SEEING NEAR AND FAR: A NEURAL OPERATOR THAT EFFICIENTLY TOKENIZES AND DECODES ON INDUSTRIAL-SCALE CFD MESHES

Anonymous authors
Paper under double-blind review

ABSTRACT

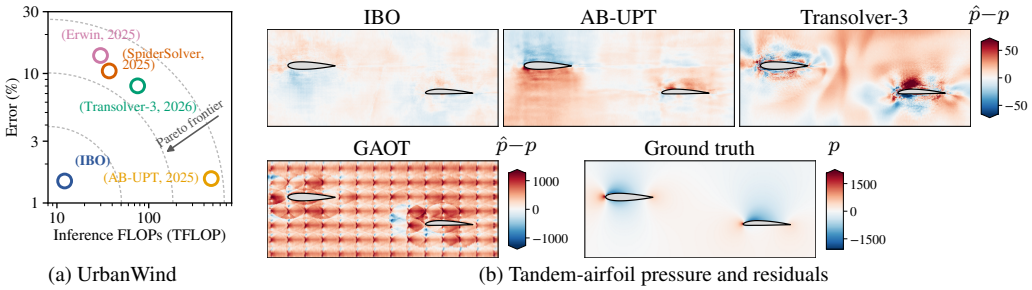
Industrial-scale computational fluid dynamics (CFD) simulations typically require more than a million mesh points to fully capture the complex non-linear physics. Consequently, recent efforts have revolved around designing neural operators which can handle such large, complex meshes without sacrificing speed or accuracy. Approaches which only aggregate local information during tokenization and decoding are highly compute-efficient, especially as meshes grow larger. However, they are currently outperformed on 3D industrial-scale benchmarks by models which retain global information through pointwise computation across the whole domain. We hypothesize this performance gap is partly attributable to a mismatch between latent-point allocation and spatially varying reconstruction demands, and propose the use of a KD-tree algorithm for more efficient allocation. To further generalize this, we propose an Importance-Balanced Operator (IBO), and show that the integration of an importance measure into the KD-tree algorithm to distribute the latent points closes this performance gap while retaining compute-efficiency. In our experiments, IBO achieves the best performance across multiple metrics on 2D single- and multi-airfoil aerodynamic problems. Crucially, on large-scale 3D datasets involving automotive aerodynamics (DrivAerML) and urban aerodynamics (a newly introduced UrbanWind dataset), IBO matches or outperforms state-of-the-art models across nearly all field metrics while requiring significantly less FLOPs per case ($\sim$5–40$\times$ less), thereby establishing a new accuracy-compute Pareto frontier for the learning of industrial-scale CFD.

1 INTRODUCTION

Scientific computing, including Computational Fluid Dynamics (CFD), is the cornerstone of modern day science and engineering, permitting virtual design optimization and what-if analysis before expensive real-world realization. However, current modelling needs are increasingly complex and time-consuming, requiring multi-CPU nodes or even supercomputers. For example, the size of a high-fidelity CFD mesh for modelling the flow over an airliner or automobile can vary from 10^7 mesh points (3D RANS) to 10^{12} mesh points (LES) (Spalart, 2000), with increasingly higher demands as the need for high-fidelity physics increases. DrivAerML, one of the benchmarks in this work, uses $\approx 1.4 \times 10^8$ points per case (Ashton et al., 2024). Meshes in real-world problems are also typically non-uniform, with denser requirements for regions like the boundary layers and wakes. While deep learning methods are touted as a means to accelerate simulations, they still face challenges at this scale, e.g., attention over every point is exorbitant at this scale due to its quadratic scaling.

To handle these large, non-uniform point sets, many neural operator models have been proposed, each broadly following an encode, process, decode paradigm. *Spatially localized models* such as GINO and GAOT (Li et al., 2023; Wen et al., 2025) follow a local–global–local design: they place latent points in physical space, aggregate only the local (nearby) information around each latent point into a token, before letting the tokens interact globally. Subsequently, they decode the field at query points based on their nearby tokens. In contrast, *globally-resolved* models such as Transolver-3 and AB-UPT (Zhou et al., 2026; Alkin et al., 2025) compute on the mesh points themselves, and encode and decode with information from the full domain. Spatially localized models are relatively cheap during inference because each output only requires a few nearby tokens, and have also been shown to be accurate on small problems, e.g. GAOT ranks at or near the top of low-dimensional PDE benchmarks (Wen et al., 2025). However, on larger CFD meshes of $\approx 10^8$ points, reproductions by

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



(a) UrbanWind

(b) Tandem-airfoil pressure and residuals

Figure 1: (a) Performance of different state-of-art-models on an UrbanWind dataset: geometric-mean relative L_2 error of the six fields against inference FLOPs per case; dashed curves are illustrative iso-utility contours. (b) Illustrative pressure for a tandem-airfoil case and error residuals across various neural operator models.

AB-UPT and Transolver-3 report GINO and GAOT to be an order of magnitude poorer (Wu et al., 2024; Alkin et al., 2025; Zhou et al., 2026).

We look into this gap with a *layout allocation analysis* of spatially localized models, and observe that allocating latent points in proportion to the flow's spatial physical complexity is the optimal layout at any given budget. Current strategies usually employ uniform distributions across the domain even though the flow's complexity can greatly differ, and such misallocation is a potential cause of performance degradation on industrial-scale problems.

Guided by this insight, we present the *Importance-Balanced Operator* (IBO), which seeks to realize this optimal latent point layout with a KD-tree partition integrated with an importance measure. Through this, latent points now concentrate where the geometry and flow are more complex. It can therefore learn more intricate geometry–flow relations. We evaluate IBO on four steady-flow benchmarks with 10^5 to 10^8 points per case, spanning 2D airfoil aerodynamics, 3D automobile aerodynamics and 3D urban aerodynamics. As many practical problems require simulation of multiple interacting objects in a single volume, e.g. in urban wind engineering, we deliberately chose to extend the paradigm of evaluation beyond the typical flow past a single geometry, e.g. in the case of airfoils or automobiles, into variable multi-body scenarios (the urban layouts have variable number of buildings per case with diverse spacings and building footprints). Across all these cases, IBO produces the best predictions, or is competitive, with recent state-of-the-art such as Transolver-3 and AB-UPT, while requiring less FLOPs in inference on the 3D benchmarks.

In summary, our main contributions are:

- **Unlike the flow-agnostic latent layouts of prior work**, we analyze the relation between latent point placement and spatial flow physics, and prove that they must be treated in tandem. Given importance priors such as a mesh, or knowledge of wall-distance, a more optimal latent point layout can be constructed which greatly reduces error.
- **We present the Importance-Balanced Operator (IBO)**, whereby the use of a KD-tree for the allocation of latent points in combination with an importance measure that estimates the flow's spatial complexity is shown to substantially improve the performance of spatially localized models.
- **On CFD problems beyond a million points**, IBO matches or exceeds the strongest models on every UrbanWind volume channel and on several DrivAerML metrics, while using an order of magnitude less inference FLOPs. This reduced inference cost can potentially compound for tasks such as design optimization, where numerous, rapid design iterations are required.
- **We release UrbanWind**, a benchmark of 1,920 steady RANS simulations over 480 real urban layouts, each with four inflow wind directions. This dataset comprises variable number of buildings per case, and provides another axis of evaluation in contrast to the majority of current datasets which focus on flow past a single object (e.g. airfoil or automobile).

2 Related Work

Neural operators and graph-based models. Neural operators (Kovachki et al., 2023) have garnered attention for their good predictive ability in physical systems, including two key variants, the DeepONet (Lu et al., 2021) and the Fourier Neural Operator (FNO) (Li et al., 2020), but these have not

been as suited to large-scale, irregular geometries. Given the prevalence of large, unstructured meshes in real-world applications, graph-based models were proposed, with methods such as MeshGraphNets showing great potential (Pfaff et al., 2020; Sanchez-Gonzalez et al., 2020; Brandstetter et al., 2022). However, as they require message passing directly on the simulation mesh, capturing long-range coupling on large meshes requires many hops or multiscale hierarchies (Fortunato et al., 2022; Nabian et al., 2024), and they still face challenges on large meshes (Zhou et al., 2026). In contrast, recent performant models have been applied to point sets directly, both to facilitate application to irregular geometries and to avoid the use of message-passing operations. These models typically compress the mesh points into a reduced set of tokens through an encode - process - decode pipeline.

Globally-resolved neural operators. Maintaining global knowledge of the entire physical system is ideal, but can be computationally intractable. Two state-of-the-art approaches, Transolver-3 and AB-UPT, introduced innovations to achieve strong industrial-scale accuracy while retaining global point-wise computation through their processing blocks. The Transolver models (Wu et al., 2024; Zhou et al., 2026) compress the points in a physics-feature space, but utilized sampled training and chunked accumulation of full-mesh slice statistics. SpiderSolver (Qi et al., 2025) likewise lets every point attend to a small set of domain-wide tokens, but forms them geometrically in spider-web-like sectors. In contrast, AB-UPT (Alkin et al., 2025) uses subsampled anchor tokens from the entire domain. However, full-field prediction for these models still require expensive processing per inference. The Erwin (Zhdanov et al., 2025) model has also been proposed, where empirically near-linear runtime has been obtained without sub-sampling or compression through the use of a hierarchical ball tree, but it still faces challenges with very heterogeneous point densities.

Spatially-localized neural operators. Other models employ a local encoding–global processing–local decoding structure, e.g. GINO (Li et al., 2023) has graph operators aggregate an irregular point cloud onto a regular latent grid, exchanges information globally via an FNO, before a local graph operator reconstructs the output. This provides good geometric flexibility with efficient global computation. GAOT (Wen et al., 2025) further used multiscale attentional graph operators and geometric embeddings with a ViT processor. However, current layout allocation and aggregation rules can incur high reconstruction errors or expensive neighborhood processing (Alkin et al., 2025; Zhou et al., 2026), and these methods generally have poorer predictive and scaling performance relative to Transolver-3 and AB-UPT on industrial-scale point sets.

Importance and latent placement. While the above methods try to circumvent the cost of operations through subsampling, or multi-scale encoding (e.g. Erwin or GAOT), others have reported that specific allocations of the latent tokens can improve model performance (Liu et al., 2023; Hagnberger et al., 2025). AB-UPT also reported that an advantage of sampling anchor positions from the simulation mesh is its implicit assignment to more highly resolved (and important) regions (Alkin et al., 2025). Hence, we investigate the treatment of point aggregation and latent token placement in the physical domain and define an importance measure through which we can better define an optimal latent point allocation strategy in this work.

3 Preliminaries: Spatially Localized Models

A CFD case consists of mesh points $\mathcal{X} = \{x_i\}_{i=1}^{N} \subset \mathbb{R}^d$, with $d \in \{2, 3\}$, a geometric descriptor field $\xi(x)$ sampled as $\xi_i = \xi(x_i)$, and an optional case condition μ, such as the inflow. A spatially localized model represents the case using $L \ll N$ latent points. Their positions form the *latent layout* $\mathcal{A} = \{a_j\}_{j=1}^{L}$, and each point carries a token of width D. The model consists of local encoding, global processing and local decoding.

Local encoding. The encoder summarizes the input in a region C_j around each latent point:

$$h_j = f_{\text{enc}}\big(\{(x_i, \xi_i) : x_i \in C_j\},\, a_j,\, \mu\big), \qquad h_j \in \mathbb{R}^D. \tag{1}$$

These regions may overlap. GINO and GAOT use regular latent grids with fixed-radius encoding neighbourhoods (Li et al., 2023; Wen et al., 2025), while PiT uses neighbourhoods of a coarse subset of the input mesh (Chen & Wu, 2024).

Global processing. The processor exchanges information among the latent tokens:

$$Z = f_{\text{proc}}(H, \mathcal{A}, \mu), \qquad H = [h_1; \ldots; h_L], \quad Z = [z_1; \ldots; z_L] \in \mathbb{R}^{L \times D}. \tag{2}$$

The processed token z_j remains associated with position a_j.

Local decoding. At a query position x, the decoder reconstructs the physical field from nearby latent points and the query's geometric descriptors:

$$\hat{u}(x) = f_{\mathrm{dec}}\big(x,\, \xi(x),\, \{(a_j, h_j, z_j)\}_{j\in\mathcal{N}(x)},\, \mu\big), \tag{3}$$

where $\mathcal{N}(x)$ indexes the latent points available to the query, selected, e.g., through a ball query or nearest-neighbour search. Evaluating this conditional field at the requested positions reconstructs the full output.

4 Importance and Resolution Allocation

In the models of Section 3, latent positions determine the spatial scales over which information is compressed and reconstructed. Thin boundary layers, wakes and complex boundaries can require finer spatial resolution than slowly varying regions. With the architecture, token width and training setup fixed, we ask: *how should a budget of L latent points be distributed to reduce error?*

4.1 From local physical complexity to resolution error

Model setup and error decomposition. Let $\Omega \subset \mathbb{R}^d$ be a bounded volume domain, and let u be the target physical field for the geometric descriptor ξ. We fix the case condition and suppress it in the notation. We describe the resolution allocation of a discrete layout $\mathcal{A}$ by a positive continuum point density $\lambda(x)$ satisfying $\int_\Omega \lambda(x)\,dx = L$. Under common local coverage conditions, the representative spatial scale varies as $\lambda(x)^{-1/d}$. Denser latent points reduce the spatial extent that each local representation must cover.

Let $q(x) > 0$ be the normalized evaluation density, with $\int_\Omega q\,dx = 1$. For any square-integrable field v, define $\|v\|_q := \big(\int_\Omega \|v(x)\|_2^2\, q(x)\,dx\big)^{1/2}$, where $\|\cdot\|_2$ is the Euclidean norm of the field components at a point. Following the distinction between encoding, approximation and reconstruction errors in operator learning (Bhattacharya et al., 2021; Lanthaler et al., 2022), we introduce reference encoding and reconstruction maps for a given layout and token capacity. These maps serve as oracle benchmarks. Let $\widetilde{\xi}_{\mathcal{A}}$ denote the geometric context recovered from the information retained by the reference encoder. The reference reconstruction $\mathcal{R}_{\mathcal{A}}[\cdot]$ maps this context to an output field and is defined at both ξ and $\widetilde{\xi}_{\mathcal{A}}$. In both evaluations, the geometric descriptors supplied directly at query locations are held fixed at their original values; the argument varies only the geometric context passed through the encoding path. For the actual prediction $\widehat{u}_{\theta,\mathcal{A}}$, with learned parameters θ, define

$$\varepsilon_{\mathrm{enc}} := \Big\|\mathcal{R}_{\mathcal{A}}[\xi] - \mathcal{R}_{\mathcal{A}}[\widetilde{\xi}_{\mathcal{A}}]\Big\|_q, \quad \varepsilon_{\mathrm{dec}} := \big\|u - \mathcal{R}_{\mathcal{A}}[\xi]\big\|_q, \quad \varepsilon_{\mathrm{learn}} := \Big\|\mathcal{R}_{\mathcal{A}}[\widetilde{\xi}_{\mathcal{A}}] - \widehat{u}_{\theta,\mathcal{A}}\Big\|_q. \tag{4}$$

These terms measure the effect of information lost during encoding, the reconstruction error at finite latent resolution, and the learned model's deviation from the reference prediction. The triangle inequality gives

$$\big\|u - \widehat{u}_{\theta,\mathcal{A}}\big\|_q \le \varepsilon_{\mathrm{enc}} + \varepsilon_{\mathrm{dec}} + \varepsilon_{\mathrm{learn}}. \tag{5}$$

We focus on the decoder resolution error $\varepsilon_{\mathrm{dec}}$ below. Appendix B gives the corresponding encoder model and a joint allocation analysis. The learning error can also depend on the layout; our analysis focuses on the explicit resolution component.

