample 2 (c) Sample 3

Figure 17: Example of random pre-training samples built as assemblies of primitive shapes (cubes, spheres, tetrahedra). The plots show the surface curvature and the normals.

H Additional results

Fine-tuning in complete datasets Prioritizing performance in low-data regimes is central to the core mission of foundation models in scientific machine learning. In real-world engineering workflows, generating massive 3D CFD/FEM datasets is computationally prohibitive—for instance, generating a single sample in DrivAerML required 61,000 CPU hours. Practical engineering teams operate predominantly in data-constrained settings. Prior pre-training frameworks exhibit the exact same trend; for instance, GeoPT achieves its highest relative gains under lower sample counts. Nonetheless, the transient results in the main text already show that pre-training can enhance accuracy even when working with complete big datasets. Besides, for steady-state problems, in Table 25 we now show that pre-training still provides slight improvements in accuracy when using complete datasets. However, these improvements are marginal for some models. Still, even in data regimes where accuracy gains plateau, pre-training significantly accelerates training convergence speed, offering substantial practical savings in training time and computational resources.

Table 25: Relative L_2 error and pre-training improvement across models for the complete DrivAerML dataset.

(Entire dataset)	DrivAerML
Transolver++ (scratch) ↓	0.120
Transolver++ (PT) ↓	0.104
Rel. Improvement ↑	13%
AB-UPT (scratch) ↓	0.098
AB-UPT (PT) ↓	0.093
Rel. Improvement ↑	5%
GMP-Net (scratch) ↓	0.075
GMP-Net (PT) ↓	0.074
Rel. Improvement ↑	1%

Regional breakdown and theoretical insights Figure 1 demonstrates that point-wise prediction errors are substantially reduced across aerodynamically critical regions, such as the side mirrors and front bumper of automotive geometries. A detailed regional breakdown quantifying these performance improvements across distinct geometric components is provided in Table 26.

Table 26: Aerodynamic regional breakdown for automotive and aerospace benchmarks. 64 training samples with GMP-Net.

Benchmark	Region	Improvement (%)
DrivAerML	Front	23.5
	Mid car	10.0
	Underbody	17.4
	Tail / Diffuser	19.8
SHIFT-Wing	Leading edge	21.6
	Upper surface	15.3
	Lower surface	22.1

I Extra plots

In the final section of the Appendix we provide additional plots and figures with qualitative results.

First, in Figures 18-26 we provide the fine-tuning loss evolutions for all the combinations of models and datasets. Secondly, Figures 27 and 28 show the improvement in convergence and data saving for all the previous combinations. Finally, Figures 29-77 show visual results for steady and autoregressive experiments.

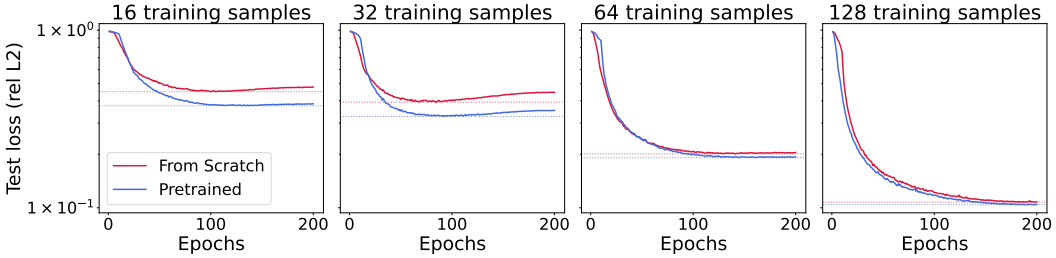


Figure 18: Test loss of GMP-Net trained from scratch and with pre-trained weights. Horizontal lines indicate the minimum achieved error. Pre-trained models achieve lower error and faster convergence on DrivAerML.

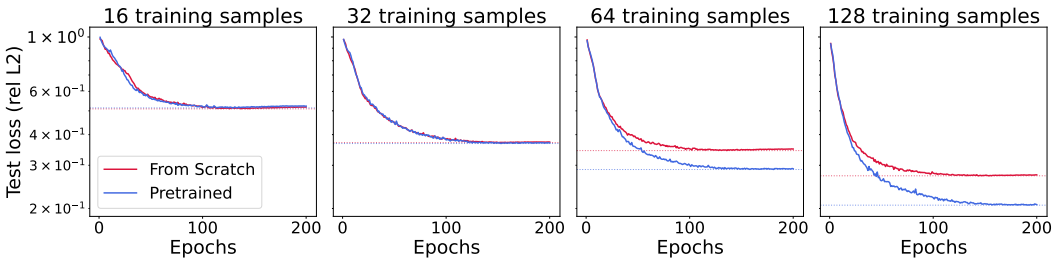


Figure 19: Test loss of AB-UPT trained from scratch and with pre-trained weights. Horizontal lines indicate the minimum achieved error. Pre-trained models achieve lower error and faster convergence on DrivAerML.

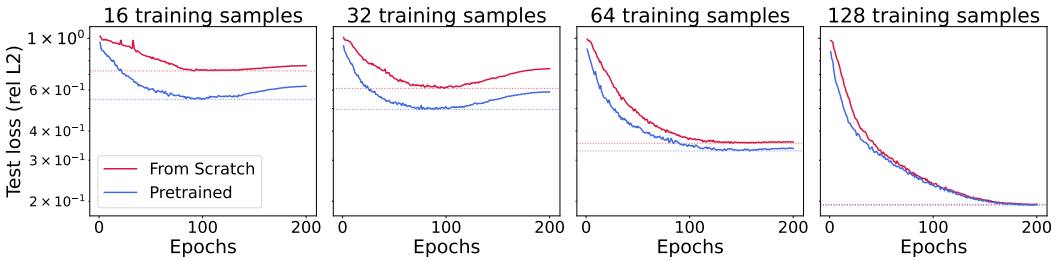


Figure 20: Test loss of Transolver++ trained from scratch and with pre-trained weights. Horizontal lines indicate the minimum achieved error. Pre-trained models achieve lower error and faster convergence on DrivAerML.

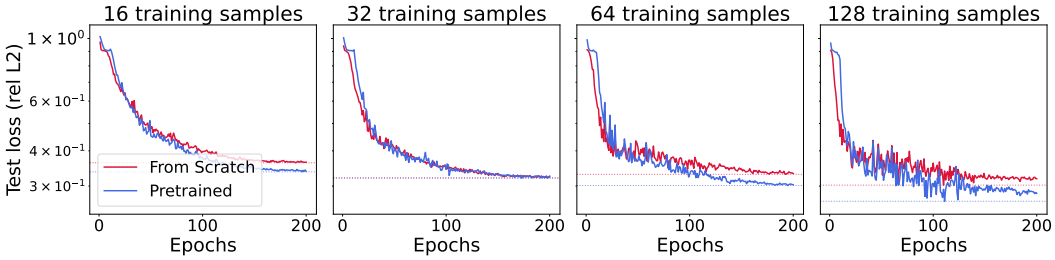


Figure 21: Test loss of GMP-Net trained from scratch and with pre-trained weights. Horizontal lines indicate the minimum achieved error. Pre-trained models achieve lower error and faster convergence on SHIFT-Wing.

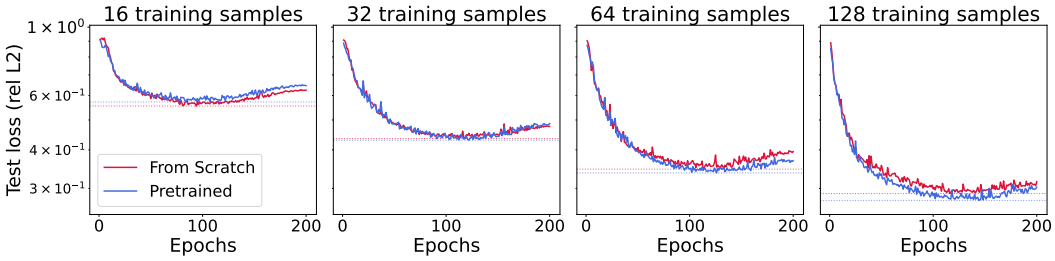


Figure 22: Test loss of AB-UPT trained from scratch and with pre-trained weights. Horizontal lines indicate the minimum achieved error. Pre-trained models achieve lower error and faster convergence on SHIFT-Wing.

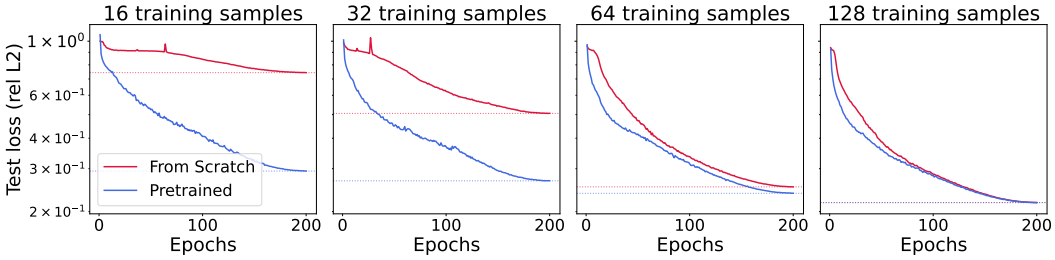


Figure 23: Test loss of Transolver++ trained from scratch and with pre-trained weights. Horizontal lines indicate the minimum achieved error. Pre-trained models achieve lower error and faster convergence on SHIFT-Wing.

