OPERATOR LEARNING WITH POINT-SET LOCAL TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling neural operators to high-resolution physical fields on irregular geometries requires efficient processing of large point sets. We introduce the Point-Set Local Transformer (PLT), which combines quadrature-weighted local attention with sampled global communication to capture both local structure and long-range dependencies – without dense point-to-point attention. Its separation of source processing and query decoding enables memory-efficient prediction at independently specified output locations. We evaluate PLT on several benchmarks, including HiLiftAeroML, where full-field aircraft predictions cover 126–151 million native surface points. PLT provides accurate full-field predictions for high-lift aircraft configurations, achieving a 6.47% mean relative error in the pre-stall benchmark. On the Plasticity benchmark, PLT achieves the lowest prediction error among all considered methods, improving the state-of-the-art by 41.46%. On the SuperWing benchmark, PLT achieves a 2.91% mean relative error for the surface pressure coefficient, while chunked full-field inference reduces peak memory by 45.6%. In scaling tests, PLT supports training with 4.19 million input points using 41.4 GiB on a single GPU and achieves the lowest peak memory among the evaluated implementations. These results demonstrate accurate physical-field prediction and memory-efficient evaluation of PLT at industrial resolution.

1 INTRODUCTION

Neural operators learn mappings between physical fields and provide surrogates for repeated numerical simulation. Fourier neural operators (Li et al., 2021) and their geometry-aware extensions (Li et al., 2023c) demonstrate the potential of this approach across discretizations. High-resolution applications nevertheless require processing large numbers of spatial samples, often distributed nonuniformly over complex geometries. Scaling to these settings requires architectures that accommodate irregular inputs while controlling the memory and computation needed for field prediction.

Attention provides content-dependent interactions, but dense pairwise attention has quadratic arithmetic cost in the number of points. Transolver (Wu et al., 2024a) reduces this cost by aggregating point features into learned physical slices, applying attention between slices, and mapping the updated features back to points. Transolver++ (Luo et al., 2025) improves slice differentiation and memory-efficient execution to handle million-point geometries. An alternative is to perform prediction in a compressed latent space. UPT (Alkin et al., 2024) combines sampled supernodes with Perceiver pooling and query-based decoding, while AeroJEPA (Giral et al., 2026) learns predictive geometry and flow embeddings with an optional continuous field decoder. These approaches establish the value of separating large discretizations from the representation used for global processing.

Local attention offers a complementary means of reducing spatial interaction costs. Image Transformer (Parmar et al., 2018) restricts attention to local neighborhoods, and Swin (Liu et al., 2021) combines local windows with cross-window communication. P3D (Holzschuh et al., 2026) and AeroTransformer (Yang et al., 2026) adapt window-based processing to physical fields on structured layouts. For unordered point sets, PointNet (Qi et al., 2017a) provides permutation-invariant aggregation, and PointNet++ (Qi et al., 2017b) introduces sampled centers and radius-based grouping. We build on this separation of geometric grouping and learned aggregation to define local attention directly on physical neighborhoods, without requiring a regular grid or a fixed patch layout.

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.

We introduce the *Point-Set Local Transformer* (PLT), which utilizes local point-to-neighborhood cross-attention. Selected points (centers) first aggregate information from spherical neighborhoods in the input point set, while a small set of global tokens pooled from sampled points provides field-wide information. Quadrature weights enter both local attention and global pooling, combining spatially resolved aggregation with long-range communication without dense point-to-point attention. The resulting representation remains anchored to physical center locations while supporting independently specified outputs.

The separation of input encoding and query decoding enables memory-efficient execution. Since decoder queries do not interact with one another, a single encoded representation can be reused across query chunks, reducing peak inference memory while preserving the mathematical operations of one-pass inference.

Our evaluation covers standard PDE benchmarks, large-scale aerodynamic prediction, and controlled computational scaling. PLT achieves the lowest error among the compared methods on the Plasticity benchmark and combines accurate prediction with memory-efficient query chunking on the SuperWing benchmark. On the HiLiftAeroML dataset, it enables full-field aircraft prediction at more than 100 million native surface output points with low peak GPU memory. Under the model and execution policies in Section 4, PLT achieves the lowest measured peak memory among the compared implementations at shared successful scales. Detailed comparisons with related architectures are reported in Appendix A. Code and pretrained checkpoints are provided in the supplementary materials and will be made publicly available on GitHub.