Local approximation class. We assume that the reference reconstruction class has an order-r_{dec} local approximation property: a positive difficulty function $\kappa_{\mathrm{dec}}(x)$ therefore satisfies

$$\big\|u(x) - \mathcal{R}_{\mathcal{A}}[\xi](x)\big\|_2 \le \left(\frac{\kappa_{\mathrm{dec}}(x)}{\lambda(x)}\right)^{r_{\mathrm{dec}}/d}. \tag{6}$$

This expresses order-r_{dec} decay with spatial scale in terms of point density. The difficulty is parameterized on the density scale: at a fixed resolution, larger κ_{dec} gives a larger local error bound and requires more latent points to maintain the same accuracy. It reflects the local representation demands of the physical field, e.g. due to rapid flow variation induced by the geometry.

Squaring and integrating gives the resolution error bound

$$\varepsilon_{\mathrm{dec}}^2 \le \bar{\varepsilon}_{\mathrm{dec}}^2[\lambda] := \int_\Omega q(x) \left(\frac{\kappa_{\mathrm{dec}}(x)}{\lambda(x)}\right)^{\beta_{\mathrm{dec}}} dx, \qquad \beta_{\mathrm{dec}} := \frac{2r_{\mathrm{dec}}}{d} > 0. \tag{7}$$

This functional combines physical complexity, allocated resolution and evaluation weight.

4.2 FROM ENGINEERING PRIORS TO IMPORTANCE ALLOCATION

Importance as a physical prior. Optimizing Eq. (7) directly requires the unknown difficulty κ_{dec}. We propose to estimate κ through an importance density ρ, with $\int_\Omega \rho\, dx = 1$. The measure $\rho(x)\, dx$ assigns importance mass to each region based on physical priors.

Assumption 4.1 (Importance matches local resolution demand)**.** There is a positive constant c_{dec}, independent of position and layout, such that

$$\kappa_{\mathrm{dec}}(x) = c_{\mathrm{dec}}\rho(x). \tag{8}$$

The matching relation assumes that the importance measure reflects the field's local approximation difficulty. This is also true for a well-constructed CFD mesh, as regions that are harder to resolve receive more points to balance a relevant local error indicator (Huang & Russell, 2011, Ch. 2). When an adaptive CFD mesh exists, we can use its normalized mesh node density as ρ. Typically, the query points one would use are the exact same mesh nodes. Hence, the evaluation density can be assumed to be the same normalized node density used for importance: $q = \rho$.

Proposition 4.2 (Optimal importance allocation)**.** *Under the evaluation setup $q = \rho$ and Assumption 4.1, the density minimizing Eq.* (7) *subject to $\int_\Omega \lambda\, dx = L$ is*

$$\lambda^\star(x) = L\rho(x). \tag{9}$$

Appendix B.2 proves this result together with the joint encoder–decoder case. Each region receives latent points in proportion to its importance mass. When the prior captures local resolution demand, complex regions naturally receive finer representations.

Cost of uniform allocation. For $\lambda_{\mathrm{unif}} = L/|\Omega|$, the ratio between the uniform and optimal error bounds is

$$\frac{\overline{\varepsilon}^2_{\mathrm{dec}}[\lambda_{\mathrm{unif}}]}{\overline{\varepsilon}^2_{\mathrm{dec}}[\lambda^\star]} = |\Omega|^{\beta_{\mathrm{dec}}} \int_\Omega \rho(x)^{1+\beta_{\mathrm{dec}}}\, dx \geq 1, \tag{10}$$

with equality only for uniform importance. A uniform layout therefore leaves difficult regions under-resolved while over-allocating latent points where less resolution is needed.

Second-order stability. Let $\lambda^\star$ be a positive minimizer of Eq. (7). For a density λ with the same budget, define $\eta := \|\lambda/\lambda^\star - 1\|_\infty$, the largest relative density deviation, up to sets of measure zero. If $\eta \leq 1/2$, then

$$0 \leq \frac{\overline{\varepsilon}^2_{\mathrm{dec}}[\lambda]}{\overline{\varepsilon}^2_{\mathrm{dec}}[\lambda^\star]} - 1 \leq C\eta^2. \tag{11}$$

Here C depends only on β_{dec}. The uniform relative bound keeps $\lambda(x) \geq \lambda^\star(x)/2$ almost everywhere. Thus a prior that gives a density sufficiently close to the optimum incurs only a quadratic relative allocation cost. This implies an approximately correct importance measure can still be beneficial; Appendix B.3 gives the proof.

Equal-importance allocation with a KD-tree. A KD-tree (Bentley, 1975) provides a direct realization of the allocation principle. We recursively partition the domain by importance mass and place one latent point in each leaf, targeting mass $1/L$ per leaf. For an ideal equal-mass partition,

$$\int_{C_j} \rho(x)\, dx = \frac{1}{L} \quad \Longrightarrow \quad \lambda_L(x) := \frac{1}{|C_j|} = L\overline{\rho}_j, \qquad x \in C_j, \tag{12}$$

where $\overline{\rho}_j$ is the mean importance density within the leaf. The partition therefore realizes proportional allocation at the scale of leaf averages: important regions receive more, smaller leaves and hence denser latent points. This transfers the physical prior embodied in mesh design to the latent layout. Section 5 and Appendix C.1 specify the construction.

5 METHOD: IMPORTANCE-BALANCED OPERATOR

The Importance-Balanced Operator (IBO) uses the encode–process–decode architecture of Section 3, with an importance-balanced layout derived from the allocation principle of Section 4 (Figure 2): latent points are distributed according to the anticipated physics.

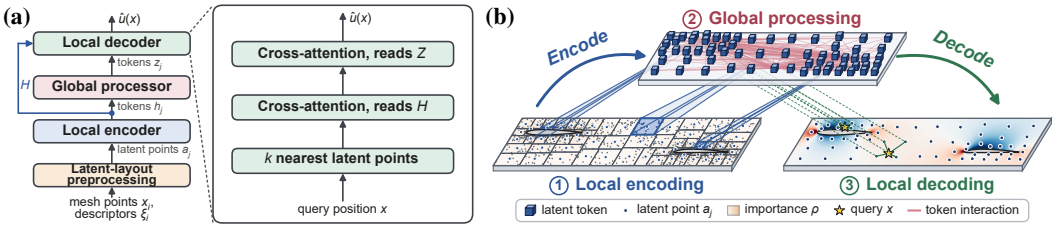


Figure 2: (a) Schematic of IBO architecture. (b) Leaves of equal importance mass define tokens at positions a_j (1), which interact globally (2); each query is decoded from its k nearest tokens (3).

Latent-layout preprocessing. For each new case, IBO constructs a KD-tree with one latent point per leaf using physical weights w_i at a_j, the weighted centroid of its leaf. When a locally refined mesh is available, equal point weights inherit its resolution prior. On a regular mesh, a wall-distance weight can be applied to supply spatially varying importance. This preprocessing has no learned parameters and runs only once per case; the partition and neighbour indices can be cached and reused. Appendices C.1 and D.3 give the split rules, member selection and weights.

Local encoder. Within each leaf, a shared PointNet (Qi et al., 2017) maps the relative positions and geometric features of selected members or subregion representatives to point features. Their channelwise maximum, mean and standard deviation are concatenated with a region descriptor and projected to h_j, realizing Eq. (1). Member caps or subregion representatives control the encoding cost. Appendices C.2 and D.3 specify the encoder and features.

Global processor. A pre-norm Transformer (Vaswani et al., 2017; Dosovitskiy et al., 2021) exchanges information among the latent tokens, using rotary position embeddings (Su et al., 2024). A small set of learned registers (Darcet et al., 2024) and long skip connections can be added. The spatial outputs are the tokens z_j in Eq. (2); Appendices C.3 and D.3 give the configurations.

Local decoder. For each query, $\mathcal{N}(x)$ contains its k nearest latent points. Local cross-attention combines token content with a distance prior to reconstruct the field. A fixed neighbour count is used to control the effective local extant. Architecture details, e.g. output heads, are in Appendix C.4.

6 Experiments

We evaluate IBO on four steady-flow benchmarks ordered by geometric complexity: two single-body cases, AirfRANS and DrivAerML, and two multi-body cases, TandemFoilSet and UrbanWind, the last introduced in this work (Table 1). The AirfRANS and DrivAerML comparisons primarily draw from published results, with AB-UPT on AirfRANS reproduced by us, while we reproduce every baseline on TandemFoilSet and UrbanWind under shared data splits, query sampling, loss definitions and training budgets, retaining model-specific architectures and optimization settings (Appendix D).

The baselines are GINO (Li et al., 2023), GAOT (Wen et al., 2025), Transolver (Wu et al., 2024), Transolver-3 (Zhou et al., 2026), AB-UPT (Alkin et al., 2025), SpiderSolver (Qi et al., 2025) and Erwin (Zhdanov et al., 2025), spanning multiple state-of-the-art globally-resolved and spatially localized neural operators. In all tables, lower is better for errors and costs and higher for correlations and R^2; the best or tied-best value per column is in bold.

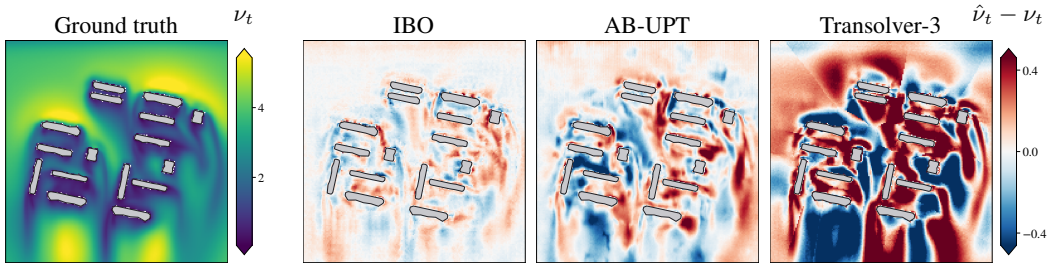
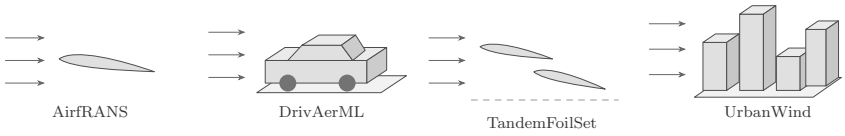


Figure 3: Turbulent viscosity and prediction residuals in an UrbanWind case.

Table 1: Benchmark scope. The four geometries are sketched below the table.

	Single body		Multiple bodies	
	AirfRANS	DrivAerML	TandemFoilSet	UrbanWind
Flow	2D steady RANS	3D hybrid RANS–LES	2D steady flow, laminar and RANS	3D steady RANS
Geometry	One airfoil	One vehicle	Two interacting airfoils	1–124 buildings
Mesh	Locally refined	Locally refined	Locally refined, overset	Cartesian with wall snapping
Points/case	$\sim 10^5$	$\sim 1.4 \times 10^8$	1.3–3.5×10^5	$\sim 3 \times 10^6$
Train/val./test	720/80/200	400/34/50	8:1:1 per subset	1,600/160/160
Collection	1,000 cases	500 cases	Five subsets; 4,141 retained cases	1,920 cases; 480 estates
Volume targets	u_x, u_y, p, ν_t	U, p	u, v, p	U_x, U_y, U_z, p, ν_t
Surface targets	p_s	p_s, τ	p_s	p_s



6.1 Prediction accuracy

AirfRANS. IBO attains the best value in every column of Table 2. Relative to GINO, the earliest spatially localized model, the volume, surface and lift errors fall by $32\times$, $78\times$ and $12\times$, which removes GINO's order-of-magnitude gap to globally-resolved models. Even against AB-UPT, the strongest baseline on the surface metrics, surface-pressure MSE is lower by 44% and the lift error integrated from it is improved by 69%. This disparity in performance is potentially as IBO better allocates latent points to the thin turbulent boundary layer crucial in the AirfRANS dataset.

Table 2: AirfRANS: standardized field MSE, lift relative error, and lift Spearman correlation. † Our reproduction; ‡ reproduced by Wu et al. (2024); the Transolver and SpiderSolver rows are the values reported by Wu et al. (2024) and Qi et al. (2025).

Model	Volume ↓	Surface ↓	C_L ↓	ρ_L ↑
GINO‡	0.0297	0.0482	0.1821	0.9958
Transolver	0.0037	0.0142	0.1030	0.9978
SpiderSolver	0.0017	0.0043	0.0741	0.9988
AB-UPT†	0.0022	0.0011	0.0470	0.9996
IBO (ours)	**0.0009**	**0.0006**	**0.0146**	**0.9999**

DrivAerML. IBO has the lowest volume-pressure error of all models but is second to Transolver-3 on the other five metrics (Table 3), with errors three to seventeen times lower than those of GINO and GAOT. To our knowledge, IBO is the first spatially localized model to reach the accuracy of the strongest models at this scale ($\approx 1.4 \times 10^8$ points per case). Crucially, we also realize the key benefit of reduced computational cost over the globally-resolved models, Transolver-3 and AB-UPT, requiring an order of magnitude lower inference FLOPs (Section 6.2).

Table 3: DrivAerML: relative L_2 field error (%), coefficient R^2, and analytic full-mesh inference cost (PFLOP per case). ‡ GINO as reproduced by Alkin et al. (2025) and GAOT as reproduced by Zhou et al. (2026); the Transolver-3 and AB-UPT rows are the values reported by Zhou et al. (2026) and Alkin et al. (2025). Evaluation protocols are in Appendix D.2.

	Surface ↓		Volume ↓		R^2 ↑		PFLOP
Model	p_s	τ	p	U	C_D	C_L	↓
GINO‡	13.03	21.71	44.90	40.58	–	–	–
GAOT‡	34.00	61.00	56.90	57.18	–	–	–
Transolver-3	**3.71**	**5.85**	5.72	**4.14**	**0.972**	**0.985**	25.3
AB-UPT	3.82	7.29	6.08	5.93	0.963	0.975	22.2
IBO (ours)	3.78	7.12	**3.44**	5.14	0.967	0.980	**2.6**

TandemFoilSet and UrbanWind. TandemFoilSet places two interacting airfoils in one domain, so the wake of the front body meets the boundary layer of the rear one; we reproduce all five baselines under one protocol (Table 5). UrbanWind is a benchmark of high-fidelity steady RANS simulations over 480 urban layouts (Table 4). Unlike the preceding benchmarks, the layouts contain up to 124 buildings of arbitrary footprint, height and placement. This benchmark therefore further evaluates the models' suitability to complex multi-body geometries.

IBO is best or tied-best on every field metric of Tables 4 and 5 except UrbanWind's surface pressure, where AB-UPT is better. Interestingly, Transolver-3 trails by an order of magnitude on the volume

Table 5: TandemFoilSet: standardized volume and surface MSE on all five subsets. All baselines are our reproductions. Costs are per full-field prediction (Appendix D.5).

	AoA 0°		AoA 5°		Random		Race Car		Takeoff		Latency	Inference
Model	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.	(s)	(TFLOP)
GAOT	0.3779	1.5488	0.3785	1.4200	0.6327	1.5234	0.3930	0.4560	0.5079	1.4007	**0.071**	**0.57**
Erwin	0.1233	0.5268	0.0395	0.2480	0.3135	1.0371	0.8817	2.5598	0.0930	0.5230	1.196	1.15
SpiderSolver	0.7446	0.5348	0.7726	0.8465	0.0165	0.0140	0.0335	0.0232	0.5141	0.3794	1.582	3.00
Transolver-3	0.0031	0.0031	0.0032	0.0044	0.0214	0.0139	0.0292	0.0171	0.0027	0.0033	0.211	2.82
AB-UPT	**0.0001**	0.0015	**0.0003**	0.0022	0.0021	0.0037	0.0141	0.0209	**0.0002**	0.0013	0.129	12.49
IBO (ours)	**0.0001**	**0.0009**	**0.0003**	**0.0020**	**0.0006**	**0.0027**	**0.0132**	**0.0141**	**0.0002**	**0.0009**	0.396	3.97

fields of four TandemFoilSet subsets, despite its superior performance on DrivAerML. In IBO, every leaf carries the same importance mass, so the regions between bodies, where the mesh or the wall-distance weight of Eq. (23) (Appendix D.3) is densest, can be deliberately assigned more leaves, ensuring adequate resolution in regions of complex physics. An anchor subset or a feature-grouped slice does not naturally prioritize these regions, potentially impacting performance. Figure 3 makes the difference visible on turbulent viscosity: IBO's residual is confined to thin bands along the building faces, AB-UPT's spreads through the passages and wakes, and Transolver-3 fails to resolve both near- and far-field structure. Other models tested, such as Erwin and GAOT, were comparatively even weaker in performance in our reproduction.