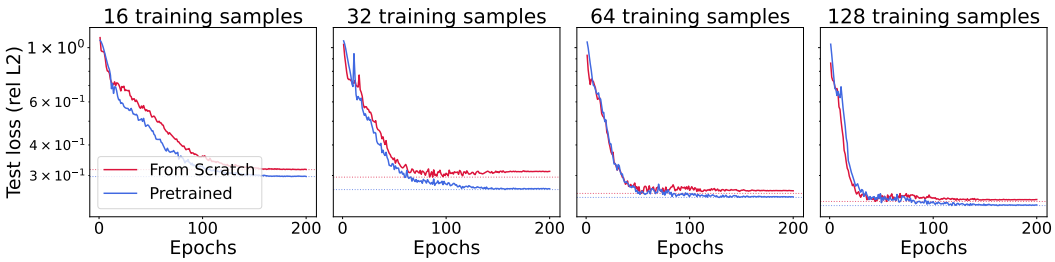


Figure 24: Test loss of GMP-Net trained from scratch and with pre-trained weights. Horizontal lines indicate the minimum achieved error. Pre-trained models achieve lower error and faster convergence on SHIFT-Crash.

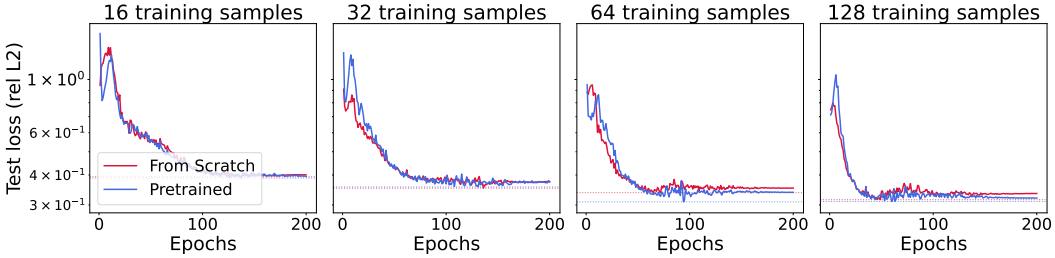


Figure 25: Test loss of AB-UPT trained from scratch and with pre-trained weights. Horizontal lines indicate the minimum achieved error. Pre-trained models achieve lower error and faster convergence on SHIFT-Crash.

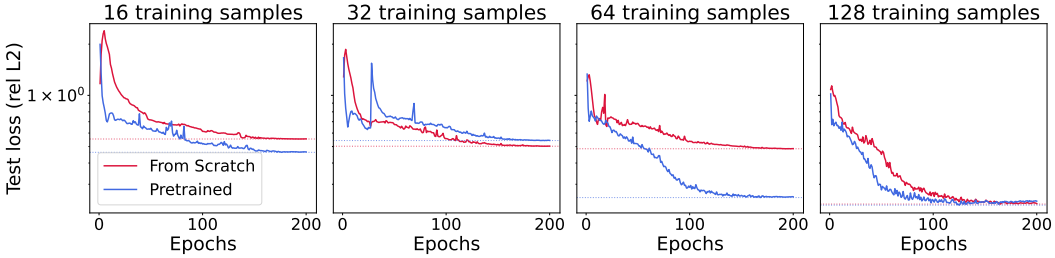
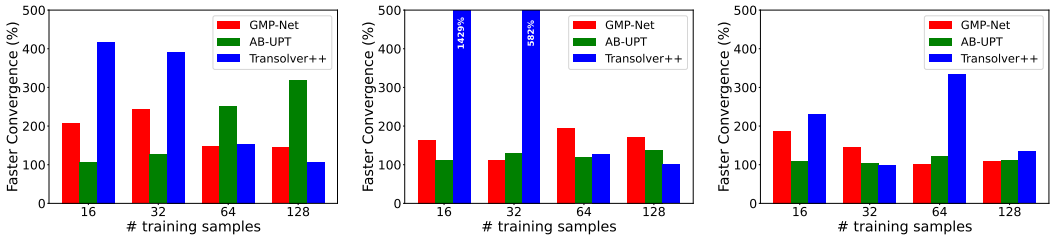
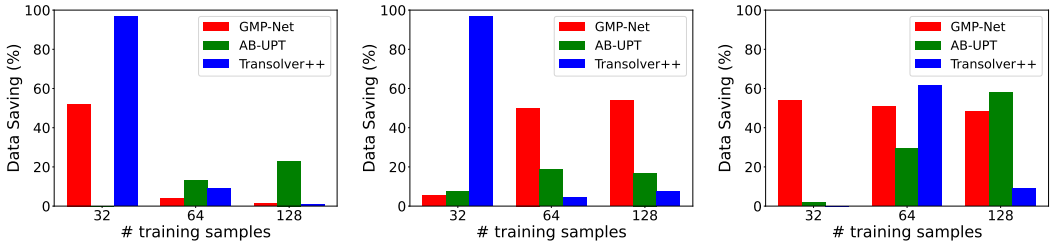


Figure 26: Test loss of Transolver++ trained from scratch and with pre-trained weights. Horizontal lines indicate the minimum achieved error. Pre-trained models achieve lower error and faster convergence on SHIFT-Crash.



(a) DrivAerML. (b) SHIFT-Wing. (c) SHIFT-Crash.

Figure 27: Faster convergence improvement of GMP pre-training.



(a) DrivAerML. (b) SHIFT-Wing. (c) SHIFT-Crash.

Figure 28: Data saving due to GMP pre-training.

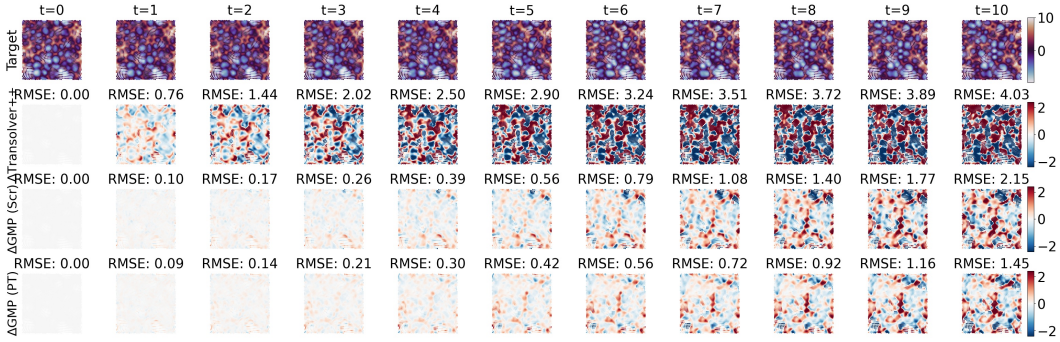


Figure 29: 2D KS equation rollout error comparison (sample 1). GMP successfully improves the accuracy over the from-scratch variant for all timesteps. GMP-Net also achieves lower errors than Transolver++.

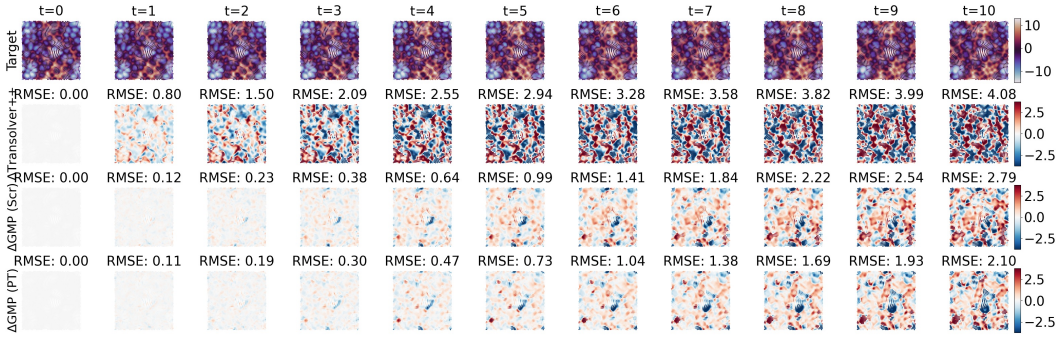


Figure 30: 2D KS equation rollout error comparison (sample 2). GMP successfully improves the accuracy over the from-scratch variant for all timesteps. GMP-Net also achieves lower errors than Transolver++.

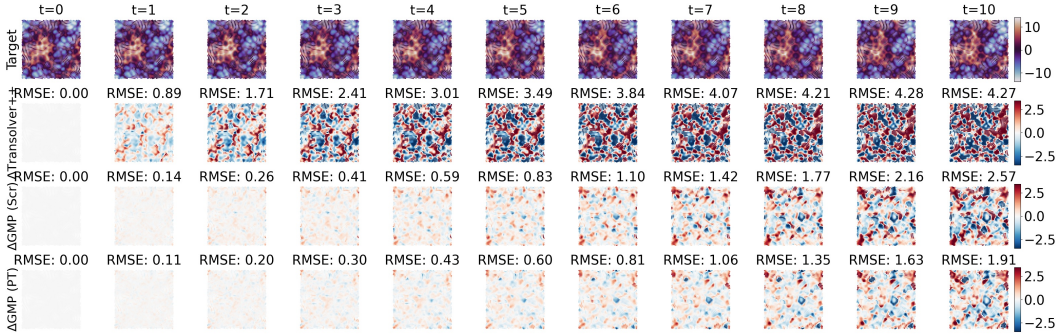


Figure 31: 2D KS equation rollout error comparison (sample 3). GMP successfully improves the accuracy over the from-scratch variant for all timesteps. GMP-Net also achieves lower errors than Transolver++.