2 Method Overview

We learn a map from geometry, an input field, and conditions to an output field. Both input and output fields may depend on time $t \in [0, T]$. The input field may represent material coefficients, initial data, and/or observed states, while the conditions specify task-dependent physical parameters. Geometry is represented by collocation points $\boldsymbol{x} \in \Omega \subset \mathbb{R}^d$ within the spatial d-dimensional domain Ω. The input field $a\colon \Omega \times [0,T] \to \mathbb{R}^{d_{\text{in}}}$ has values $a(\boldsymbol{x}, t)$, physical conditions are collected in the vector $\boldsymbol{c}$, and the output field $u : \Omega \times [0,T] \to \mathbb{R}^{d_{\text{out}}}$ has values $u(\boldsymbol{x}, t; a, \Omega, \boldsymbol{c})$, where the semicolon denotes parameterization. PLT learns an operator $\mathcal{G}_{\boldsymbol{\theta}}$, parameterized in trainable model parameters $\boldsymbol{\theta}$, to approximate the output field as follows:

$$u(\boldsymbol{x}, t; a, \Omega, \boldsymbol{c}) \approx \mathcal{G}_{\boldsymbol{\theta}}(a, \Omega, \boldsymbol{c})(\boldsymbol{x}, t) \in \mathbb{R}^{d_{\text{out}}}. \tag{1}$$

Consider a dataset of collocation points $\mathcal{X} = \{\boldsymbol{x}_k\}_{k=1}^N \subset \Omega$ with associated inputs $\boldsymbol{a}_{kn} = a(\boldsymbol{x}_k, t_n) \in \mathbb{R}^{d_{\text{in}}}$ and supervised outputs $\boldsymbol{u}_{know} = u(\boldsymbol{x}_k, t_n; \boldsymbol{a}_k, \Omega_o, \boldsymbol{c}_w) \in \mathbb{R}^{d_{\text{out}}}$, where $t_n \in [0,T]$ denotes the time at time step $0 \le n < n_{\text{steps}}$. Each collocation point $\boldsymbol{x}_k$ is assigned a positive weight w_k, $\boldsymbol{w} \in \mathbb{R}^N$, which is related to a volume or surface measure on Ω; see Appendix C for details. For training, we sample $N_{\text{s}} \le N$ points from $\mathcal{X}$, hereinafter referred to as source points forming the subset $\mathcal{X}_{\text{s}} = \{\boldsymbol{x}_i\}_{i \in \mathcal{I}_{\text{s}}} \subseteq \mathcal{X}$ with index set $\mathcal{I}_{\text{s}} \subseteq \{1, \dots, N\}$.

Downsampling. The first step for the PLT implementation of $\mathcal{G}_{\boldsymbol{\theta}}$, Equation 1, is to downsample source points onto N_{c} center points $\mathcal{X}_{\text{c}} = \{\boldsymbol{x}_i\}_{i \in \mathcal{I}_{\text{c}}}$, where $\mathcal{I}_{\text{c}} \subseteq \{1, \dots, N_{\text{s}}\}$. As in PointNet++ (Qi et al., 2017b), center points are selected via Farthest Point Sampling (FPS) and radius grouping is done as follows: The geometric ball of radius r for a selected center $i \in \mathcal{I}_{\text{c}}$ defines the local neighborhood $\mathcal{N}_i^r(\boldsymbol{x}_i, \boldsymbol{x}_j, r) = \{j \in \mathcal{I}_{\text{s}} \mid \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \le r\}$. Via an optional cap $|\mathcal{N}_i^r| \le K$, only the K closest neighbors to the center are retained. Balls are allowed to overlap and may contain different numbers of neighbors. The complete connectivity $\mathcal{N} = \{\mathcal{N}_i^r\}_{i=1}^{N_{\text{c}}}$ is established via all N_{c} local neighborhoods. Each center $i \in \mathcal{I}_{\text{c}}$ provides one query, while its neighbors $j \in \mathcal{N}_i^r$ provide keys and values, yielding at most $1 \times K$ cross-attention rather than $K \times K$ neighborhood self-attention.