6.2 Complexity Analysis and Inference Cost

On DrivAerML, IBO needs an order of magnitude less inference FLOPs than Transolver-3 and AB-UPT (Table 3), with a similar advantage for UrbanWind (Figure 1a). Crucially, IBO's full-field inference cost is $C_{\text{case}} + O(bNkD_{\text{dec}}^2)$, which does not depend on the mesh size N. Each point is recovered from only its k nearest latent points through b decoder stages of width D_{dec}, which makes decoding highly efficient. Appendix D.5 provides further analysis.

Table 4: UrbanWind: relative L_2 errors in physical units and full-field inference costs. All baselines are our reproductions.

Model	U_x	U_y	U_z	p	ν_t	p_s	Latency (s)	Inference (TFLOP)
SpiderSolver	0.9145	0.1571	0.8192	0.0073	0.1886	0.0081	19.697	37.12
Erwin	0.6359	0.1246	0.6093	0.0146	0.3767	0.0256	15.538	29.85
Transolver-3	0.5811	0.0954	0.5678	0.0070	0.1263	0.0096	1.771	75.70
AB-UPT	0.1365	0.0222	0.1448	0.0006	0.0427	**0.0012**	3.246	479.50
IBO (ours)	**0.1267**	**0.0199**	**0.1335**	**0.0004**	**0.0337**	0.0023	**0.364**	**12.14**

6.3 Ablation Study

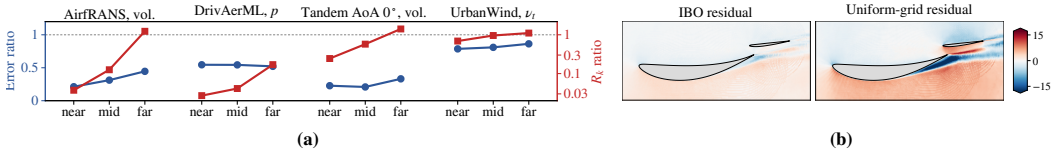


Figure 4: At the same token budget, IBO lowers the error at every wall distance. (a) IBO-to-uniform-grid ratios over equal-count wall-distance thirds. The R_k ratio (in red) quantifies the mean distance from a query to further, k latent point used. The low ratio suggests a concentration of nearer points. The error ratio (in blue) stays below one in every third even where the R_k ratio exceeds one, demonstrating consistent improvement. (b) u residuals on a TandemFoilSet Race Car case: the uniform grid's error concentrates in the gap and wakes. (c) Ground truth u.

We isolate the latent layout in a series of ablation studies, whereby the token budget, encoder, processor, decoder and training protocol of each configuration are fixed, and only change the latent points allocation. *Uniform grid* places them at the cell centres of a regular grid over the domain, similar to GINO and GAOT (Section 3); *adaptive, equal count* uses IBO's KD-tree partition with $w_i = 1$ (Section 5), thereby strictly following the pre-defined mesh. On UrbanWind's nearly regular mesh, *adaptive, importance-weighted* further uses the wall-distance weight of Eq. (23). Table 6 further shows an experiment where the latent token budget is halved. Appendix D.6 provides ablation results for all five TandemFoilSet subsets.

Table 6: Layout ablation on AirfRANS and DrivAerML (top) and on the TandemFoilSet Cruise AoA subsets and UrbanWind (bottom); metrics as in Tables 2–5. The importance weight applies only to UrbanWind.

	AirfRANS				DrivAerML					
Latent layout	Vol.	Surf.	C_L	ρ_L	p_s	τ	p	U	$C_D\ R^2$	$C_L\ R^2$
Uniform grid	0.0026	0.0050	0.0318	0.9996	8.74	12.70	6.37	8.25	0.823	0.666
Adaptive, equal count	**0.0009**	**0.0006**	**0.0146**	**0.9999**	**3.78**	**7.12**	**3.44**	**5.14**	**0.967**	**0.980**
Adaptive, $\frac{1}{2}$ token number	**0.0009**	**0.0006**	0.0168	0.9998	3.93	7.38	3.52	5.26	0.954	0.978
	AoA 0°		AoA 5°		UrbanWind					
Latent layout	Vol.	Surf.	Vol.	Surf.	U_x	U_y	U_z	p	ν_t	p_s
Uniform grid	0.0005	0.0099	0.0006	0.0075	0.1785	0.0272	0.1868	0.0007	0.0410	0.0029
Adaptive, equal count	**0.0001**	**0.0009**	**0.0003**	**0.0020**	0.1429	0.0222	0.1498	0.0005	0.0360	0.0027
Adaptive, importance-weighted	/	/	/	/	**0.1267**	**0.0199**	**0.1335**	**0.0004**	**0.0337**	**0.0023**

Table 6 shows that placement matters: with model and budget fixed, IBO's layout beats the uniform grid on every metric across all four benchmarks, and it still does so at half the budget on AirfRANS and DrivAerML. Figure 4a plots IBO-to-uniform-grid ratios of the error and of R_k, a proxy for local read-out resolution, over equal-count wall-distance thirds of the volume points. IBO consistently shows an improved balance between points both near and far away, with low error ratios regardless of distance. This is also consistent with the analysis in Section 4.

7 Limitations and Future Work

While we show the benefits of an improved allocation of latent tokens across the spatial domain, there remains a reliance on domain knowledge to determine the importance measure, e.g. in identifying locations one would refine as in a CFD solve, or a wall-distance weight. Importantly, this also opens avenues for future extensions, e.g. importance measures that incorporate more sophisticated estimates of wake zones or boundary layer thicknesses can further improve layout allocation. Related to this, we note that our focus is the layout-controlled part of the error, such that efforts to constrain inference costs are not unduly penalized by the spatially local nature of the tokenization and decoding. Hence, this is complementary to other efforts focusing on reducing the learning error. Lastly, the current demonstration is on steady-state flows, where we can pre-compute the importance measure and layout allocation. On transient flows, the most relevant spatial regions may be continually evolving, such that the latent token layout may need to be periodically re-computed. Ensuring this is computationally feasible, especially for scenarios with very rapidly changing flows, remains an interesting extension.

8 Conclusion

In current literature, spatially localized neural operators, i.e. with local encoding and local decoding, lag the strongest models by up to an order of magnitude. We analyze this through the lens of latent token placement, and show that the layout-controlled error is minimized when the token spatial spacing best approximates the flow complexity and geometry. By defining importance measures which take flow physics into account, and integrating that with a KD-tree partition, the newly introduced IBO improves layout-related error, and consequently model prediction, significantly. This results in state-of-the-art performance, with a uniquely Pareto-optimal combination of accuracy and inference computational cost on the 3D benchmarks. Overall, the work shows that while improved learning algorithms (e.g. in the processing step) are important, informed choices of the latent token placement in the spatial domain can also lead to remarkable gains in performance.

AI USE STATEMENT

In this work, we made extensive use of generative AI tools (large language models) across the research workflow. Among the tasks with required disclosure, AI tools were used extensively to help develop the theoretical model and conceptual framework, formulate the mathematical claims, provide ingredients for and assist in the writing of the proofs, propose and refine hypotheses, design and provide feedback on the methodology and experiments, implement the methods, clean and reformat data, and support the data analysis. They made minimal contributions to interpreting the results, and were not used to generate synthetic data sets. Among the tasks with recommended disclosure, AI tools were used extensively to draft and edit the manuscript, to create the figures and the figure-rendering code, and to identify, verify and format the cited literature. We have reviewed all AI-assisted work: the proofs were checked line by line by the authors, all code was tested and its outputs verified against the reference simulations, every citation was checked against the cited paper's text, and all results for IBO and our reproductions were produced by the authors' own runs, while published baseline results are attributed to their cited sources. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work involves no human subjects and uses no personal or sensitive data. The UrbanWind benchmark introduced in this work consists of steady RANS simulations of wind flow over the building footprints and heights of urban layouts (Low et al., 2022) but contains no personal information and will be released under the CC BY-SA 4.0 license. The other benchmarks, AirfRANS, DrivAerML and TandemFoilSet, are publicly available and are used in accordance with their licenses. We are not aware of direct negative societal impacts of this work. As with any surrogate model, predictions used for engineering decisions should be verified against high-fidelity simulation.

REPRODUCIBILITY STATEMENT

The method is detailed in Appendix C, and the architecture, features, training and decoding settings of IBO as used for each benchmark is in Appendix D.3. The datasets and splits, including the new UrbanWind benchmark, are described in Appendix D.1. The evaluation protocol, metrics and training seeds are in Appendix D.2. The baseline configurations and benchmarking protocol are in Appendix D.4. Appendix B states all assumptions and gives proofs of relevant theoretical results. Anonymized code is provided in the supplementary material; the code and UrbanWind benchmark will be made publicly available upon publication.

REFERENCES

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.

Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schönwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, Burkhard Hupertz, and Danielle C. Maddix. DrivAerML: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics. *arXiv preprint arXiv:2408.11969*, 2024.

Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A ViT backbone for diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model reduction and neural networks for parametric PDEs. *The SMAI Journal of Computational Mathematics*, 7:121–157, 2021. doi: 10.5802/smai-jcm.74.

Florent Bonnet, Jocelyn Ahmed Mazari, Paola Cinnella, and Patrick Gallinari. AirfRANS: High fidelity computational fluid dynamics dataset for approximating Reynolds-averaged Navier–Stokes solutions. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Junfeng Chen and Kailiang Wu. Positional knowledge is all you need: Position-induced transformer (PiT) for operator learning. In *International Conference on Machine Learning (ICML)*, 2024.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations (ICLR)*, 2024.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning (ICML)*, 2023.

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 (ICLR)*, 2021.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996.

Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. *arXiv preprint arXiv:2210.00612*, 2022.

Allen Gersho. Asymptotically optimal block quantization. *IEEE Transactions on Information Theory*, 25(4): 373–380, 1979. doi: 10.1109/TIT.1979.1056067.

Jan Hagnberger, Daniel Musekamp, and Mathias Niepert. CALM-PDE: Continuous and adaptive convolutions for latent space modeling of time-dependent PDEs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Weizhang Huang and Robert D. Russell. *Adaptive Moving Mesh Methods*. Springer, 2011.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

Loh Sher En Jessica, Naheed Anjum Arafat, Wei Xian Lim, Wai Lee Chan, and Adams Wai-Kin Kong. Finite volume features, global geometry representations, and residual training for deep learning-based CFD simulation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 21555–21576, 2024. URL https://proceedings.mlr.press/v235/jessica24a.html.

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

Samuel Lanthaler, Siddhartha Mishra, and George E. Karniadakis. Error estimates for DeepONets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022. doi: 10.1093/imatrm/tnac001.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Zongyi Li, Nikola B. Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya P. Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3D PDEs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Wei Xian Lim, Sher En Jessica Loh, Zenong Li, Thant Zin Oo, Wai Lee Chan, and Adams Wai-Kin Kong. TandemFoilSet: Datasets for flow field prediction of tandem-airfoil through the reuse of single airfoils. In *International Conference on Learning Representations (ICLR)*, 2026.

Songming Liu, Zhongkai Hao, Chengyang Ying, Hang Su, Ze Cheng, and Jun Zhu. Non-uniform neural operator. In *International Conference on Machine Learning (ICML)*, 2023.

Shi Jer Low, Venugopalan SG Raghavan, Harish Gopalan, Jian Cheng Wong, Justin Yeoh, and Chin Chun Ooi. Fastflow: Ai for fast urban wind velocity prediction. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 147–154. IEEE, 2022.

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, 2021.

Mohammad Amin Nabian, Chang Liu, Rishikesh Ranade, and Sanjay Choudhry. X-meshgraphnet: Scalable multi-scale graph neural networks for physics simulation. *arXiv preprint arXiv:2411.17164*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Kai Qi, Fan Wang, Zhewen Dong, and Jian Sun. SpiderSolver: A geometry-aware transformer for solving PDEs on complex geometries. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 23rd ACM National Conference*, pp. 517–524, 1968.

Philippe R. Spalart. Strategies for turbulence modelling and simulations. *International Journal of Heat and Fluid Flow*, 21(3):252–263, 2000.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17: 261–272, 2020.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2505.18781v4. Extended version: arXiv:2505.18781v4, 2026.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *International Conference on Machine Learning (ICML)*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Maksim Zhdanov, Max Welling, and Jan-Willem van de Meent. Erwin: A tree-based hierarchical transformer for large-scale physical systems. In *International Conference on Machine Learning (ICML)*, 2025.

Hang Zhou, Haixu Wu, Haonan Shangguan, Yuezhou Ma, Huikun Weng, Jianmin Wang, and Mingsheng Long. Transolver-3: Scaling up transformer solvers to industrial-scale geometries. In *International Conference on Machine Learning (ICML)*, 2026.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*, 2018.

A SPATIALLY LOCALIZED MODELS

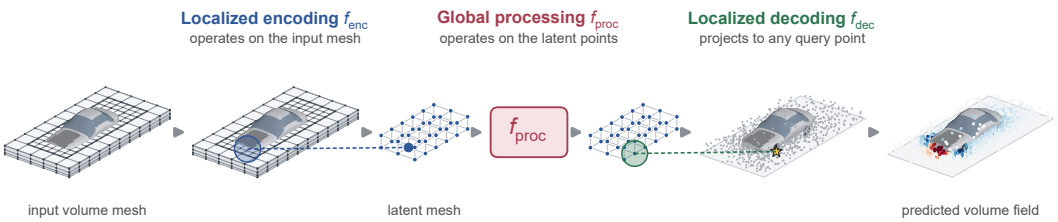


Figure 5: The spatially localized family. The local encoder f_{enc} aggregates the input mesh points in a neighbourhood of each latent point into one token; the processor f_{proc} exchanges information among the latent points; and the local decoder f_{dec} reconstructs the field at any query point, marked by the star, from the latent points around it.

This appendix extends Section 3. Figure 5 traces one case through the three stages of Eqs. (1)–(3). Local encoding starts from the input volume mesh: each latent point summarizes the mesh points in its region, so the latent points form a mesh of their own, the latent layout $\mathcal{A}$. Global processing operates on this latent mesh only. Local decoding then projects the processed tokens to any query location, which need not coincide with a mesh point, by reading the latent points in its neighbourhood.

Definition. A *spatially localized model* is a neural operator that computes through a latent layout $\mathcal{A}$ of positions in physical space. Its encoder forms the token of each latent point only from input points in a spatial neighbourhood of that point; its decoder predicts the output at a query point from latent points in a spatial neighbourhood around the query. The latent layout is therefore the medium of the model: the input mesh is used only by the local encoder, and the output locations only by the local decoder.

B Additional Proofs and Theory

Table 7: Notation of Section 4 and this appendix. The subscript $s \in \{\mathrm{enc}, \mathrm{dec}\}$ refers to the encoding and decoding sides.

Symbol	Meaning
$\Omega,\ d,\ \lvert\Omega\rvert$	Bounded volume domain, spatial dimension, and domain volume.
$\xi,\ u,\ \mu$	Geometric descriptor field, target physical field, and case condition (fixed and suppressed).
$\mathcal{A},\ L$	Discrete latent layout and its number of latent points, the budget.
$\lambda,\ \lambda^\star,\ \lambda_{\mathrm{unif}}$	Continuum point density with $\int_\Omega \lambda\, dx = L$, its minimizer, and the uniform density $L/\lvert\Omega\rvert$.
$q,\ \rho$	Normalized evaluation and importance densities; the evaluation setup gives $q = \rho$.
$\lVert\cdot\rVert_2\,,\ \lVert\cdot\rVert_q$	Euclidean norm at a point, and q-weighted L^2 norm of a field.
$\mathcal{R}_{\mathcal{A}},\ \widetilde{\xi}_{\mathcal{A}}$	Reference reconstruction, and geometric context recovered from the reference encoder.
$\widehat{u}_{\theta,\mathcal{A}},\ \theta$	Learned prediction and its parameters.
$\varepsilon_{\mathrm{enc}},\ \varepsilon_{\mathrm{dec}},\ \varepsilon_{\mathrm{learn}}$	Encoding, reconstruction and learning errors, Eq. (4).
$\overline{\varepsilon}_s^2[\lambda]$	Squared resolution-error bound, Eqs. (7) and (13).
$\kappa_s,\ r_s,\ \beta_s$	Local difficulty, approximation order, and exponent $\beta_s = 2r_s/d$.
c_s	Positive matching constant in $\kappa_s = c_s\rho$.
$\eta,\ C,\ \nu$	Relative deviation $\lVert\lambda/\lambda^\star - 1\rVert_\infty$, stability constant, and first-order constant.
$C_j,\ \overline{\rho}_j,\ \lambda_L$	Partition leaf, its mean importance density, and the induced piecewise-constant density.