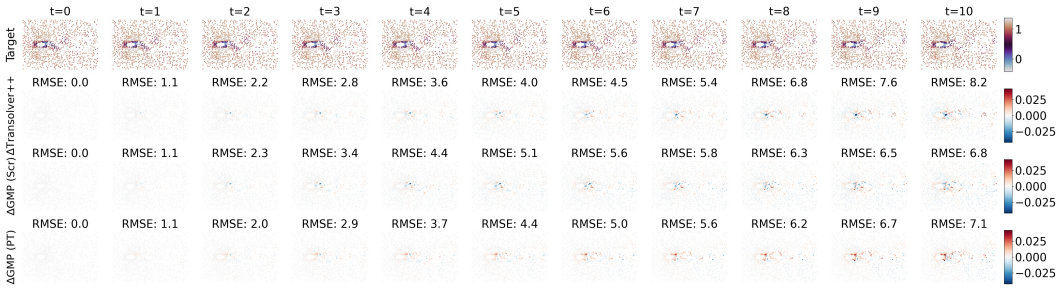


Figure 32: 2D flow around an ellipse rollout error comparison (sample 1). GMP successfully improves the accuracy over the from-scratch variant for all timesteps. Compared to Transolver++, GMP-Net achieves lower errors in longer rollouts, but higher in the intermediate timesteps.

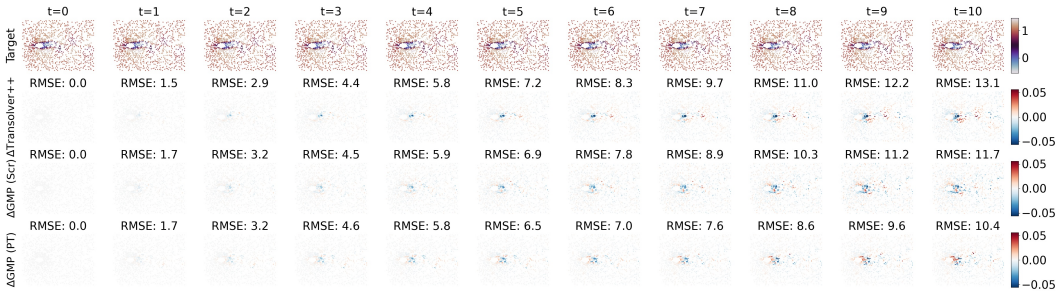


Figure 33: 2D flow around an ellipse rollout error comparison (sample 2). GMP successfully improves the accuracy over the from-scratch variant for all timesteps. Compared to Transolver++, GMP-Net achieves lower errors in longer rollouts, but higher in the intermediate timesteps.

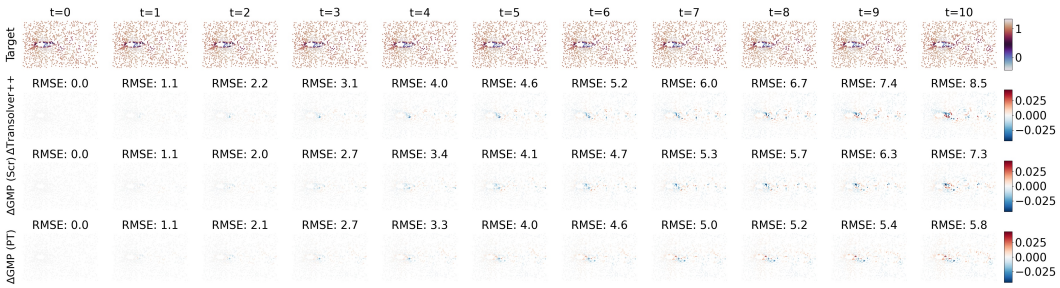


Figure 34: 2D flow around an ellipse rollout error comparison (sample 3). GMP successfully improves the accuracy over the from-scratch variant for all timesteps. Compared to Transolver++, GMP-Net achieves slightly lower errors overall.

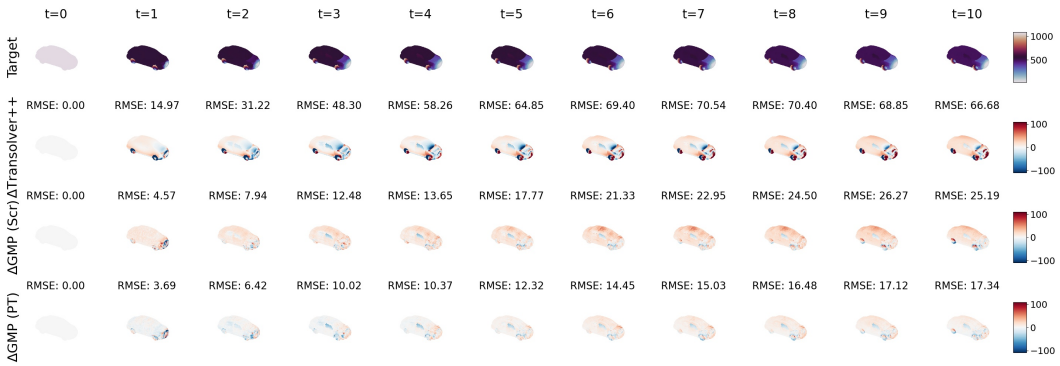


Figure 35: Crash deformation rollout error comparison (sample 1). GMP successfully improves the accuracy over the from-scratch variant. Compared to Transolver++, GMP-Net achieves lower errors.

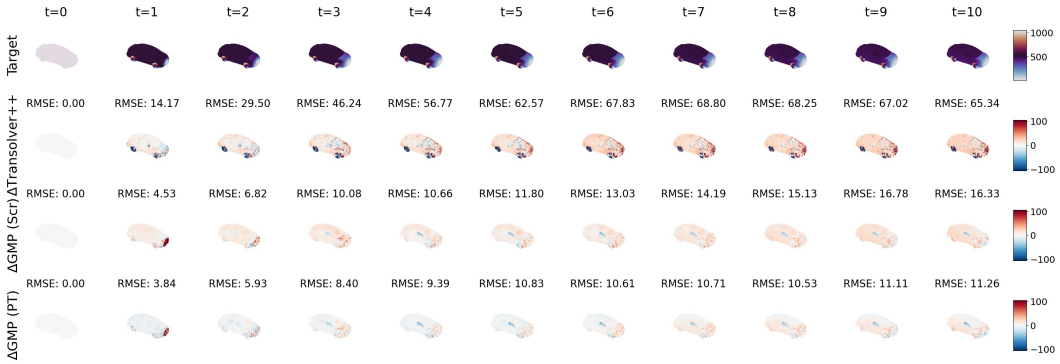


Figure 36: Crash deformation rollout error comparison (sample 2). GMP successfully improves the accuracy over the from-scratch variant. Compared to Transolver++, GMP-Net achieves lower errors.

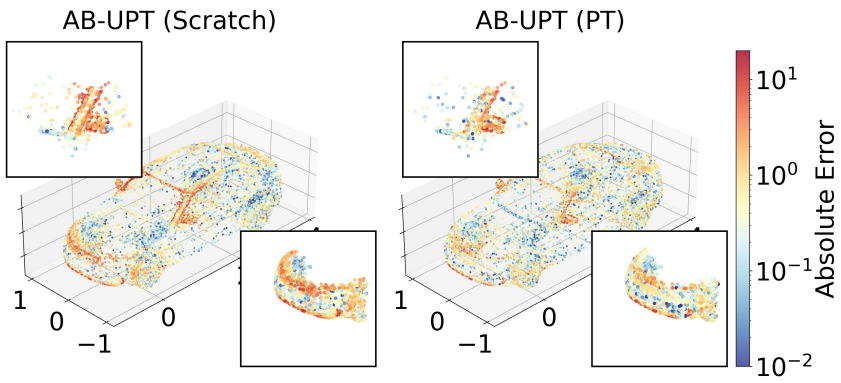


Figure 37: DrivAerML surface wall shear stress absolute error comparison. 32 training samples for AB-UPT.

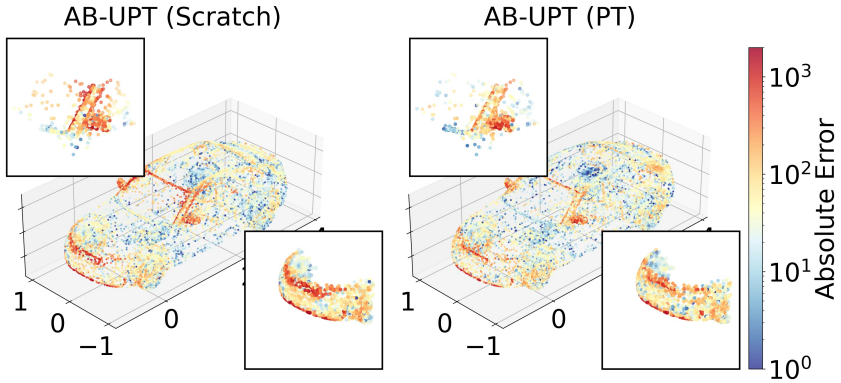


Figure 38: DrivAerML surface pressure absolute error comparison. 32 training samples for AB-UPT.