B.1 Encoder resolution error

For $\varepsilon_{\mathrm{enc}}$ in Eq. (4), assume the corresponding order-r_{enc} resolution bound

$$\varepsilon_{\mathrm{enc}}^2 \le \overline{\varepsilon}_{\mathrm{enc}}^2[\lambda] := \int_\Omega q(x) \left(\frac{\kappa_{\mathrm{enc}}(x)}{\lambda(x)}\right)^{\beta_{\mathrm{enc}}} dx, \qquad \beta_{\mathrm{enc}} := \frac{2r_{\mathrm{enc}}}{d} > 0. \tag{13}$$

Here $\kappa_{\mathrm{enc}}(x)$ is the effective local approximation difficulty on the encoding side, including the effect of lost geometric information on field reconstruction. It is fixed across layouts. The encoder and decoder bounds have the same form, with potentially different approximation orders and difficulty profiles.

B.2 Joint optimal allocation

Given the evaluation setup $q = \rho$ and the matching relation $\kappa_{\mathrm{dec}} = c_{\mathrm{dec}}\rho$ from Assumption 4.1, and the corresponding encoder relation $\kappa_{\mathrm{enc}} = c_{\mathrm{enc}}\rho$, substitution gives

$$\overline{\varepsilon}_{\mathrm{enc}}^2[\lambda] + \overline{\varepsilon}_{\mathrm{dec}}^2[\lambda] = \sum_{s\in\{\mathrm{enc},\mathrm{dec}\}} c_s^{\beta_s} \int_\Omega \rho(x)^{1+\beta_s} \lambda(x)^{-\beta_s}\, dx. \tag{14}$$

We use the Hölder argument from classical point-density allocation (Gersho, 1979, Sec. VI). For either term, the budget constraint gives

$$\begin{aligned} 1 = \int_\Omega \rho(x)\, dx &= \int_\Omega \left(\rho(x)^{1+\beta_s}\lambda(x)^{-\beta_s}\right)^{\frac{1}{1+\beta_s}} \lambda(x)^{\frac{\beta_s}{1+\beta_s}}\, dx \\ &\le \left(\int_\Omega \rho(x)^{1+\beta_s}\lambda(x)^{-\beta_s}\, dx\right)^{\frac{1}{1+\beta_s}} L^{\frac{\beta_s}{1+\beta_s}}. \end{aligned} \tag{15}$$

Hence

$$\overline{\varepsilon}_{\mathrm{enc}}^2[\lambda] + \overline{\varepsilon}_{\mathrm{dec}}^2[\lambda] \ge \left(\frac{c_{\mathrm{enc}}}{L}\right)^{\beta_{\mathrm{enc}}} + \left(\frac{c_{\mathrm{dec}}}{L}\right)^{\beta_{\mathrm{dec}}}. \tag{16}$$

Equality in Hölder's inequality holds if and only if

$$\rho(x)^{1+\beta_s}\lambda(x)^{-\beta_s} \propto \lambda(x), \qquad \text{equivalently} \qquad \lambda(x) \propto \rho(x)$$

almost everywhere. The budget fixes the proportionality constant to L. Thus $\lambda^\star = L\rho$ attains the lower bound in Eq. (16), which simultaneously minimizes both terms. At this density, $\kappa_s(x)/\lambda^\star(x) = c_s/L$: the local approximation bounds are equalized across the domain. Regions with greater

importance receive more latent resolution, even when the two sides have different approximation orders.

The decoder term alone gives Proposition 4.2. Substituting $\lambda_{\mathrm{unif}} = L/|\Omega|$ into that term gives Eq. (10), since Hölder's inequality also implies

$$\int_\Omega \rho(x)^{1+\beta_{\mathrm{dec}}}\, dx \geq |\Omega|^{-\beta_{\mathrm{dec}}},$$

with equality if and only if ρ is constant almost everywhere.

B.3 Second-order stability

This result allows general positive difficulty functions and does not require importance matching. Suppose the joint bound has a positive minimizing density $\lambda^\star$ with a finite, nonzero value. For a density λ with the same budget, define $\eta := \|\lambda/\lambda^\star - 1\|_\infty$. If $\eta \leq 1/2$, then

$$0 \leq \frac{\sum_{s\in\{\mathrm{enc,dec}\}} \overline{\varepsilon}_s^{\,2}[\lambda]}{\sum_{s\in\{\mathrm{enc,dec}\}} \overline{\varepsilon}_s^{\,2}[\lambda^\star]} - 1 \leq C\eta^2, \qquad C := \max_{s\in\{\mathrm{enc,dec}\}} 2^{\beta_s+1}\beta_s(\beta_s+1). \tag{17}$$

Proof. The first-order condition gives a spatial constant ν such that

$$\sum_s \beta_s q(x)\kappa_s(x)^{\beta_s}\lambda^\star(x)^{-1-\beta_s} = \nu. \tag{18}$$

The second-derivative bound for the power function and the relative deviation bound give the pointwise inequality

$$\begin{aligned} 0 &\leq \lambda^{-\beta_s} - (\lambda^\star)^{-\beta_s} + \beta_s(\lambda^\star)^{-1-\beta_s}(\lambda - \lambda^\star) \\ &\leq 2^{\beta_s+1}\beta_s(\beta_s+1)\eta^2(\lambda^\star)^{-\beta_s}. \end{aligned} \tag{19}$$

Multiply by $q\kappa_s^{\beta_s}$, integrate and sum. By Eq. (18), the integrated linear term is $\nu\int_\Omega(\lambda-\lambda^\star)\,dx = 0$. The remaining terms satisfy

$$0 \leq \sum_s \big(\overline{\varepsilon}_s^{\,2}[\lambda] - \overline{\varepsilon}_s^{\,2}[\lambda^\star]\big) \leq C\eta^2 \sum_s \overline{\varepsilon}_s^{\,2}[\lambda^\star].$$

Division by the minimum gives Eq. (17). Keeping only the decoder term gives Eq. (11) by the same argument. □

C METHOD DETAILS

This appendix specifies the spatial partition, input encoding, neural modules and training procedure of IBO independently of the benchmark; Appendix D.3 gives specific settings used on each dataset. IBO only uses geometric descriptors and case conditions and does not require mesh connectivity.

C.1 LATENT-LAYOUT CONSTRUCTION

Partition. Each KD-tree node is split along the axis of largest extent. With equal point weights, the coordinates are sorted along that axis and divided into two groups whose sizes differ by at most one; coordinate ties can be separated by the stable ordering when membership is defined by point indices. With positive weights w_i, the sorted cumulative mass determines the median cut. To preserve a coordinate-based partition when points share the split coordinate, the cut moves to the end of the tied run that gives the closest feasible half-mass split, keeping both children nonempty when possible. A node with one point or identical coordinates cannot be subdivided further. The number of levels fixes L. Membership is stored explicitly, and the latent point is the weighted centroid of its region, the mean of its coordinates when $w_i = 1$. Any unused token slot takes its box centre and has zero member features; it remains a decoder location and carries no importance mass.

Encoder inputs. The partition records every input's membership. To control the encoding cost, the encoder processes either selected members or subregion representatives. Members are drawn uniformly without replacement up to a member cap, using a fixed case seed, and all are retained if the region contains fewer than the cap; padding slots are zero and excluded from pooling. Subregion representatives come from further levels of the same tree inside each region, and the encoder then uses the subregion mean positions and aggregated geometry. Every input thus contributes to a subregion representative.

C.2 LOCAL ENCODER

For a query or member position, s denotes signed distance to the nearest wall and n its unit wall direction; on a wall, n is the outward normal. Let S_j be the selected members or representatives of region C_j, and let ϕ_i concatenate a member's relative position $x_i - a_j$ and its geometric features. The shared pointwise network ψ and the token projection f_h compute

$$v_j^{\mathrm{pool}} = \left[\max_{i \in S_j} \psi(\phi_i);\ \mathrm{mean}_{i \in S_j}\, \psi(\phi_i);\ \mathrm{std}_{i \in S_j}\, \psi(\phi_i) \right], \tag{20}$$

$$h_j = f_h([v_j^{\mathrm{pool}}; b_j]), \tag{21}$$

where the region descriptor b_j contains the latent point and coarse geometry at that position. All three reductions are channelwise and mask padding; a region with no valid rows receives zero pooled statistics. The pointwise network ψ has two linear layers with a pointwise nonlinearity between them, and f_h has one or two linear layers; pooling runs in fp32. When a case condition is present, it modulates the encoder through FiLM (Perez et al., 2018). FiLM applies a learned per-channel scale and shift initialized to zero, so its initial action is the identity.

C.3 GLOBAL PROCESSOR

The processor uses pre-norm self-attention and SwiGLU feed-forward blocks (Shazeer, 2020), with token width $D = 384$, feed-forward hidden width $4D$, dropout 0.1 and a final RMS normalization. Queries and keys are RMS-normalized per head (Zhang & Sennrich, 2019; Dehghani et al., 2023). Rotary embeddings (Su et al., 2024) allocate dimensions separately to the coordinate axes. Positions are normalized using fixed dataset-level coordinate ranges. Each axis carries a geometric ladder of frequencies. Its coordinate is mapped to $[0, 1]$ over the extent of the latent points across all cases, padded by 10% on each side; the angular scale is $\mathrm{round}\big(\frac{32}{6} \times \mathrm{extent}\big)$, so the finest frequency turns through $32/6$ radians per unit length.

Where used, learned registers (Darcet et al., 2024) are appended to the spatial tokens and receive zero rotary angle. We write the processor output as spatial rows $Z \in \mathbb{R}^{L \times D}$ and separate register rows Z_{reg}. Gated long skips, where used, connect corresponding early and late blocks (Bao et al., 2023),

using RMS normalization, a learned projection, and scalar gates initialized to 0.01. Case-conditioned blocks use zero-initialized scale, shift and residual-gate modulation as in adaLN-Zero (Peebles & Xie, 2023); these residual branches initially act as the identity.

C.4 Local decoder

Query and pair features. Let $g(x) \in \mathbb{R}^{D_{\mathrm{dec}}}$ be a learned query feature, with decoder width D_{dec}, and let $\mathcal{N}(x)$ index its k nearest latent points in physical coordinates. The neighbours are found once per case with a KD-tree query (Virtanen et al., 2020) and stored as indices sorted by distance. Define $d_j(x) = \|x - a_j\|$ and $\bar{d}(x) = k^{-1} \sum_{j \in \mathcal{N}(x)} d_j(x)$. A pair feature $p_j(x)$ embeds the normalized displacement $(x - a_j)/\bar{d}$ and the log distance ratio $\ln(d_j/\bar{d})$. Distances inside logarithms and denominators use a positive numerical floor at coincident positions. Query and pair features use Fourier coordinates (Tancik et al., 2020)

$$\Phi_F(v) = [v, \sin(2^t \pi v), \cos(2^t \pi v)]_{t=0}^{F-1},$$

applied componentwise to their normalized inputs with F frequency bands. Query features also include wall geometry, $\ln \bar{d}$, the available geometric descriptors and, when present, the case condition. Separate first layers for surface and volume queries feed a shared second layer and learned type embedding.

Local attention. For a head of width d_{head}, let $t_j \in \mathbb{R}^D$ denote the token read by the current decoder stage: h_j in a stage that reads H, and z_j in a stage that reads Z. The pair embedding $p_j \in \mathbb{R}^{D_{\mathrm{dec}}}$ matches g; $p_j^{\mathrm{head}} \in \mathbb{R}^{d_{\mathrm{head}}}$ is the slice of its channels assigned to this head. The learned maps W_Q, W_K, W_V project to d_{head} dimensions:

$$\begin{aligned} Q_j &= \mathrm{RMSNorm}\left(W_Q(g + p_j)\right), \qquad K_j = \mathrm{RMSNorm}(W_K t_j), \quad V_j = W_V t_j, \\ \omega_j &= \mathrm{softmax}_{j \in \mathcal{N}(x)} \left(\frac{Q_j^\top K_j}{\sqrt{d_{\mathrm{head}}}} - \zeta \ln \frac{d_j}{\bar{d}} \right), \\ o &= \sum_{j \in \mathcal{N}(x)} \omega_j (V_j + \gamma \odot p_j^{\mathrm{head}}). \end{aligned} \tag{22}$$

Here ω_j is the normalized attention weight, ζ is a learned scalar per head, $\gamma \in \mathbb{R}^{d_{\mathrm{head}}}$ is a learned vector, and $\odot$ denotes elementwise multiplication. Concatenating the head outputs and applying an output projection W_O gives the residual updates

$$g \leftarrow g + W_O \,\mathrm{Concat}_{\mathrm{heads}}(o), \qquad g \leftarrow g + \mathrm{FFN}(\mathrm{RMSNorm}(g)),$$

both in $\mathbb{R}^{D_{\mathrm{dec}}}$, where FFN is a SwiGLU feed-forward layer of hidden width $4D_{\mathrm{dec}}$. The distance term multiplies content-based attention by $(d_j/\bar{d})^{-\zeta}$; initializing $\zeta = 1$ gives an inverse-distance *prior* (Shepard, 1968), not pure inverse-distance attention in the presence of nonconstant content scores. Scores are accumulated in fp32. Keys and values are projected once per token before queries are decoded in chunks.

Stages and output heads. The decoder has one or two stages. With two stages, the first reads H and the second reads Z. When registers are present, the last stage also reads them using a separate softmax and a gated addition, without spatial pair features. A two-layer SiLU head maps the final query feature to the output channels, either through a shared output vector with supervision restricted to the channels of each query type, or through separate volume and surface heads.

C.5 Training and inference

Targets are standardized per channel. Relative positions and scalar geometric features are standardized separately for member and region populations; unit vectors, flags and fractions retain their original scales. Physical coordinates remain available for distances and rotary embeddings. Each step samples queries uniformly from a case's query pool and minimizes a squared error on standardized targets, combined over channels as specified for each benchmark in Appendix D.3. Training uses Lion (Chen et al., 2023) with coefficients $(0.9, 0.99)$, decoupled weight decay, linear warm-up, cosine

learning-rate decay and gradient-norm clipping at 1.0. Computation uses bf16 autocast with fp32 reductions, normalizations, trigonometric features and attention scores. Fixed-size member and query tensors support compiled execution. Final weights average epoch-end checkpoints from the last quarter of training (Izmailov et al., 2018). At inference, the encoder and processor are evaluated once per case, and the requested field locations are decoded in fixed-size chunks, padding the last chunk. Partitions, geometric descriptors and neighbour indices are reused across training epochs and evaluation. Appendix D.2 describes the training seeds.

D EXPERIMENTAL DETAILS

This appendix describes the benchmarks (Appendix D.1), the evaluation protocol and training seeds (Appendix D.2), the implementation of IBO on each benchmark (Appendix D.3), the baselines and benchmarking protocol (Appendix D.4), the complexity and cost analysis (Appendix D.5), additional ablations (Appendix D.6) and qualitative results (Appendix D.7).