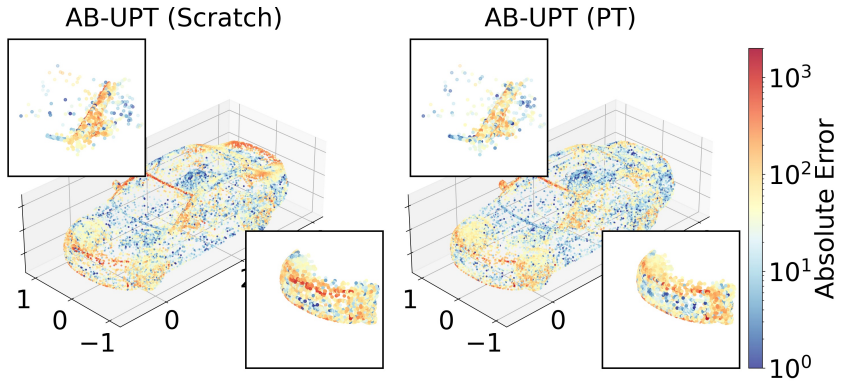


Figure 39: DrivAerML surface pressure absolute error comparison. 32 training samples for AB-UPT.

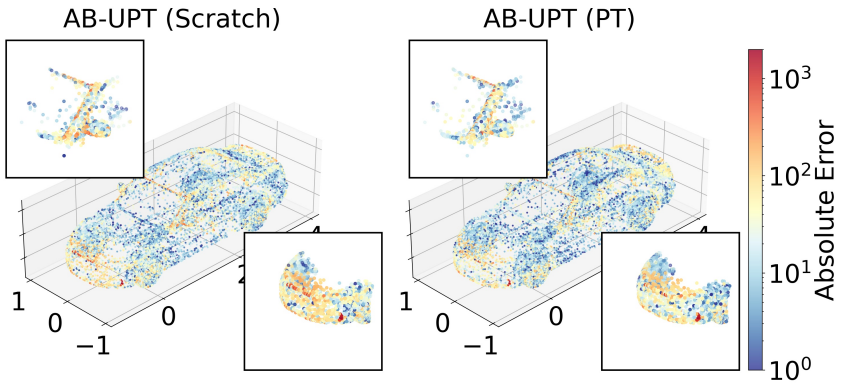


Figure 40: DrivAerML surface pressure absolute error comparison. 64 training samples for AB-UPT.

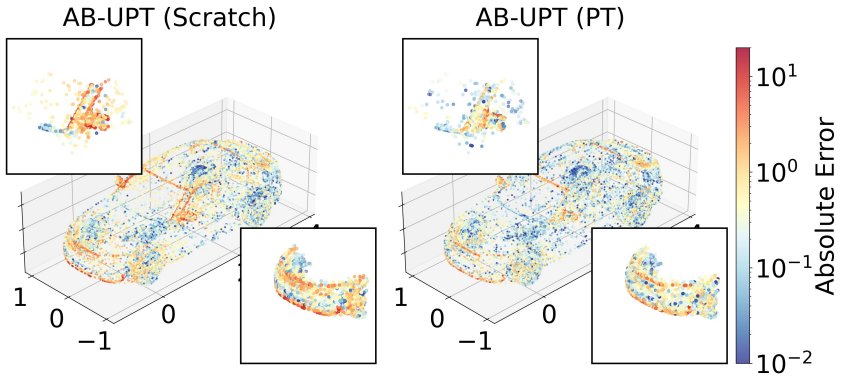


Figure 41: DrivAerML surface wall shear stress absolute error comparison. 64 training samples for AB-UPT.

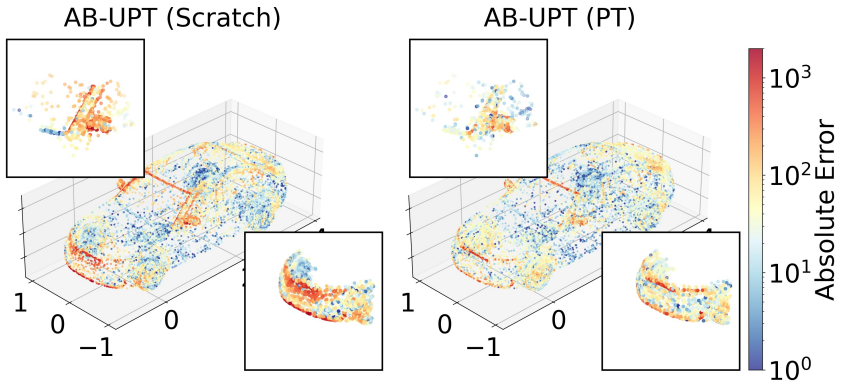


Figure 42: DrivAerML surface pressure absolute error comparison. 64 training samples for AB-UPT.

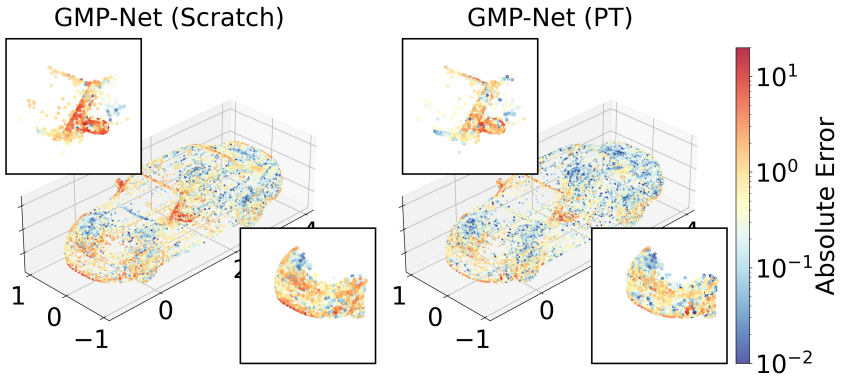


Figure 43: DrivAerML surface wall shear stress absolute error comparison. 16 training samples for GMP-Net.

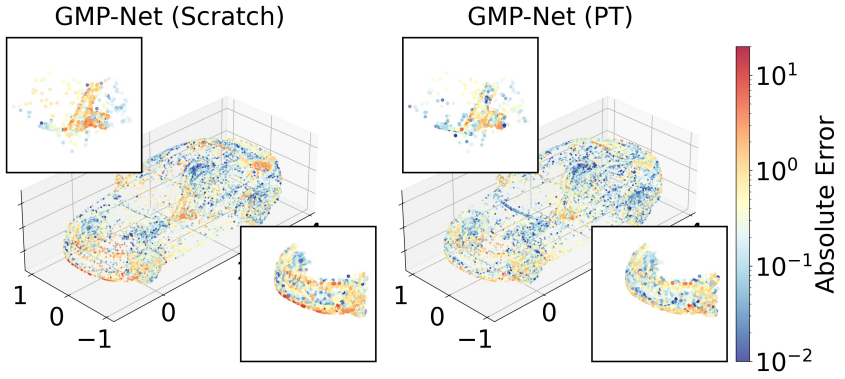


Figure 44: DrivAerML surface wall shear stress absolute error comparison. 32 training samples for GMP-Net.

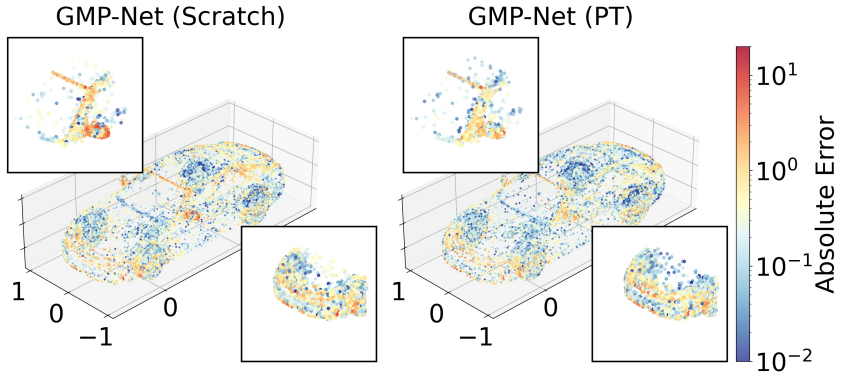


Figure 45: DrivAerML surface wall shear stress absolute error comparison. 32 training samples for GMP-Net.

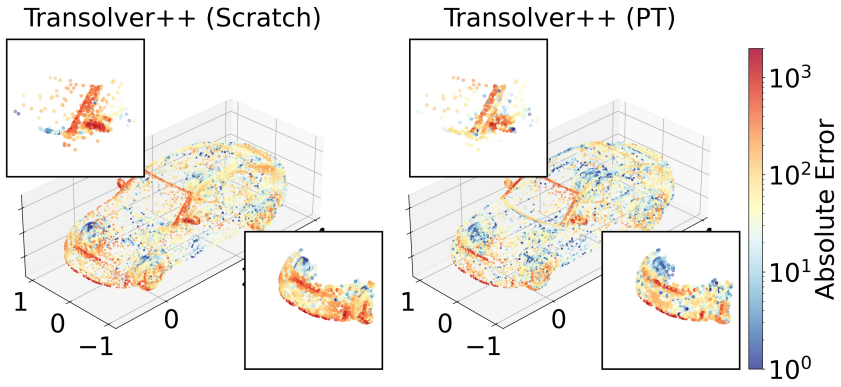


Figure 46: DrivAerML surface pressure absolute error comparison. 16 training samples for Transolver++.

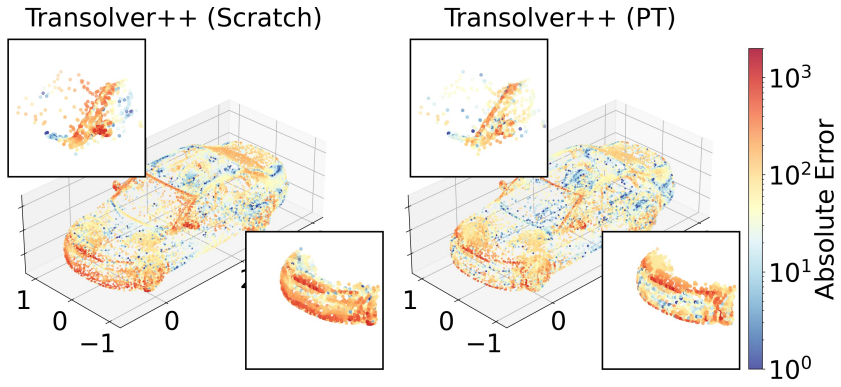


Figure 47: DrivAerML surface pressure absolute error comparison. 16 training samples for Transolver++.

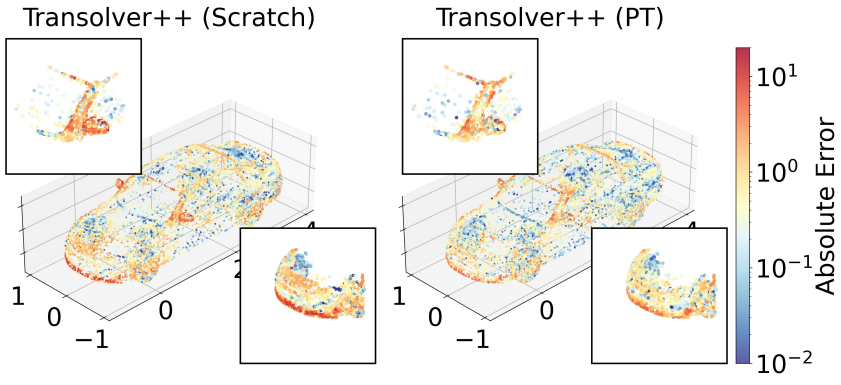


Figure 48: DrivAerML surface wall shear stress absolute error comparison. 32 training samples for Transolver++.

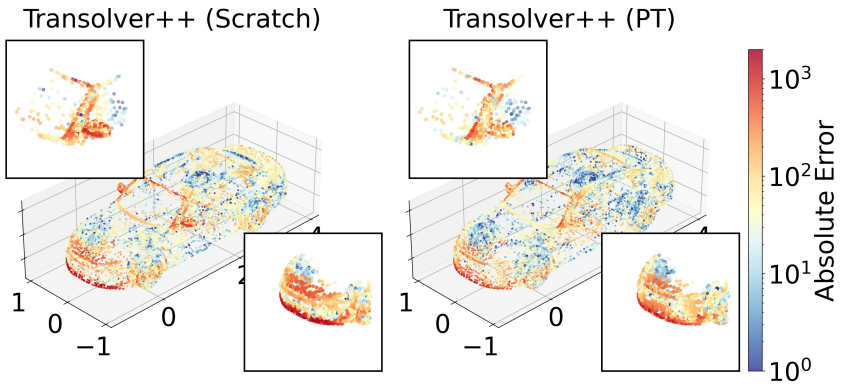


Figure 49: DrivAerML surface pressure absolute error comparison. 32 training samples for Transolver++.

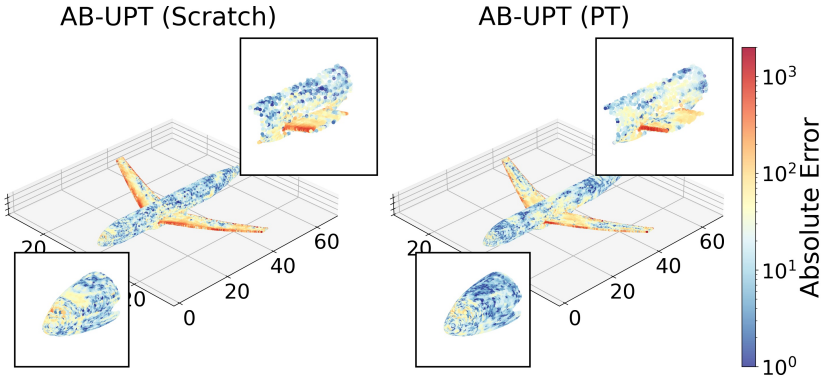


Figure 50: SHIFT-Wing surface pressure absolute error comparison. 128 training samples for AB-UPT.

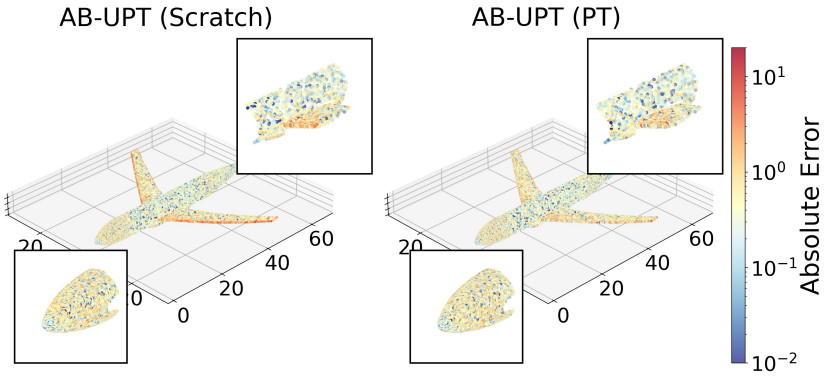


Figure 51: SHIFT-Wing surface wall shear stress absolute error comparison. 128 training samples for AB-UPT.

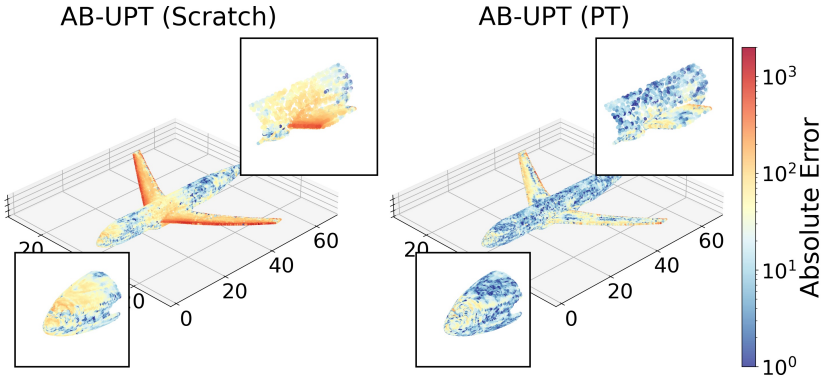


Figure 52: SHIFT-Wing surface pressure absolute error comparison. 128 training samples for AB-UPT.

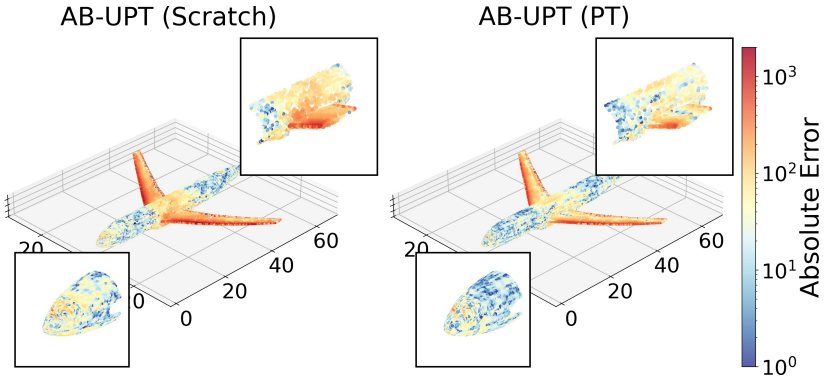


Figure 53: SHIFT-Wing surface pressure absolute error comparison. 64 training samples for AB-UPT.

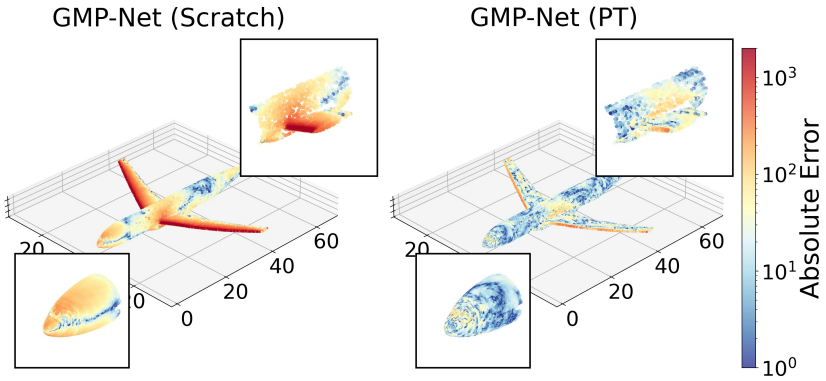


Figure 54: SHIFT-Wing surface pressure absolute error comparison. 128 training samples for GMP-Net.

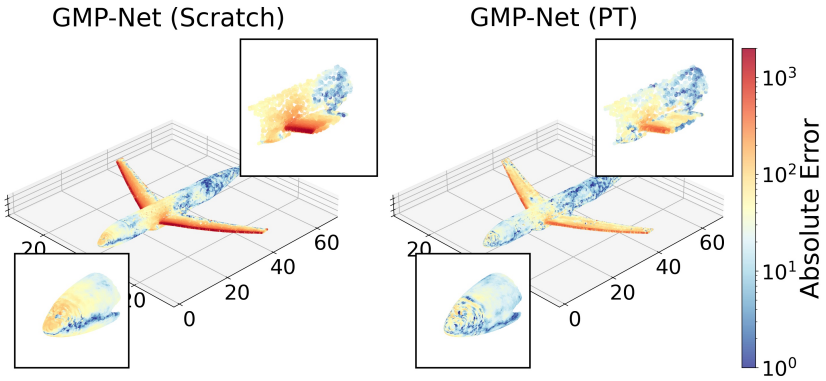


Figure 55: SHIFT-Wing surface pressure absolute error comparison. 128 training samples for GMP-Net.