D.1 DATASETS

AirfRANS. AirfRANS (Bonnet et al., 2022) contains 1,000 steady incompressible RANS simulations of two-dimensional flow around NACA four- and five-digit airfoils, generated by varying the airfoil shape, the Reynolds number ($Re = 2$–6×10^6) and the angle of attack (-5° to 15°). The simulations use the k–ω SST turbulence model, and pressure is kinematic. Each case provides a body-fitted mesh of about 10^5 nodes, refined towards the wall and cropped to $(-2, 4) \times (-1.5, 1.5)$ m, with the velocity components u_x, u_y, pressure p and turbulent viscosity ν_t at every node. The benchmark predicts these four volume fields and the surface pressure; surface velocity and turbulent viscosity are fixed by the boundary condition and are not learned surface targets. The full-data task contains 800 training and 200 test cases. We reserve 80 of the training cases with a fixed seed, giving a 720/80/200 split; the held-out validation cases are not used for model selection.

DrivAerML. DrivAerML (Ashton et al., 2024) contains high-fidelity simulations of 500 parametric morphs of the DrivAer car at fixed inflow conditions. The flow is computed with hybrid RANS–LES, and all provided fields are time-averaged. A case has approximately 1.4×10^8 volume cells and 8.8×10^6 surface cells. The benchmark predicts surface pressure p_s and wall shear stress τ, and volume pressure p and velocity U; pressure and shear stress are kinematic. The dataset has no official split. Our comparison uses AB-UPT's split sizes of 400/34/50, covering the 484 cases with available simulations (Alkin et al., 2025). The complete prediction cloud contains the vehicle surface and the volume crop $[-30, 60] \times [-8, 8] \times [-0.5, 8]$ m, approximately 1.4×10^8 points in total.

TandemFoilSet. TandemFoilSet (Lim et al., 2026) provides 8,104 two-dimensional steady flow simulations of single and tandem airfoils in nine subsets, computed with OpenFOAM and the k–ω SST model. A tandem case combines a structured background mesh with two body-fitted overset meshes, one per airfoil, and is solved with an overset SIMPLE solver; pressure is kinematic. We use the five tandem subsets in Table 8, training a separate model on each subset. Cruise AoA 0°, Cruise AoA 5° and Takeoff are flows at $Re = 500$, and Takeoff adds a ground plane; Cruise Random and Race Car are high-Reynolds-number flows, and Race Car places an inverted main element and flap above the ground. The benchmark predicts the two volume velocity components and pressure, and surface pressure. An 8:1:1 split is drawn with a fixed seed after filtering non-finite fields. For the two high-Reynolds-number subsets, the preprocessing also rejects cases whose maximum pressure coefficient exceeds 1.1. This removes 11 Race Car cases.

Table 8: The five tandem-airfoil subsets of TandemFoilSet (Lim et al., 2026): number of cases after the data-quality gate, Reynolds number, angle of attack, and the ranges of stagger, gap and height above ground (in chord lengths), with the mean cell count per case.

Subset	Cases	Re	AoA	Stagger	Gap	Height	Cells
Cruise AoA 0°	784	500	0°	$[0.5, 2]$	$[-0.4, 0.4]$	–	3.5×10^5
Cruise AoA 5°	784	500	5°	$[0.5, 2]$	$[-0.4, 0.4]$	–	3.5×10^5
Cruise Random	900	10^5–5×10^6	$[-5^\circ, 6^\circ]$	$[0.5, 2]$	$[-0.8, 0.8]$	–	2.1×10^5
Race Car	889	10^6–5×10^6	$[-10^\circ, 0^\circ]$	$[-0.5, 0.05]$	$[0.05, 0.1]$	$[0.1, 1.1]$	1.3×10^5
Takeoff	784	500	5°	$[0.5, 2]$	$[-0.2, 0.6]$	$[0.4, 1]$	2.7×10^5

UrbanWind. UrbanWind is introduced in this work. The full collection contains 6,976 steady RANS simulations of atmospheric boundary-layer flow over 1,836 urban layouts (Low et al., 2022), with up to four wind directions per estate; building footprints and heights are taken from real-world layouts. Simulations use an in-house OpenFOAM-based solver on a $6\,\mathrm{km} \times 6\,\mathrm{km}$ domain whose height is six times the rounded-up height of the tallest building, so the domain height varies between

estates. A 100 m background spacing is refined through five nested regions to 3.125 m horizontally near the estate, with vertical grading and wall snapping and without prism layers. The inflow uses a standard atmospheric log-law profile, with speed 2 m/s at a reference height of 15 m and aerodynamic roughness 1 m, reduced to 0.05 m on the ground and building walls within 600 m of the estate centre. Each simulation runs at most 750 under-relaxed outer iterations with a pressure residual target of 10^{-4}. The pressure is kinematic and contains a hydrostatic component linear in height. Wind directions are rigid rotations of the inflow over the same geometry.

The benchmark uses 400 training, 40 validation, and 40 test layouts, each with four wind directions: 1,600/160/160 cases. Validation estates are held out and are not used for model selection. All directions of one estate remain in the same split. The benchmark, rather than the full collection, is the 1,920-case release described in Section 1. Learning uses the central $1{,}100 \times 1{,}100 \times 160\,\mathrm{m}$ region. Geometry is rotated to a common wind-aligned frame; the four directions share the same estate geometry. Training estates are selected from those with at least 16,384 surface points, accommodating AB-UPT's surface anchor budget. Targets comprise three volume velocity components, pressure, turbulent viscosity, and building-surface pressure. All benchmark results use this subset. Normalization and detrending, including the use of held-out cases in the UrbanWind normalization statistics, are documented in Appendix D.3. Table 9 reports statistics of the full collection.

Table 9: UrbanWind: per-estate statistics over the 1,836 clean estates (the geometry is shared by the wind directions of an estate), and the number of wind directions per estate.

Quantity	mean	median	P5	P95	range
Surface points per estate	42,618	35,792	3,073	102,696	637–142,406
Volume cells per estate	2.72×10^6	2.51×10^6	1.73×10^6	4.53×10^6	1.38–5.16×10^6
Buildings per estate	36.4	32	3	87	1–124
Building height (m)	36.3	34.7	6.4	71.0	3.0–141.1
Wind directions per estate	4: 1,629 estates; 3: 69; 2: 115; 1: 23 (6,976 cases in total)				

D.2 Evaluation protocol and training seeds

Evaluation. On AirfRANS, evaluation uses all test-mesh nodes. Field MSE is computed on standardized targets and averaged over cases and the channels in each group. Lift is integrated from the predicted surface pressure and the wall shear stress of the predicted velocity field; its relative error and Spearman correlation are evaluated across the 200 test cases. On DrivAerML, reported accuracy uses a frozen uniform sample of up to five million points of each type per test case; query-pool evaluation and the full-mesh reconstruction procedure are separate choices. Field errors are computed in physical units, jointly over the components of velocity U and wall shear stress τ, then averaged over the 50 cases. C_D and C_L are integrated over the complete surface mesh with area-weighted quadrature, and coefficient R^2 is computed across cases; the coefficient definitions follow AB-UPT. On TandemFoilSet, volume MSE averages (u, v, p) over all nodes and surface MSE evaluates pressure at wall nodes, both on the physical fields standardized with training-split statistics; the complete test meshes are decoded. On UrbanWind, all volume and surface points of the 160 test cases are decoded, and relative L_2 errors are computed per channel and case.

Metrics and aggregation. For a scalar channel with normalization scale σ, standardized MSE is $N^{-1}\sum_i((\hat{u}_i - u_i)/\sigma)^2$. Group errors average over channels and then cases. Relative L_2 is $\|\hat{u} - u\|_2/\|u\|_2$ in physical units after inverting normalization and detrending; vector norms include all components. AirfRANS lift error is the mean of $|\hat{C}_L - C_L|/|C_L|$. The geometric mean in Figure 1a uses the six UrbanWind field errors $(U_x, U_y, U_z, p, \nu_t, p_s)$, expressed as percentages before aggregation.

Training seeds. Tables report means over training seeds without standard deviations. IBO and our AB-UPT reproduction are trained with five seeds on AirfRANS and TandemFoilSet (42, 1042, 2042, 3042 and 4042) and with three on UrbanWind (42, 1042 and 2042); on DrivAerML, where no baseline is reproduced, IBO uses three seeds (42, 1042 and 2042). The remaining reproduced baselines, and configurations that appear only in ablations, are single runs with seed 42. Rows shared

between a main table and an ablation table report the multi-seed mean in both. We use several seeds for IBO and AB-UPT because they are the two strongest models on these benchmarks and are separated by small margins on several metrics. A seed sets the weight initialization and dropout, the per-epoch order of the training cases, and the per-step training draws: IBO's query subsample and AB-UPT's training anchors. These draws are keyed by case and epoch and salted by the seed, so seed 42 reproduces the original draws. The data split, the latent partition and encoder-member selection, the geometric descriptors, the normalization statistics and the evaluation points are fixed across seeds. AB-UPT's inference anchor draws are seeded by the test case and are identical across training seeds.

D.3 IBO IMPLEMENTATION PER BENCHMARK

Table 10 lists the architecture on each benchmark and Table 11 the training settings. The paragraphs below give the remaining benchmark-specific choices for the components of Appendix C.

Table 10: IBO architecture by dataset.

	AirfRANS	TandemFoilSet	DrivAerML	UrbanWind
Spatial dimension d	2	2	3	3
Token count L	4,096	4,096	32,768	16,384
Token width D	384	384	384	384
Member cap K_{enc}	25	subset mean	128 subregions	32
Member feature dimension	15	18	8	13
Region descriptor dimension	13	16	7	22
PointNet widths	15–128–192	18–128–192	8–128–192	13–64–256
Token projection widths	589–384–384	592–384–384	583–384	790–384
Processor blocks	5	5	12	12
Processor heads × width	6×64	6×64	12×32	12×32
Register tokens	4	4	8	0
Long skips	yes	yes	yes	no
Decoder width	384	384	384	192
Decoder heads	6	6	6	6
Local neighbours k	8	8	16	8
Decoder stages	2	2	2	1
Output-head hidden width	512	512	256	256

Table 11: Training and full-field decoding settings. Learning rates refer to their peak values.

	AirfRANS	TandemFoilSet	DrivAerML	UrbanWind
Queries per case and step	32,000	32,000	128,000	128,000
Epochs	398	400	400	400
Cases/GPU × GPUs	4×4	1×4	1×4	1×4
Peak learning rate	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$3 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
Weight decay	0.05	0.05	0.01	0.01
Warm-up	5%	5%	5%	8,000 steps
Final/peak learning rate	0.02	0.02	0.10	0.10
Decode chunk size	32,000	32,000	262,144	262,144

AirfRANS and TandemFoilSet. Both benchmarks use all input mesh nodes with $w_i = 1$, so the partition is the equal-count tree. AirfRANS selects at most 25 members per region; TandemFoilSet uses a cap equal to the rounded mean occupancy of its subset. Two-dimensional walls are polygons, built per body by chaining its surface nodes; the total change in angle around the boundary determines whether a point is inside or outside. TandemFoilSet retains separate (s, n) pairs for the front and rear airfoils. Both use an eight-sector directional distance descriptor based on Directional Integrated Distance (DID) (Jessica et al., 2024). Each sector averages capped distances to polygon vertices, uses the cap when empty, and blends toward the cap under partial coverage. The cap is two chord lengths on AirfRANS and five on TandemFoilSet; two-body descriptors are combined with larger weight for the nearer body. A curvature proxy is the circular variance of wall-direction angles among eight neighbouring mesh points. The region descriptor contains position, wall distances and directions, and DID. AirfRANS uses the two Cartesian inflow components as the case condition,

while TandemFoilSet uses inflow speed and angle of attack. Their standardized conditions modulate the encoder through FiLM, modulate processor blocks through a separate condition projection per block, and enter the query features. The PointNet is Linear–SiLU–Linear, and the token projection has two linear layers with SiLU between them; the first pointwise layer also runs in fp32. Rotary embeddings allocate $(32, 32)$ dimensions to the two axes; the frequency base is set per axis so that the coarsest frequency sweeps exactly one cycle across the axis, and AirfRANS and every TandemFoilSet subset use their own calibrated values. Both processors use registers and gated long skips, and both decoders use two stages with eight Fourier bands; ζ and $\ln(d_j/\bar{d})$ are clipped to $[-8, 8]$. Targets are standardized per channel using training-split statistics, and query coordinates use training-split percentile scaling; TandemFoilSet first scales velocity by inflow speed and pressure by its square. Training samples its queries from all mesh points, and the loss gives equal weight to the volume-group and surface-pressure means.

DrivAerML. The volume input is the crop of Appendix D.1, while the surface input is retained. An equal-count tree over the combined points has 15 levels defining $L = 32{,}768$ regions and a further seven levels defining 128 subregions within each region. The feature encoder uses the subregion mean positions, wall distances, wall directions and surface fractions, and the latent point of the parent region is its full point mean. Wall distance uses the nearest point on the triangulated vehicle surface, found with the ray-casting scene of Open3D (Zhou et al., 2018), and the surface normals are oriented once per case by integrating the reference drag and lift; volume nodes outside the vehicle have nonnegative distance, while a region representative may lie inside and receives signed distance. The region descriptor contains position, signed wall distance and wall direction, and no case-level condition is used. The token projection is a single linear layer, and the first pointwise layer runs in fp32. Rotary embeddings allocate $(12, 12, 8)$ dimensions with bases $(152, 9.0, 5.3)$ and scales $(462, 39, 21)$, and the processor uses registers and gated long skips. The decoder uses two stages, and the neighbour indices store 16 neighbours per point. Pair features add unit-direction and local-spacing features, and the Fourier coordinates use six bands for pair displacement, five for wall distance and four for position; ζ and $\ln(d_j/\bar{d})$ are clipped to $[-8, 8]$. Targets are standardized with training-split statistics, and query coordinates use training-split percentile scaling. The training pool uses one volume point per subregion, 2^{22} in total, and up to five million sampled surface points, and the loss averages per-channel errors, each over the rows of the appropriate type.

UrbanWind. UrbanWind partitions volume points under the wall-distance weight

$$w_i = \big[\max(s_i, d_0)\big]^{-1/2}, \qquad d_0 = 2\,\mathrm{m}, \tag{23}$$

where $s_i \geq 0$ is the distance from a volume point to the nearest building face and d_0 caps the weight near the wall. Surface points are attached after the volume partition. Candidate (surface point, region) pairs are visited in increasing distance order, assigning each surface point to the nearest region with an available slot. These points occupy the first of the 32 encoder slots, and uniformly selected volume members fill the remaining slots. The importance measure refers to the volume partition; the added surface features supply boundary geometry. Buildings are represented by footprint polygons extruded through their height intervals, extracted from the surface points by DBSCAN clustering of the wall points (Ester et al., 1996) and simplification of their concave hull, and distances are computed to wall faces and roofs. Edge and corner directions use blended face normals. Six ray distances to the nearest building face are taken along the signed coordinate axes, capped at 550 m. Query features also include learned four-dimensional embeddings for each of three categorical codes: query type, geometric region or face orientation, and surface-edge type combined with wind group. The region descriptor contains position and 19 regional statistics: log point count, log point density, surface fraction, six trace-normalized position-covariance entries, three moments of distance to the latent point, four wall-distance summaries, and three box extents; empty regions use zero statistics. The processor is conditioned on five morphology scalars, log building count, total footprint area, wind-facing frontal area, mean building height and maximum building height, through a shared width-64 condition network with per-block output heads. The PointNet uses ReLU in place of SiLU, and the token projection is a single linear layer. Coordinates use the 550 m spatial scale; rotary embeddings allocate $(12, 12, 8)$ dimensions with bases $(450, 450, 64)$ and scale 500 on each axis, and the processor omits registers and long skips. The decoder uses one stage over Z and no register path, and its neighbour search runs when a case is loaded. Pair features add unit-direction and local-spacing features, and the Fourier coordinates use eight bands for horizontal and six for vertical inputs; separate volume and

surface output heads are used. Before standardization, a logarithmic reference profile is removed from U_y by subtracting $c \ln \max(z, 0.1\,\mathrm{m})$, with $c = -0.4494$ fitted by least squares on the training split, and $9.81z$ is added to the volume and surface pressure; both terms are restored after prediction. The normalization statistics are computed over all clean cases, including held-out cases, except those of U_y, which use the training estates only. Training samples its queries from all mesh points, and the loss is the mean of the volume-channel errors weighted by $(2, 1, 2, 1, 1)$ for $(U_x, U_y, U_z, p, \nu_t)$, plus the surface-pressure error.