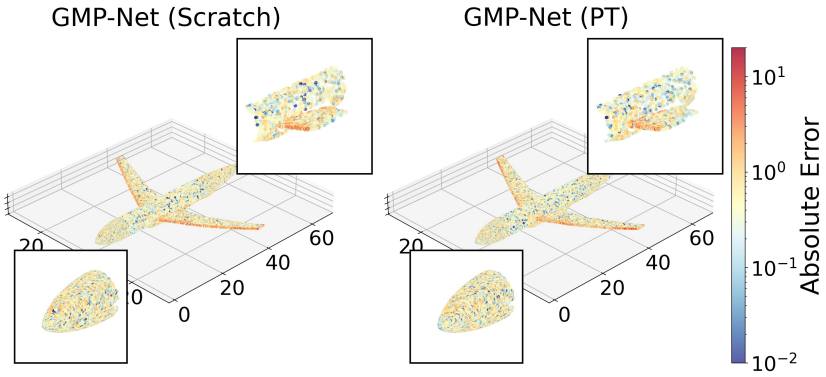


Figure 56: SHIFT-Wing surface wall shear stress absolute error comparison. 16 training samples for GMP-Net.

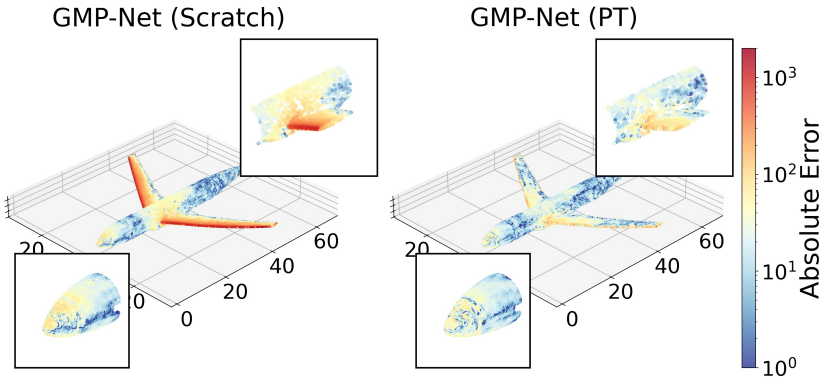


Figure 57: SHIFT-Wing surface pressure absolute error comparison. 64 training samples for GMP-Net.

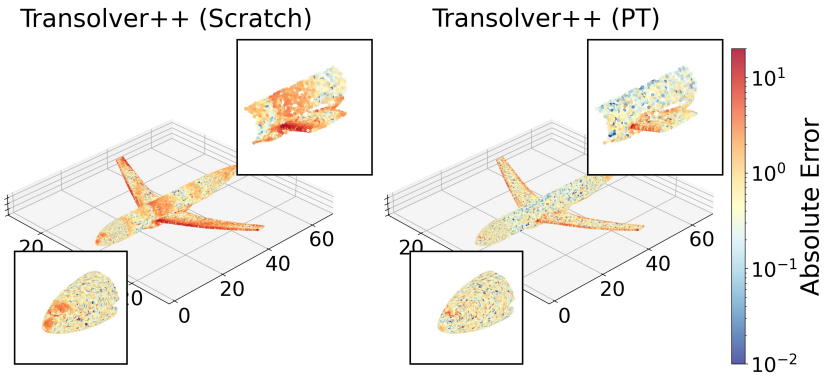


Figure 58: SHIFT-Wing surface wall shear stress absolute error comparison. 16 training samples for Transolver++.

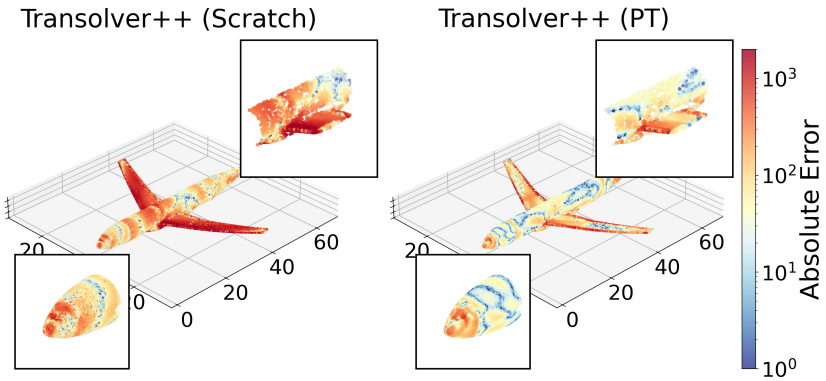


Figure 59: SHIFT-Wing surface pressure absolute error comparison. 16 training samples for Transolver++.

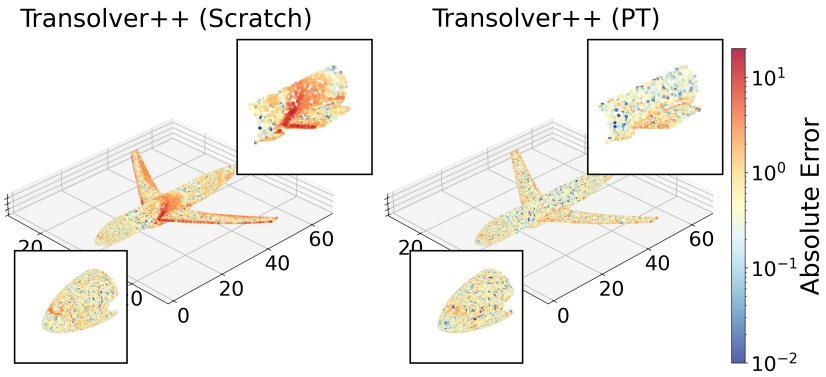


Figure 60: SHIFT-Wing surface wall shear stress absolute error comparison. 32 training samples for Transolver++.

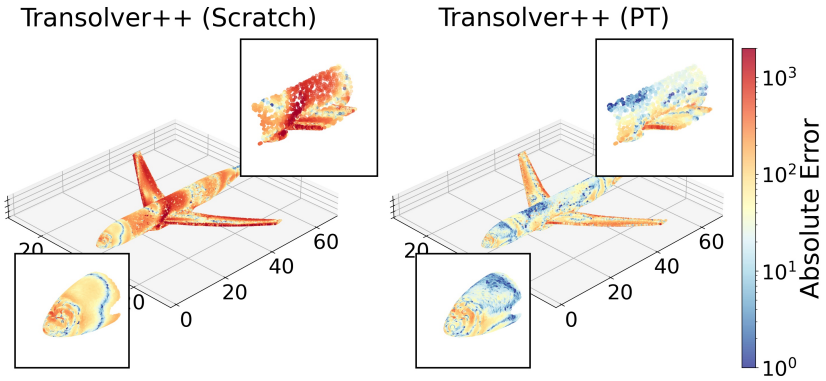


Figure 61: SHIFT-Wing surfacepressure absolute error comparison. 32 training samples for Transolver++.

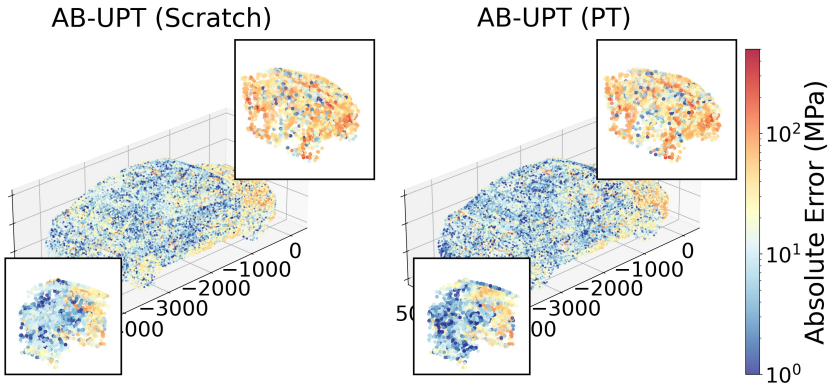


Figure 62: SHIFT-Crash Von Mises stress absolute error comparison. 16 training samples for AB-UPT.

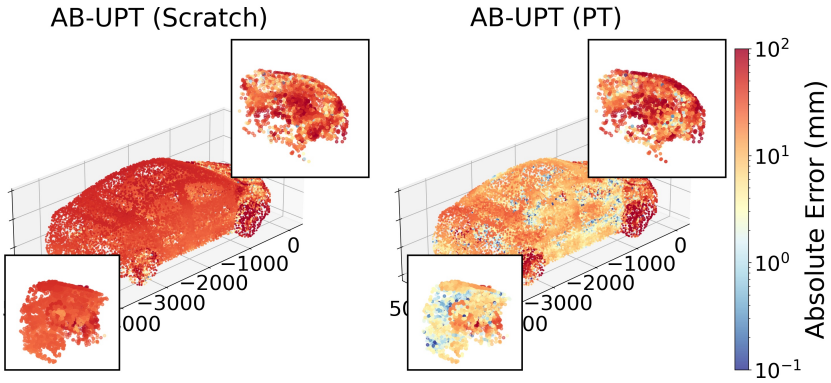


Figure 63: SHIFT-Crash dissplacement absolute error comparison. 16 training samples for AB-UPT.

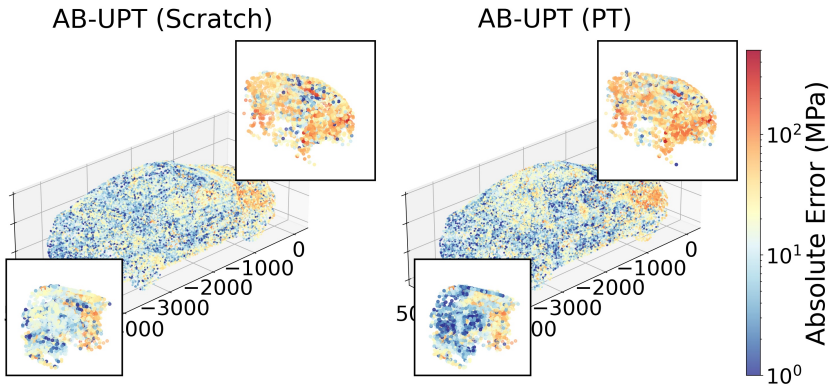


Figure 64: SHIFT-Crash Von Mises stress absolute error comparison. 16 training samples for AB-UPT.

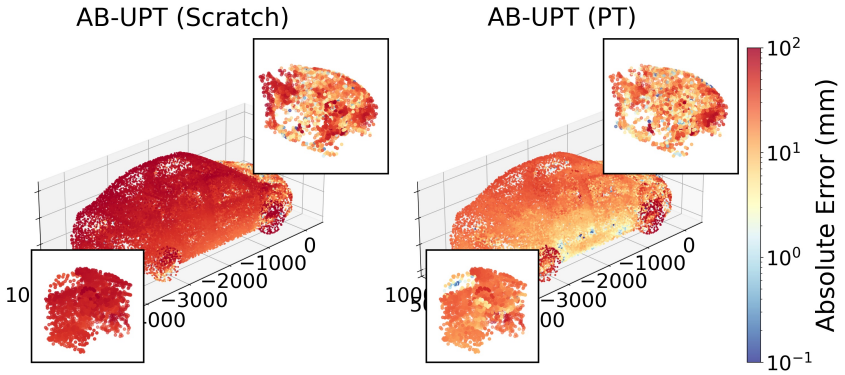


Figure 65: SHIFT-Crash displacement absolute error comparison. 64 training samples for AB-UPT.

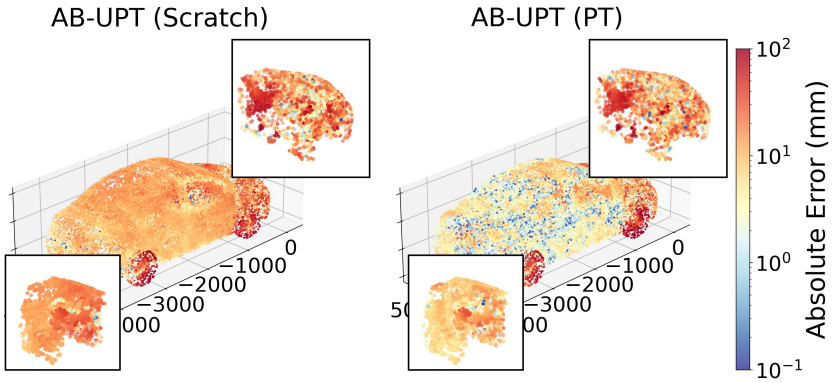


Figure 66: SHIFT-Crash displacement absolute error comparison. 64 training samples for AB-UPT.

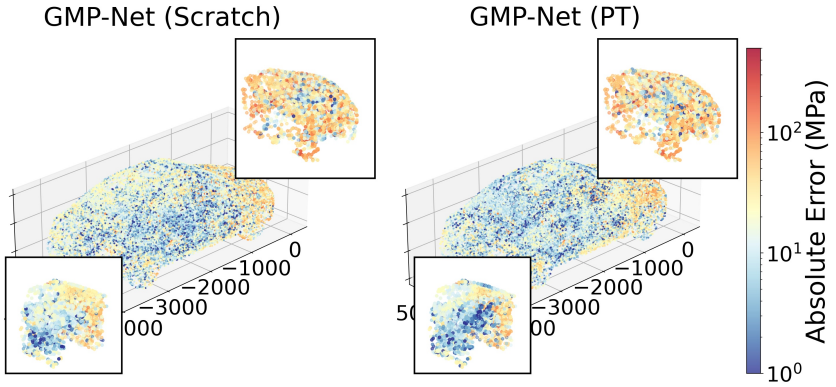


Figure 67: SHIFT-Crash Von Mises stress absolute error comparison. 16 training samples for GMP-Net.

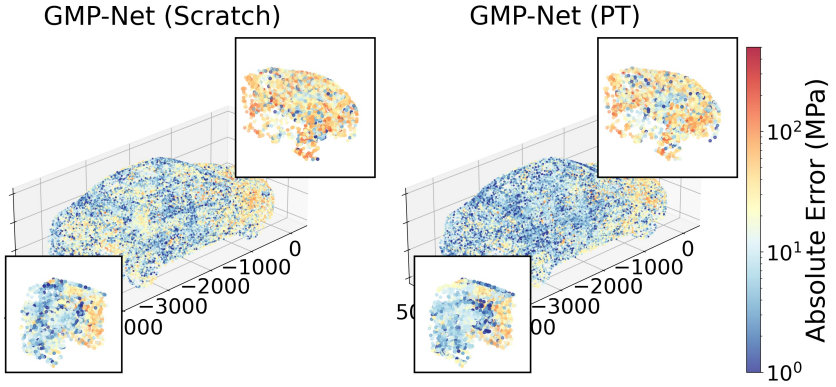


Figure 68: SHIFT-Crash Von Mises stress absolute error comparison. 32 training samples for GMP-Net.

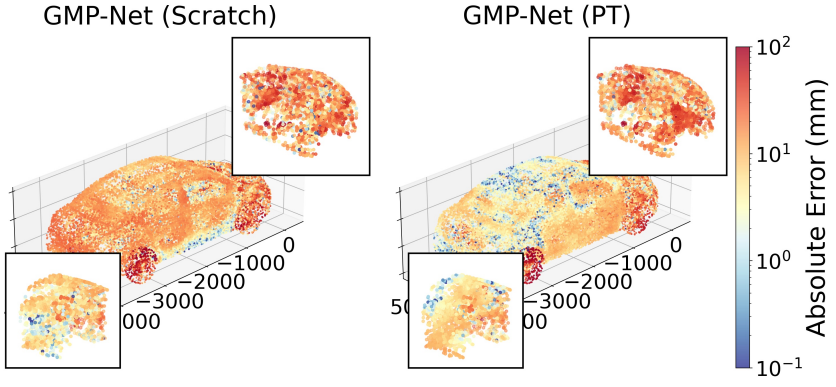


Figure 69: SHIFT-Crash displacement absolute error comparison. 32 training samples for GMP-Net.

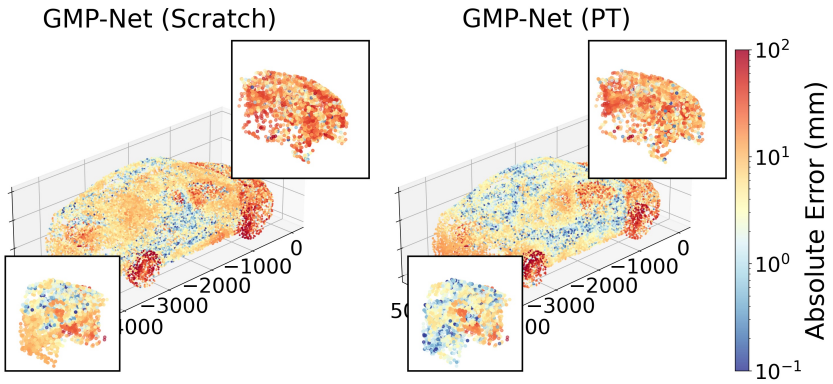


Figure 70: SHIFT-Crash displacement absolute error comparison. 64 training samples for GMP-Net.

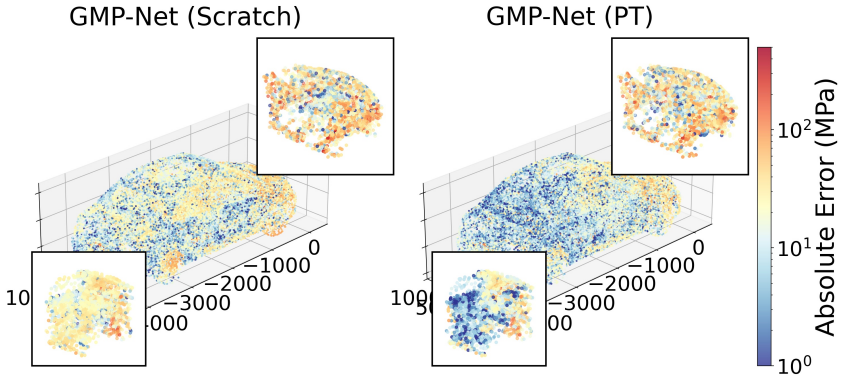


Figure 71: SHIFT-Crash Von Mises stress absolute error comparison. 64 training samples for GMP-Net.