D.4 Baselines and benchmarking protocol

Published baselines. On AirfRANS, rows other than our AB-UPT reproduction retain the values reported by their cited sources; the GINO row is Transolver's reproduction. SpiderSolver reports a stronger reproduction of Transolver than the row published by Transolver itself, which is the row we quote (Qi et al., 2025). On DrivAerML, published baseline rows retain their evaluation protocols and training budgets. IBO uses 400 epochs. These published comparisons are distinct from our controlled reproductions on TandemFoilSet and UrbanWind.

Shared settings. Our reproductions share data caches, splits, query pools, target normalization, loss reductions, evaluation code, epoch budgets, and checkpoint averaging. Training uses 400 epochs, or 398 on AirfRANS, with 32,000 queries per 2D case and 128,000 per 3D case, bf16 autocast, and gradient clipping at 1.0; AB-UPT is supervised on its anchor points, and SpiderSolver stratifies its queries between volume and surface points. On TandemFoilSet, the seed-42 runs of IBO, Transolver-3, Erwin and GAOT draw the same query rows for each case and epoch. The last quarter of epoch-end weights is averaged. Architectures and optimizer schedules remain model-specific; no validation sweep selects a configuration. The clip of 1.0 is the native value of Transolver-3, Erwin and AB-UPT; for SpiderSolver it replaces the effectively unbounded 10^6 of the released code, the one optimizer setting of a baseline that we changed, in the direction of greater stability. Appendix D.2 lists the training seeds of each row.

Transolver-3. Transolver and Transolver-3 share the Physics-Attention operator and differ in training- and inference-time scalability, so on the 10^5-point 2D meshes the two coincide; AirfRANS quotes the published Transolver results, and we reproduce Transolver-3. The 3D reproduction uses the released DrivAerML volume configuration: 16 layers, width 256, eight heads, and 64 slices (the paper's table lists 24 layers); AdamW with learning rate 10^{-3}, weight decay 0.01, and per-epoch cosine decay to 10^{-5}. TandemFoilSet uses the Transolver AirfRANS configuration: eight layers, width 256, eight heads, 32 slices, and Adam with the released one-cycle schedule. Our reproductions use a shared backbone with separate surface and volume heads, position and wall geometry as inputs, and a dataset-domain positional embedding. For TandemFoilSet, the two wall distances are combined by their minimum. Full-field inference accumulates slice statistics over chunks; the cost accounting includes the passes needed by the chosen inference schedule.

AB-UPT. Our reproduction uses width 192 and three heads, with a shared trunk followed by separate surface and volume paths. Each path traverses twelve blocks. The geometry branch is omitted because none of our datasets provides a geometry point cloud separate from the query points; this coordinates-only variant is the method as published, and the shared trunk uses a third split-attention block in place of the geometry block. Anchor counts are 16,384 volume and 16,384 surface on UrbanWind, 4,096 and 960 on TandemFoilSet (750 surface anchors on Race Car and Takeoff), and 4,096 and 840 on AirfRANS. An MLP adds the varying inflow condition on the 2D datasets. Optimization uses Lion at 5×10^{-5}, weight decay 0.05, 5% warm-up, and cosine decay to 10^{-6}. On AirfRANS, the reproduction shares IBO's split, normalization, loss, optimizer schedule and 398-epoch budget, and trains one case per GPU. Anchor keys and values are cached before chunked query decoding.

On TandemFoilSet and UrbanWind, accuracy averages five independent inference anchor draws per test case; AirfRANS uses one fixed draw per case. For the seed-42 runs, the draw-to-draw standard deviations for TandemFoilSet (volume, surface) are (0.0000, 0.0001), (0.0000, 0.0001), (0.0007, 0.0003), (0.0006, 0.0024), and (0.0000, 0.0001), in the main-table subset order. UrbanWind values are 0.0023, 0.0003, 0.0020, 0.0000, 0.0004, and 0.0000 in channel order. Values displayed as zero

are rounded at four decimal places. These statistics measure inference sampling, not variation across training seeds.

GAOT. GAOT is reproduced on TandemFoilSet only: at its released configuration, its radius gathering on a regular grid does not fit a three-dimensional scene of millions of points in the memory of a single GPU. The reproduction uses a uniform 64×64 latent layout, five processor blocks, and ball-based MAGNO encoding and decoding, with the width raised from the released 256 to IBO's 384. The released absolute radius is replaced by a cell-relative base radius of 1.05 latent spacings, so that every decoder query has neighbours; empty sites are retained within the token budget. The main-table run uses the baseline optimization schedule, with AdamW warming from 8×10^{-4} to 10^{-3} and subsequent decay over 400 epochs, against the 1,000 of the release. Table 12 compares the single-radius gathering with GAOT's multiscale variant at radii R and $1.5R$ under a shared training protocol, with the per-scale tokens averaged rather than combined by GAOT's attentional scale weights: the second radius changes the subset errors by at most 14% while enlarging the encoder graph severalfold, so the reproduction retains the single radius. The released three-radius configuration (R, $1.5R$, $2R$) exceeded the memory of a 40 GB A100 on these meshes.

Table 12: GAOT on TandemFoilSet with a single gathering radius R against two radii (R and $1.5R$), trained under a shared protocol; volume and surface MSE as in Table 5. The main-table GAOT row uses the published schedule instead.

	AoA 0°		AoA 5°		Random		Race Car		Takeoff	
Gathering	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.
Ball query, radius R	0.4302	1.6367	0.4484	1.3677	0.4837	0.8576	0.2083	0.1452	0.2682	0.8363
Ball query, radii R and $1.5R$	0.4242	1.6346	0.4545	1.3751	0.4418	0.7633	0.1796	0.1351	0.2610	0.8105

SpiderSolver. TandemFoilSet uses the AirfRANS configuration: five distance bands, four boundary clusters per body, eight layers, and width 256. The two bodies' clusters are built from training templates and aligned by optimal transport. UrbanWind uses ten distance bands and six boundary clusters. Building scenes have no shared template, so their clusters are defined by dominant surface-normal direction in the wind-aligned frame. Both use the released Adam/one-cycle schedule. The training-set median surface fraction determines the number of surface queries.

Erwin. TandemFoilSet uses width 64, ball size 256, encoder and decoder depths of six, 45° rotation, and eight neighbours, with AdamW at 10^{-3}, weight decay 10^{-5}, and 5% warm-up. UrbanWind uses the four-level cosmology hierarchy with depths 2/2/6/2, stride two, ball size 256, and widths 64–512, with AdamW at 5×10^{-4}. The released airfoil wrapper encodes only the first coordinate, at frequencies up to $2^{63}\pi$; both datasets instead use the Gaussian random Fourier coordinates of the same repository's ShapeNet wrapper, so that every coordinate is represented. TandemFoilSet adds inflow features; both datasets use separate volume and surface heads.

Whole-case coverage. IBO, GAOT, AB-UPT, and Transolver-3 use their case-level representations while decoding output chunks. SpiderSolver and Erwin are evaluated using stride chunks at training density, recomputing their representations within each chunk. All requested points are predicted once; repeated rows used for final-chunk padding are discarded.

D.5 Complexity and inference cost

This subsection derives the complexity of IBO and then explains the inference costs in Tables 3–5.

Complexity. Let K be the maximum number of encoder members or representatives per token, c_ψ the cost of their pointwise feature map, B the number of processor blocks, b the number of decoder stages and D_{dec} the decoder width. The encoder costs $C_{\mathrm{enc}} = O(LKc_\psi)$, the processor, B self-attention blocks of width D over the L latent points, costs $C_{\mathrm{proc}} = O\big(B(L^2D + LD^2)\big)$, and the decoder stages project the token keys and values once, at $C_{\mathrm{KV}} = O(bLD\, D_{\mathrm{dec}})$. None of these depends on the mesh size N; together they form the once-per-case cost $C_{\mathrm{case}} = C_{\mathrm{enc}} + C_{\mathrm{proc}} + C_{\mathrm{KV}}$

of Section 6.2. Each decoded point then reads its k nearest latent points through the b stages, so decoding m points costs

$$C_{\mathrm{IBO}}(m) = C_{\mathrm{case}} + O\big(bmkD_{\mathrm{dec}}^2\big), \tag{24}$$

with $m = Q$ sampled queries per case for a training step and $m = N$ for a full-field prediction; backpropagation changes only constant factors. At inference, N is multiplied by the constant k, not by the depth, width or token count of the network (Table 13).

Training-step accounting. In Eq. (24), c_ψ includes all dense layers of the member feature map. Pooling and region projection add $O(LKD + LD^2)$; with the implemented feature widths, the pooling term is absorbed by C_{enc} and the projection by C_{proc}. Each processor block contributes self-attention $O(L^2D)$ and projections and feed-forward layers $O(LD^2)$; registers have fixed count. C_{KV} never exceeds the order of C_{proc}, because $b \le B$ and $D_{\mathrm{dec}} \le D$ in all configurations. Each decoder stage forms query–neighbour features and projections at $O(mkD_{\mathrm{dec}}^2)$, with $m = Q$ per training step or $m = N$ per full-field prediction. The output head and fixed descriptor maps do not change the order, and backpropagation multiplies these forward costs by constant factors. A training step therefore scales with the query count Q rather than the mesh size: on DrivAerML it processes 128,000 queries per case, although a complete case has about 1.4×10^8 points. These are neural-computation costs; the cached mesh partition and feature preprocessing are outside Eq. (24). Table 15 reports the resulting peak training memory.

Cost model. Table 13 splits a full-field prediction into work done once per case and work repeated at each of the N output points. IBO's once-per-case part is the largest: on UrbanWind, full self-attention over $L = 16{,}384$ tokens accounts for 5.1 of the processor's 6.0 TFLOP. A token summarizes a region of about $N/L \approx 180$ mesh points rather than a single point, which is why IBO uses the width $D = 384$; the processor dominates this once-per-case cost. IBO's per-point part is small: each query passes through a decoder of width D_{dec} and reads its k nearest latent points, independently of L, B and D. In contrast, every output point of AB-UPT traverses twelve blocks, each attending to $M = 16{,}384$ anchors, and every point of Transolver-3 traverses all 16 layers, each with slice assignment and de-slicing over $S = 64$ slices. On UrbanWind these amount to 2.0, 161 and 25.6 MFLOP per point.

Table 13: Cost of one full-field prediction, split into work done once per case and work per output point. IBO: L latent points of width D, B processor blocks, K encoder members per latent point with feature-map cost c_ψ, and b decoder stages of width D_{dec} reading k neighbours. AB-UPT: M anchors per branch and B_A blocks of width D_A. Transolver-3: B_T layers of width D_T with S slices. [a] Single pass; its released DrivAerML inference schedule recomputes earlier layers, about 183 MFLOP per point.

			MFLOP per point	
Model	Once per case	Per output point	UrbanWind	DrivAerML
IBO	$O\big(LKc_\psi + B(L^2D + LD^2)\big)$	$O(bkD_{\mathrm{dec}}^2)$	2.0	18.6
AB-UPT	$O\big(B_A(M^2D_A + MD_A^2)\big)$	$O\big(B_A(MD_A + D_A^2)\big)$	161	161
Transolver-3	$O(B_T S^2 D_T)$	$O\big(B_T(D_T^2 + SD_T)\big)$	25.6	25.6[a]

Measurement. The latencies in Tables 4 and 5 average the wall-clock time of all test cases on one A100 GPU, after one uncounted warm-up case, with bf16 inference. Disk reads and the cached preprocessing, namely IBO's partition, geometric descriptors and neighbour indices, are excluded. Host-side collation, AB-UPT's anchor sampling, Erwin's tree construction inside the forward pass, and host–device transfers are included. Operations are counted at the executed chunk shapes, with one multiply–accumulate counted as two FLOPs; compiled modules are metered on an eager copy.

Table 14 times one UrbanWind test case in the same setting (A100-SXM4-40GB, PyTorch 2.6, CUDA graphs). The case is the one whose point count is closest to the test-set mean: 2,838,911 volume and 73,114 surface points, against a test-set range of 1.7–4.6×10^6 points. AB-UPT decodes its surface and volume queries in separate chunk loops. Each value is the median of ten runs after three warm-up runs, and the ten runs differ by less than 3%. Every stage ends with a GPU synchronization; the stage sums differ from the unsynchronized end-to-end times by less than 0.4%, and the three end-to-end

times lie within 5% of Table 4. Peak training memory (Table 15) is measured after compile warm-up and a fixed number of updates.

Table 14: Stage timing of one UrbanWind test case on one A100 (ms, median of ten runs). Once per case: IBO's encoder (2.1 ms) and processor (48.2 ms), AB-UPT's anchor encoder, and Transolver-3's slice attention. Per output point: IBO's decoder, AB-UPT's query decoder, and Transolver-3's embedding (19.9 ms), slice statistics (671.5 ms), de-slicing and feed-forward layers (1,017.2 ms) and output heads (14.1 ms). Per-point values divide the per-point stage by the decoded rows, including chunk padding; throughput is the stage's operations divided by its time.

	IBO	AB-UPT	Transolver-3
Once per case	50.3	39.6	3.9
Per output point	263.4	3,011.2	1,722.7
Host-side collation	9.8	320.8	40.0
Host–device transfers	29.0	14.0	15.5
Sum of stages	352.5	3,386.6	1,782.0
End to end, this case	351.5	3,380.6	1,776.0
Table 4, mean over 160 cases	364	3,246	1,771
MFLOP per point	2.0	161	25.6
Time per point (ns)	84	1,021	585
Per-point throughput (TFLOP/s)	24	158	44

Why IBO is faster. All three models spend most of their time in the per-point stage, 75%, 89% and 97% of the stage sum for IBO, AB-UPT and Transolver-3, and this stage separates them: IBO decodes a point in 84 ns, AB-UPT in 1,021 ns and Transolver-3 in 585 ns. IBO's once-per-case cost is the largest, 50.3 ms and almost entirely the processor; in the UrbanWind configuration it is recovered after about 10^4 output points relative to AB-UPT and about 10^5 relative to Transolver-3, and it is 14% of IBO's time on this case. The latency ratios of Table 4, $8.9\times$ and $4.9\times$, are smaller than the operation ratios, $39.5\times$ and $6.2\times$, because the stages run at different hardware efficiency.

DrivAerML. The operation counts of Table 3 use the same cost model for the full field of approximately eight million surface and 130 million volume points. AB-UPT is configured as timed above, giving 161 MFLOP per point and 22.2 PFLOP per case. IBO uses two decoder stages of width 384 with $k = 16$ neighbours, giving 18.6 MFLOP per point; encoding and the twelve-block processor over 32,768 tokens add approximately 0.02 PFLOP, for 2.6 PFLOP per case. A single Transolver-3 pass costs only $1.4\times$ more per point than IBO, but its exact slice statistics require the features of all points of one layer before the next layer starts. This working set grows linearly with N: it is 5.7 GiB for the timed UrbanWind case, extrapolating to about 270 GiB at 1.4×10^8 points. IBO and AB-UPT instead decode fixed-size chunks against a fixed set of latent points or anchors, so their memory does not grow with N. Transolver-3's released inference schedule therefore recomputes the preceding layers chunk by chunk, for a total of about seven times a single pass, 25.3 PFLOP. Because stage efficiencies differ, we report operation counts on DrivAerML and do not infer wall-clock time from them.

Table 15: Peak training memory on one A100 (GiB). The dash means that GAOT does not fit UrbanWind on one A100 at its released configuration.