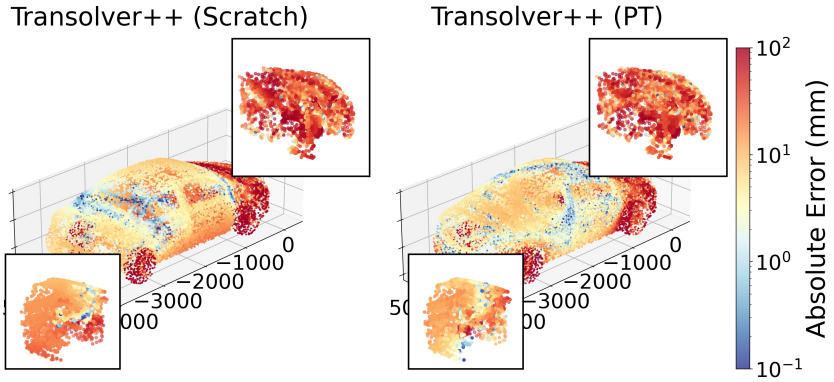


Figure 72: SHIFT-Crash displacement absolute error comparison. 16 training samples for Transolver++.

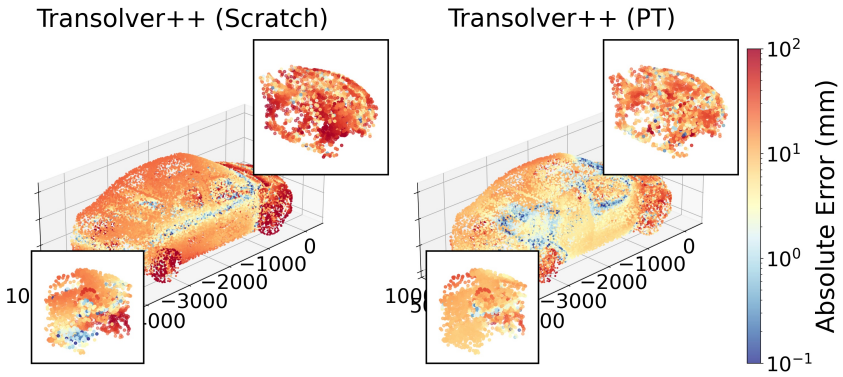


Figure 73: SHIFT-Crash displacement absolute error comparison. 64 training samples for Transolver++.

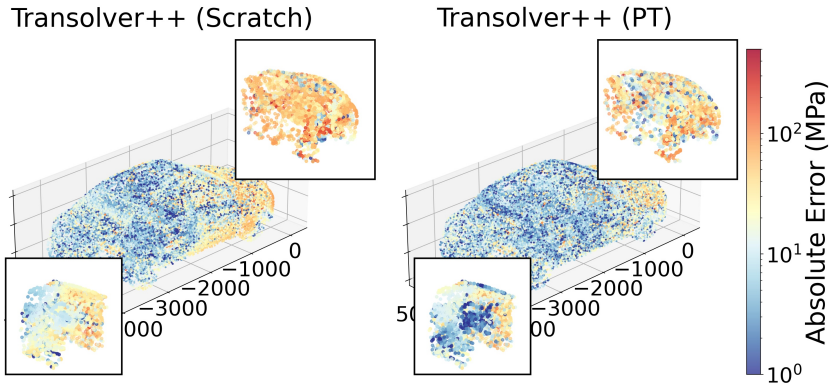


Figure 74: SHIFT-Crash Von Mises stress absolute error comparison. 64 training samples for Transolver++.

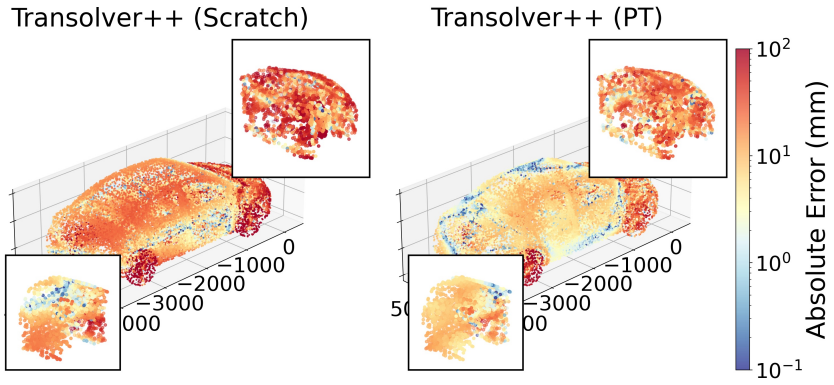


Figure 75: SHIFT-Crash displacement absolute error comparison. 64 training samples for Transolver++.

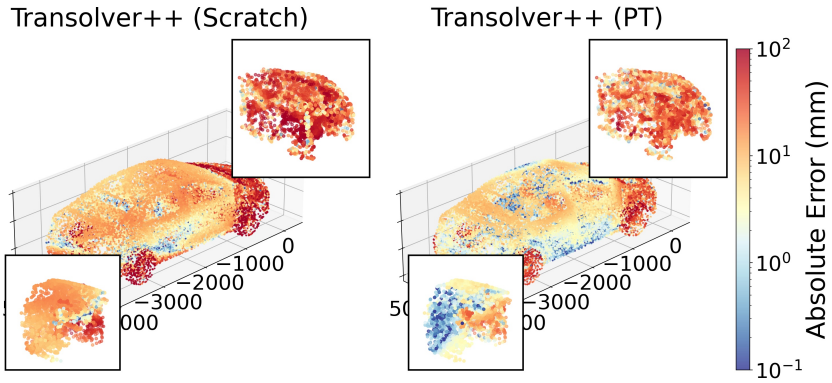


Figure 76: SHIFT-Crash displacement absolute error comparison. 64 training samples for Transolver++.

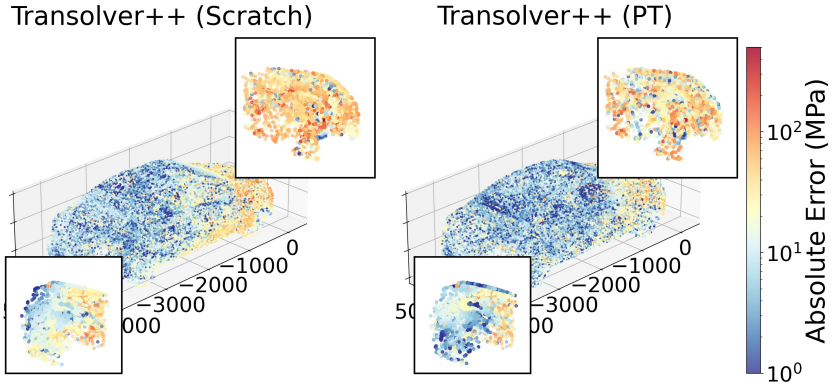


Figure 77: SHIFT-Crash Von Mises stress absolute error comparison. 64 training samples for Transolver++.

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: Our main claims, specifically the development of two distinct pre-training strategies, for steady and transient physics problems, and their performance gains, are clearly stated in the abstract and introduction (Section 1). These claims are supported by empirical results in Section 4.

   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 explicitly discussed in Section 4 and 5.

   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: This work is purely experimental and does not present new theoretical results, theorems, or formal proofs. All mathematical formulations are used strictly to define the problem setup.

   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 detailed information on model configurations and training hyperparameters in the appendix. Furthermore, the code and pre-trained model weights are publicly available in GitHub and Hugging Face.

   Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The code and pre-trained model weights are publicly available in GitHub and Hugging Face.

   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: We describe the experimental setup in Sections 3. Besides, the Appendix contains exhaustive hyperparameter tables for all training runs, baseline configurations, and downstream benchmarks.

   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: [No]

   Justification: Following other works in this area, tables and plots in the main manuscript may lack some statistical properties, but these are extended in the appendix.

   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: Part of the pre-training compute resources are mentioned in Section 4, but most details are provided in the Appendix, containing utilized hardware, memory and throughput comparisons, and average training times.

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: Our research focuses on developing neural surrogates for engineering simulations and does not involve human subjects, personal data, or sensitive information. We have strictly adhered to the NeurIPS Code of Ethics, including maintaining anonymity. While our framework facilitates rapid design iteration, we note that safety-critical validations should still be performed using high-fidelity numerical solvers. Our work focuses on civilian aerospace and automotive applications to promote sustainable engineering practices. Furthermore, we believe the computational cost of pre-training constitutes a net positive impact, as the resulting speedups and data efficiencies significantly reduce the total energy required for downstream simulation tasks.

   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: [Yes]

    Justification: As stated in Section 5, we believe that a framework for accelerated and high-fidelity physical modeling facilitates the development of more efficient, but also safer and more reliable products. Negative impacts include the risk of hallucinations in safety-critical applications and the environmental cost of large-scale training.

    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: This work does not involve high-risk generative models (e.g., LLMs or image generators) or scraped datasets. The models and data are strictly related to scientific or engineering simulations and do not pose the risks for societal misuse described in the guidelines.

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: We cite the original papers for all datasets and baseline models. Public assets like DrivAerML are used under their respective licenses. The proprietary SHIFT datasets were used with explicit permission from the original creators (Luminary Cloud).

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: New assets (2D KS data, method implementation and pre-trained weights) are published or currently being finalized for public release.

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: This research does not involve crowdsourcing or research with human subjects. All experiments were conducted using simulated physical datasets.

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: This research does not involve human subjects, study participants, or crowdsourcing; therefore, IRB approval was not required.

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: The core methodology and original research components of this work were developed without the use of LLMs, which were utilized for writing and coding assistance.

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.