Model	TandemFoilSet	UrbanWind
GAOT	21.11	–
Erwin	6.90	13.32
SpiderSolver	3.21	6.69
Transolver-3	4.34	16.54
AB-UPT	1.98	2.78
IBO	14.55	9.29

D.6 ADDITIONAL ABLATIONS

Uniform-grid layouts. The uniform-grid rows of the layout ablation (Section 6.3) replace the tree by a fixed grid with the same number of cells L, expressed as the same tree structure with split planes at cell midpoints, and place each latent point at its cell centre. On AirfRANS and TandemFoilSet the grid has 64×64 cells over the domain; a cell with fewer points than the member cap is filled with the mesh points nearest to its centre, so an empty cell becomes a ball of the nearest points around its centre. On UrbanWind the grid has $32 \times 32 \times 16$ cells over the domain, and empty cells are handled as the unused token slots of Appendix C.1. On DrivAerML the grid has 32^3 cells over the crop box, and seven further levels define the subregion representatives within each cell; a cell with fewer points than subregions repeats its points, and every representative of an empty cell sits at the cell centre. The rotary scales are re-derived for the grid extent. The equal-count row on UrbanWind sets $w_i \equiv 1$ in the weighted pipeline.

Error and read-out resolution by wall distance. Figure 4 resolves the layout ablation by distance to the walls, and Table 16 lists the underlying values for every benchmark, TandemFoilSet subset and UrbanWind channel. For each test case, the volume nodes are sorted by wall distance and split into three equal-count bins; surface nodes are left out, and the bin edges are averages over cases. The wall distance is the minimum over the two bodies on TandemFoilSet and the distance to the nearest building on UrbanWind. On AirfRANS and TandemFoilSet the error is the standardized volume MSE within the bin, so the unbinned value is the mean of the three bins. On DrivAerML and UrbanWind it is the relative L_2 error in physical units with the bin's error norm divided by the truth norm over all volume points of the case, so that near^2+mid^2+far^2 = all^2 for every case. R_k is the mean distance from a volume point to the k latent points its decoder reads ($k = 16$ on DrivAerML and 8 elsewhere), computed from each run's own latent layout. Values are averaged over the test cases, and ratios are formed from the averages. Both layouts are scored on IBO's nodes against the same reference, and the recomputed unbinned errors reproduce each run's stored metrics to a relative deviation below 5×10^{-7}. Both layouts are seed-42 runs, so the unbinned IBO values can differ slightly from the multi-seed means of the main tables.

Table 16: Error and read-out resolution R_k in three equal-count wall-distance bins, for IBO and the uniform grid of the same budget (seed-42 runs; Appendix D.6). Bin edges are case averages. TandemFoilSet R_k is in 10^{-2} mesh units. UrbanWind values are relative L_2 fractions at the printed precision of four decimals.

	IBO				Uniform grid			
	near	mid	far	all	near	mid	far	all
AirfRANS (edges 1.81 mm and 87.1 mm)								
Volume MSE (10^{-4})	3.45	5.51	19.0	9.33	16.3	17.7	42.7	25.6
R_k (mm)	2.64	8.85	89.8	33.8	71.0	70.5	73.0	71.5
TandemFoilSet, Cruise AoA 0° (edges 1.10 and 3.73)								
Volume MSE (10^{-4})	3.21	0.254	0.162	1.21	14.1	1.22	0.487	5.27
R_k	8.86	20.8	51.8	27.2	36.3	36.3	36.7	36.4
TandemFoilSet, Cruise AoA 5° (edges 1.10 and 3.73)								
Volume MSE (10^{-4})	7.32	1.25	1.11	3.23	14.4	1.75	1.67	5.94
R_k	8.86	20.8	51.8	27.2	36.3	36.3	36.7	36.4
TandemFoilSet, Cruise Random (edges 0.0137 and 0.299)								
Volume MSE (10^{-4})	12.6	2.70	0.473	5.27	132	22.4	3.02	52.4
R_k	0.734	3.24	32.3	12.1	35.3	35.2	35.5	35.3
TandemFoilSet, Race Car (edges 0.00645 and 0.108)								
Volume MSE (10^{-4})	177	148	60.8	129	420	257	79.1	252
R_k	0.496	1.47	19.7	7.22	23.7	23.8	24.1	23.9
TandemFoilSet, Takeoff (edges 0.757 and 2.79)								
Volume MSE (10^{-4})	4.55	0.742	0.294	1.86	12.6	1.39	0.322	4.76
R_k	6.70	13.2	38.8	19.6	27.1	27.4	27.2	27.3
DrivAerML (edges 0.224 cm and 1.26 cm)								
p, relative L_2 (%)	2.53	1.99	1.24	3.45	4.64	3.66	2.38	6.37
U, relative L_2 (%)	3.34	3.18	2.32	5.18	5.35	4.85	3.96	8.25
R_k (cm)	2.32	3.66	16.3	7.43	86.0	88.8	95.8	90.2
UrbanWind (edges 71.8 m and 164.9 m)								
U_x	0.1166	0.0408	0.0153	0.1261	0.1657	0.0559	0.0204	0.1785
U_y	0.0180	0.0068	0.0030	0.0198	0.0248	0.0094	0.0039	0.0272
U_z	0.1244	0.0397	0.0153	0.1329	0.1755	0.0538	0.0201	0.1868
p	0.0004	0.0001	0.0000	0.0004	0.0007	0.0001	0.0001	0.0007
ν_t	0.0217	0.0191	0.0151	0.0331	0.0276	0.0236	0.0175	0.0410
R_k (m)	15.4	21.3	25.0	20.6	22.3	22.3	22.6	22.4

The uniform grid's R_k varies by less than 4% across the three bins on every benchmark except DrivAerML (11%), as expected from a layout that ignores the walls. IBO's R_k instead grows by more than an order of magnitude from the near to the far bin on the refined 2D meshes (34× on AirfRANS, 44× on Cruise Random and 40× on Race Car), by 7× on DrivAerML, and by 1.6× on UrbanWind, whose nearly regular mesh leaves the importance weight less room. Near the walls, IBO's read-out is 48 times finer than the grid's on the two high-Reynolds-number TandemFoilSet subsets and about 4 times finer on the three subsets at $Re = 500$; on the high-Reynolds-number subsets it remains finer than the grid's even in the far bin. The error ratio rises monotonically from the near to the far bin on Cruise Random (0.10 to 0.16), Race Car (0.42 to 0.77) and Takeoff (0.36 to 0.91), as on AirfRANS. On Cruise AoA 0° the largest gain is in the mid bin, and on Cruise AoA 5° the far-bin ratio (0.67) lies slightly below the mid-bin one (0.72). On UrbanWind, the near bin carries most of the velocity error of both layouts, for example 0.117 of IBO's 0.126 for U_x, whereas the ν_t error is spread more evenly across the bins. The ratio rises from the near to the far bin for U_x, U_y, U_z and ν_t; the pressure contributions lie at the printed precision and are not resolved. On DrivAerML, the gain is nearly uniform across the bins for pressure (ratios 0.52 to 0.55) and velocity (0.58 to 0.66). Finer local read-out is associated with the near-wall gains, but the error ratio need not follow R_k monotonically, because token content and the learned reconstruction also shape the error.

All TandemFoilSet subsets. Table 17 extends the layout comparison of Table 6 to all five subsets. The pattern is unchanged: the adaptive layout improves both metrics of every subset, by almost an order of magnitude (9×) on the Cruise Random volume field.

Table 17: Layout ablation on all five TandemFoilSet subsets (volume and surface MSE as in Table 5). The adaptive, equal-count row is the main-table configuration and reports its five-seed mean; the uniform-grid row is a single run.

	Cruise AoA 0°		Cruise AoA 5°		Cruise Random		Race Car		Takeoff	
Latent layout	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.
Uniform grid	0.0005	0.0099	0.0006	0.0075	0.0052	0.0409	0.0252	0.0552	0.0005	0.0049
Adaptive, equal count	**0.0001**	**0.0009**	**0.0003**	**0.0020**	**0.0006**	**0.0027**	**0.0132**	**0.0141**	**0.0002**	**0.0009**

Width sensitivity on DrivAerML. Halving the width from 384 to 192 at the full latent budget raises every field error by 0.11–0.14 points of relative L_2 and lowers the C_D R^2 by 0.009, with the C_L R^2 unchanged (Table 18).

Table 18: DrivAerML: sensitivity to the token width; metrics as in Table 3. The default row is the mean over three training seeds; the halved width is a single run.

Configuration	p_s	τ	p	U	C_D R^2	C_L R^2
Default ($L = 32{,}768$, width 384)	3.78	7.12	3.44	5.14	0.967	0.980
Width halved (192)	3.90	7.24	3.58	5.25	0.958	0.980

D.7 QUALITATIVE RESULTS

The following figures show one illustrative case per benchmark, together with the available coefficient-parity plots. The AirfRANS case is `airFoil2D_SST_74.044_-0.131_0.701_3.971_0.0_15.683` ($U_\infty = 74.0$ m/s, $\alpha = -0.13°$); the TandemFoilSet case is Cruise Random `p2.0191` ($|U| = 58.6$ m/s, AoA $= 3.3°$, $Re = 4.0 \times 10^6$); DrivAerML is test case 350; the UrbanWind case is estate 0562b016 (16 buildings) with northerly wind. These panels are meant to illustrate observed differences rather than typical errors; aggregate test performance is reported in Section 6.1. Within each comparison, residuals are prediction minus reference and share a colour scale across the models.

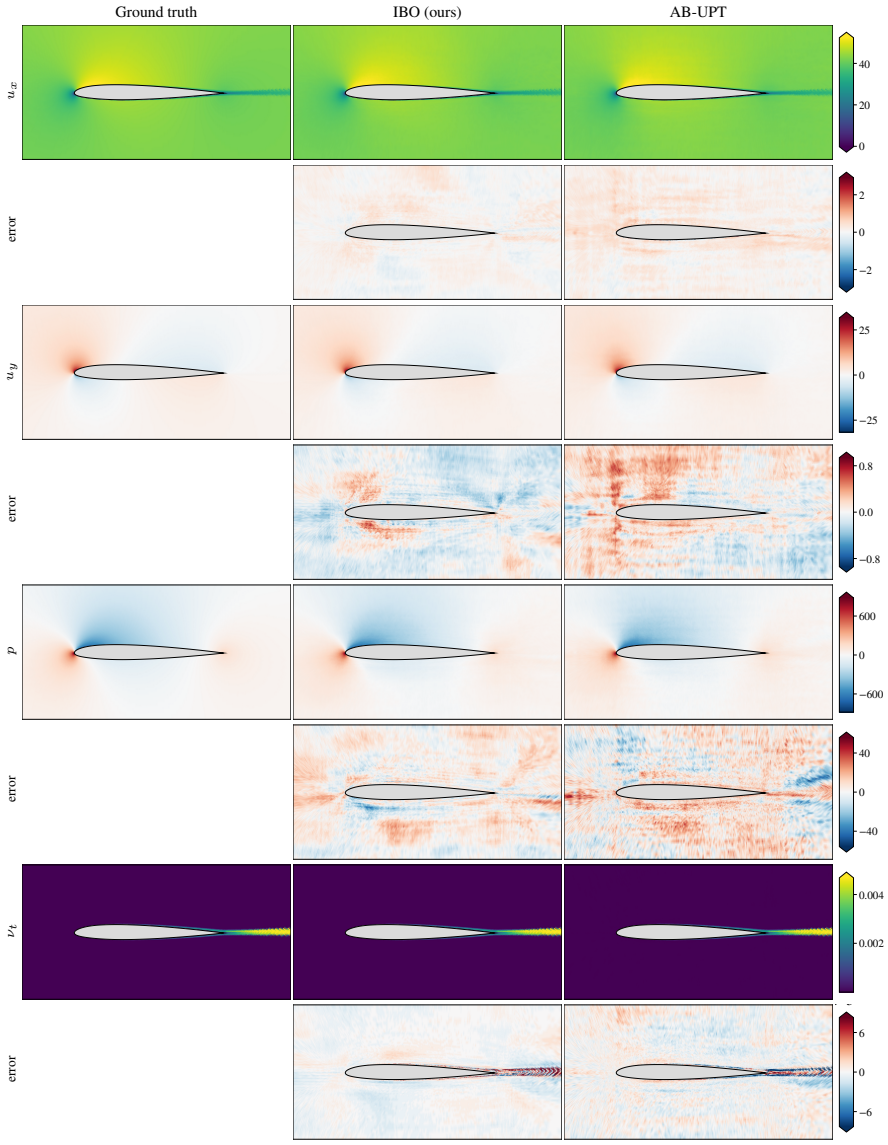


Figure 6: AirfRANS: velocity, pressure, and turbulent-viscosity predictions and residuals for one airfoil case.

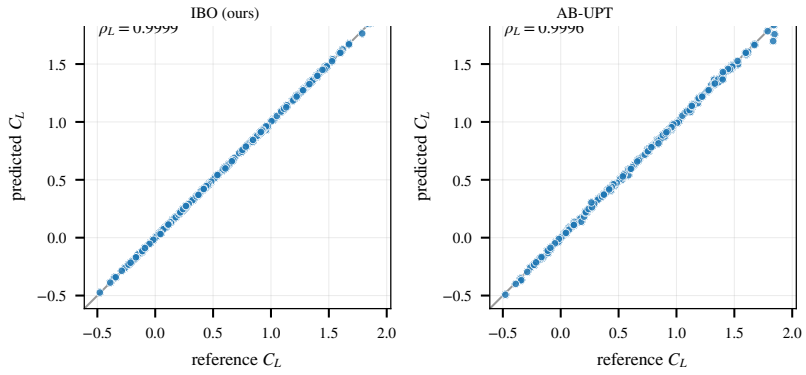


Figure 7: AirfRANS lift-coefficient parity: IBO and AB-UPT on the test set.

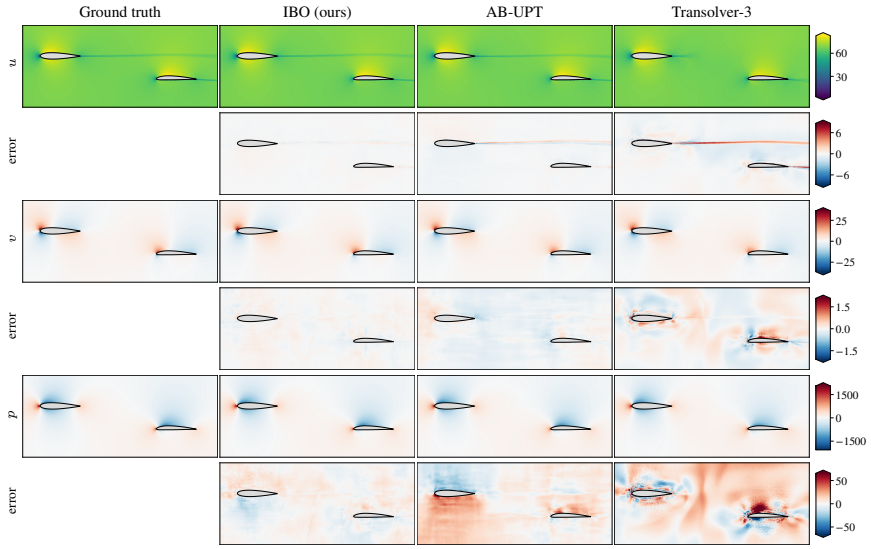


Figure 8: TandemFoilSet: velocity and pressure predictions and residuals for one Cruise Random case.

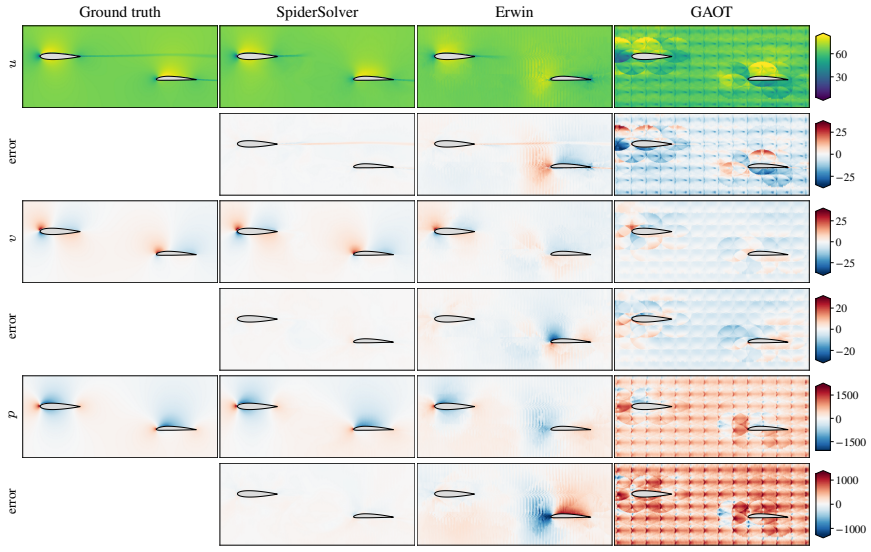


Figure 9: TandemFoilSet: the remaining baselines on the same case as Figure 8.

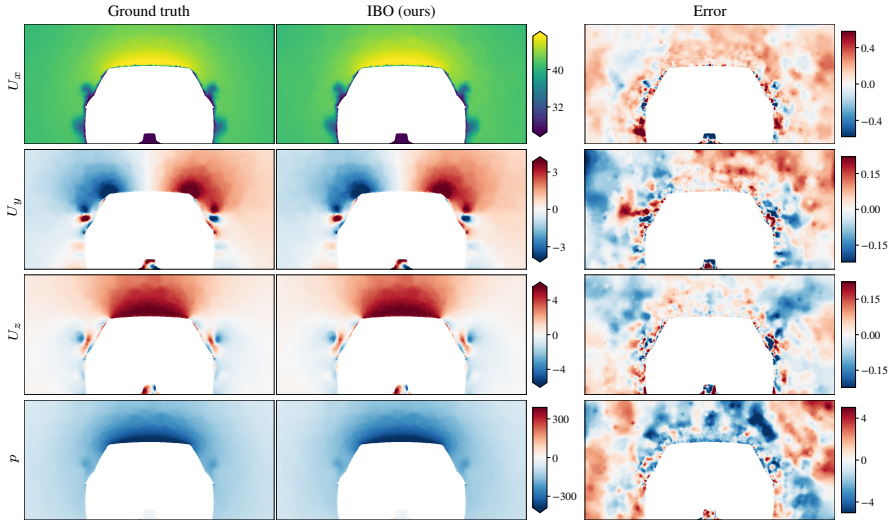


Figure 10: DrivAerML: volume predictions and residuals on a transverse plane of test case 350.

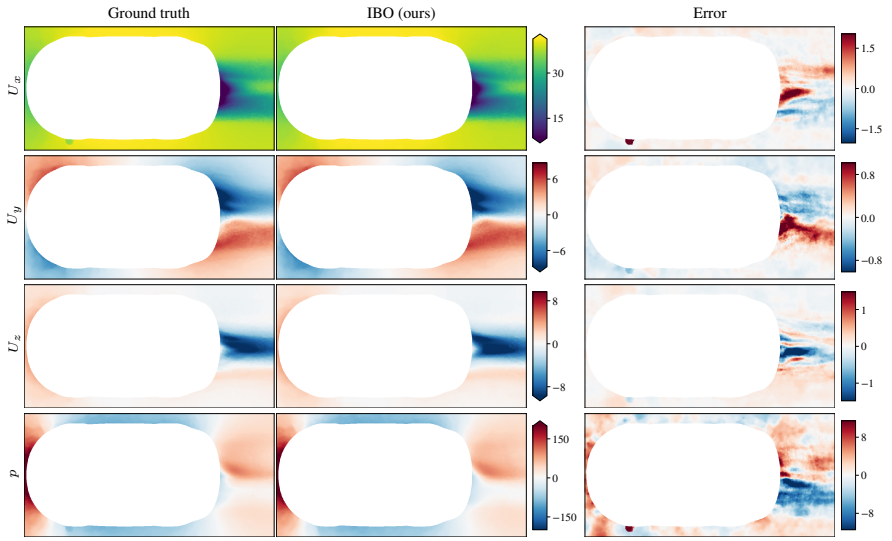


Figure 11: DrivAerML: volume predictions and residuals on a horizontal plane of the same case.

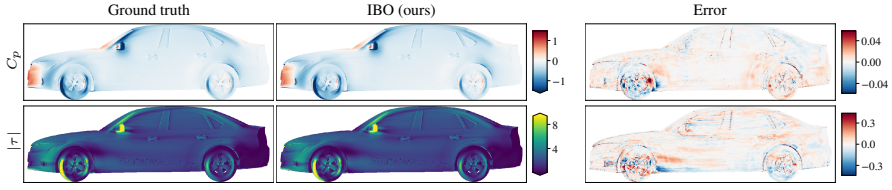


Figure 12: DrivAerML: surface pressure and wall shear stress, side view.

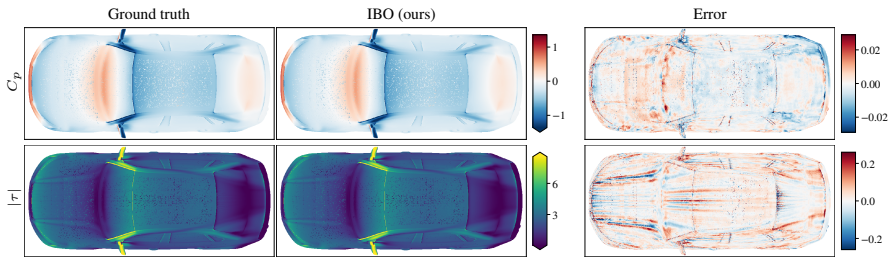


Figure 13: DrivAerML: surface pressure and wall shear stress, top view.

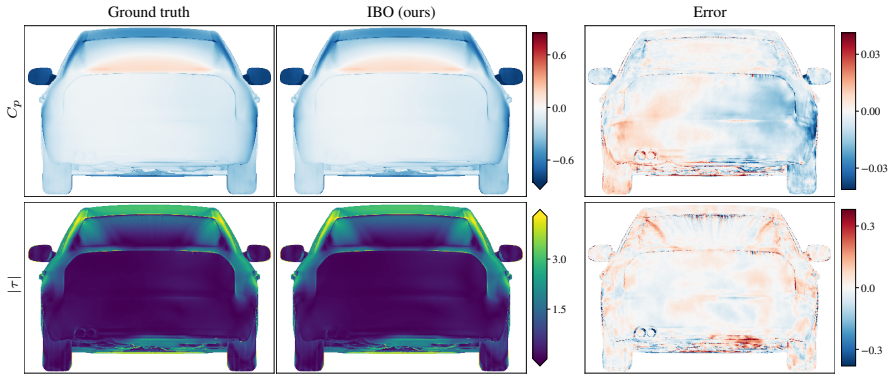


Figure 14: DrivAerML: surface pressure and wall shear stress, rear view.

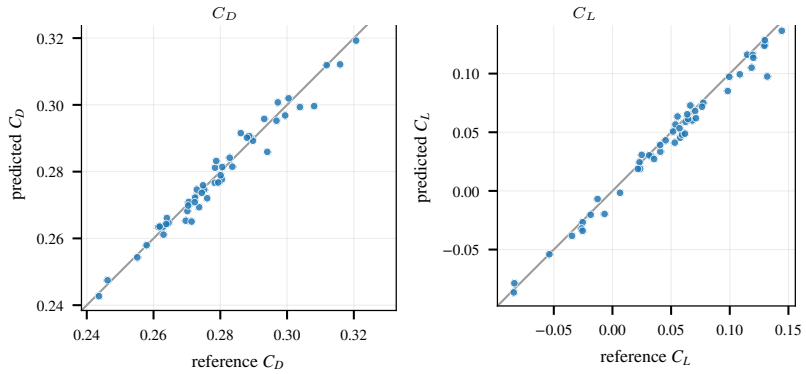


Figure 15: DrivAerML: IBO drag- and lift-coefficient parity over 50 test cases.

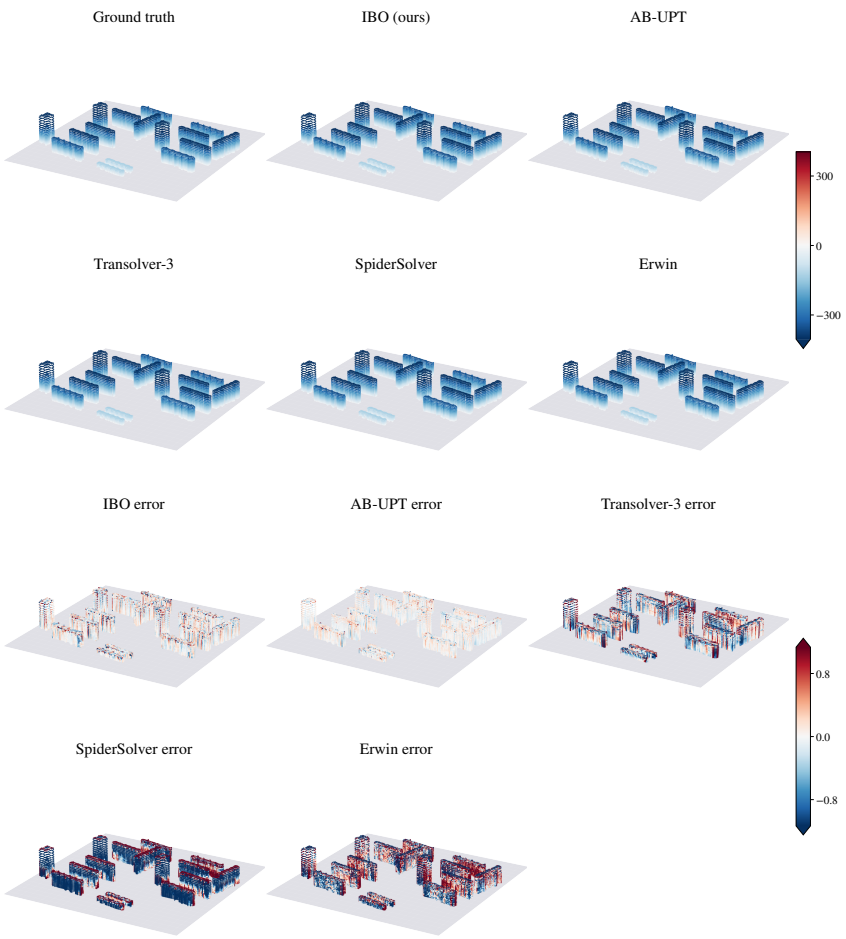


Figure 16: UrbanWind: building-surface pressure predictions and residuals for one estate.

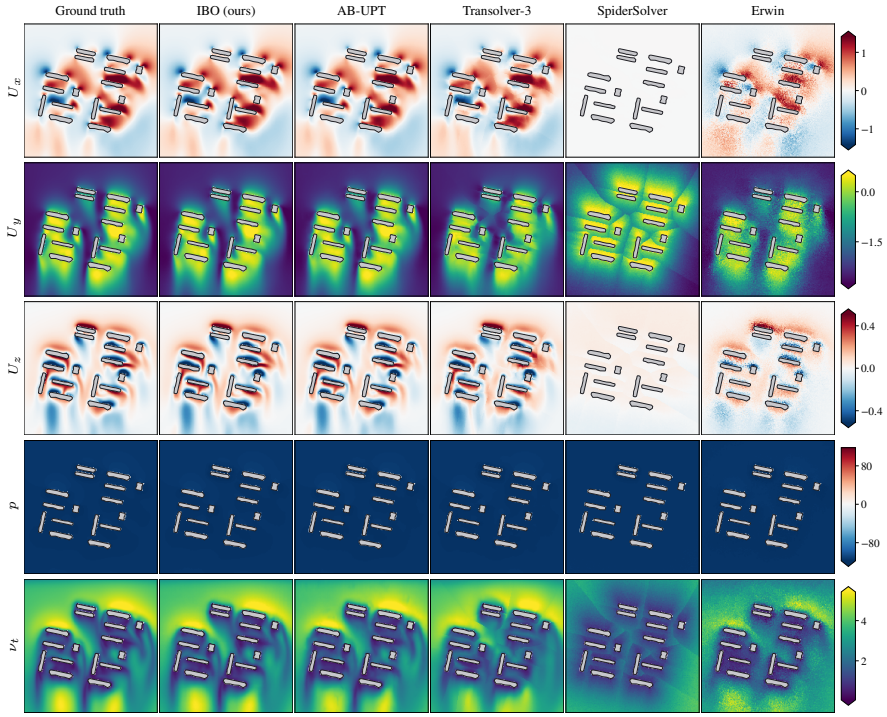


Figure 17: UrbanWind: all volume fields on a horizontal slice through the selected estate.

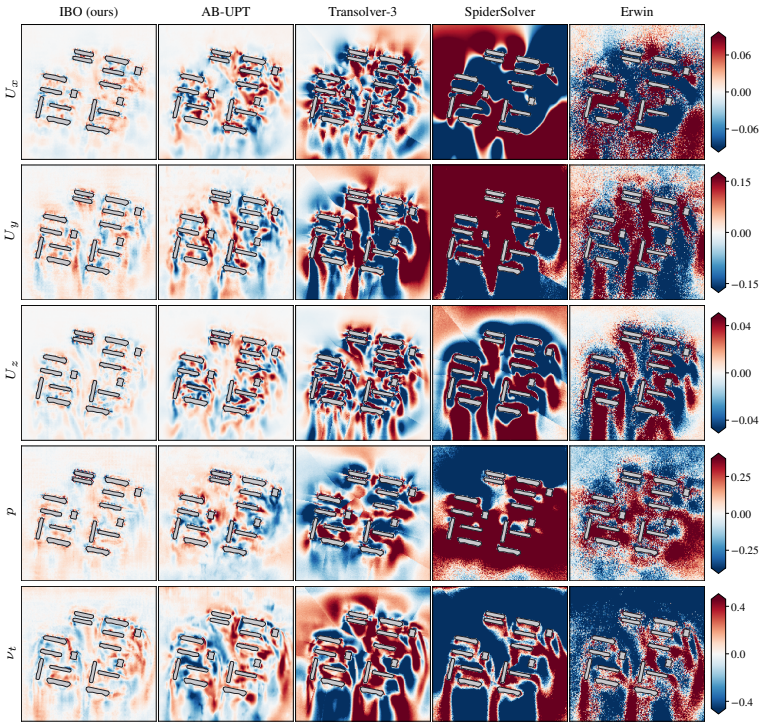


Figure 18: Residuals corresponding to Figure 17.

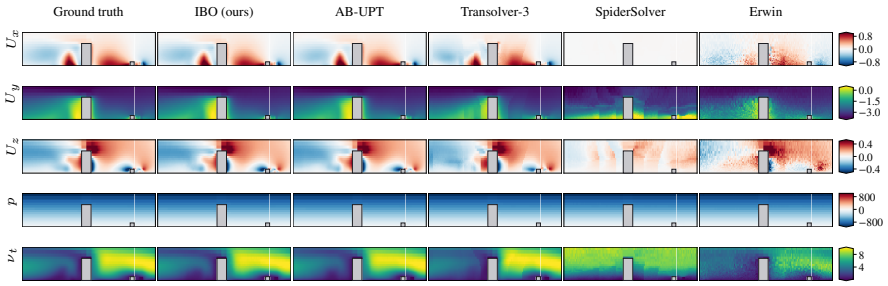


Figure 19: UrbanWind: all volume fields on a vertical slice through the selected estate.

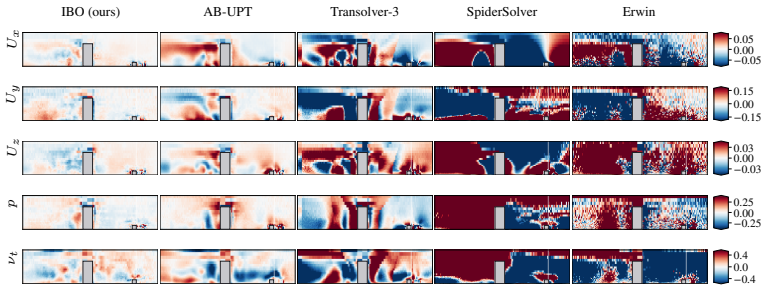


Figure 20: Residuals corresponding to Figure 19.