Source lifting. An affine lift maps each pair $(\boldsymbol{x}_i, \boldsymbol{a}_i)$, $i \in \mathcal{I}_{\text{s}}$, to a source state $\boldsymbol{s}_i^{(0)} = \phi_{\text{s}}([\boldsymbol{x}_i, \boldsymbol{a}_i]) + p(\boldsymbol{x}_i) \in \mathbb{R}^C$, where $C = Hd_v$ is the channel width, H is the number of attention heads and d_v is the value dimension per head. The query and key dimension per head is denoted by d_k. (Note that the total query/key width $D = Hd_k$ can differ from the value and hidden width $C = Hd_v$). ϕ_{s} is a trainable embedding, and p is an optional fixed positional embedding. Stacking N_{s} source states $\boldsymbol{s}_i^{(0)}$ yields $\boldsymbol{S}^{(0)} \in \mathbb{R}^{N_{\text{s}} \times C}$. As visualized in Figure 1, PLT implements Equation 1 by mapping source

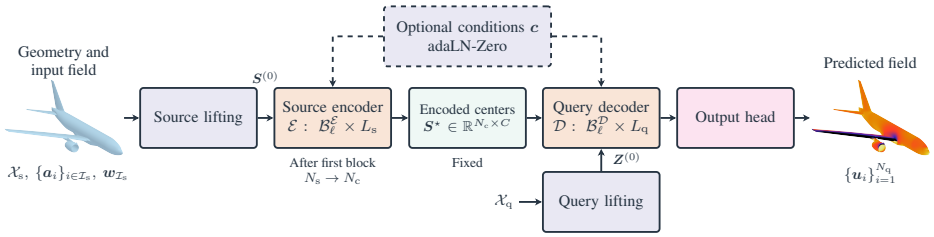


Figure 1: PLT source encoder $\mathcal{E}$ (composed of L_s source blocks $\mathcal{B}_\ell^{\mathcal{E}}$) and query decoder $\mathcal{D}$ (composed of L_q query blocks $\mathcal{B}_\ell^{\mathcal{D}}$). The encoder retains updated centers $i \in \mathcal{I}_c$ after its first block. Dashed elements indicate optional conditioning, and Figure 2 details the source and query blocks.

states $\boldsymbol{S}^{(0)}$ via a source encoder $\mathcal{E}$ and a query decoder $\mathcal{D}$ to the output. The specific encoder and decoder architectures are outlined below.

2.1 Source encoder

The source encoder is composed of L_s attention blocks $\mathcal{B}_\ell^{\mathcal{E}}$, $\ell = 1 \dots L_s$. As visualized in Figure 2, each block combines both local attention and global mixing paths as well as optional conditioning gates. Cross-attention is only performed in the local neighborhood, while in the global path, Efficient Attention (Shen et al., 2021) is employed.

While the first block inputs all N_s sampled source states $\boldsymbol{S}^{(0)} \in \mathbb{R}^{N_s \times C}$, only centers are retained in subsequent blocks, the input of which is $\boldsymbol{S}^{(\ell-1>0)} \in \mathbb{R}^{N_c \times C}$. Each block first applies layer normalization. For problems conditioned on $\boldsymbol{c}$, condition-dependent scaling and shifting are incorporated through adaLN-Zero. We denote the resulting normalized, optionally modulated states by $\bar{\boldsymbol{S}}^{(\ell-1)}$.

Local attention $\mathcal{A}_\ell$. Local aggregation is learned through cross-attention between centers and their local neighborhoods. For one head, the projections are $\boldsymbol{q}_i = \boldsymbol{W}_q \bar{\boldsymbol{s}}_i \in \mathbb{R}^{d_k}$ (with center points $i \in \mathcal{I}_c$), as well as $\boldsymbol{k}_{ij} = \boldsymbol{W}_k \bar{\boldsymbol{s}}_{ij} \in \mathbb{R}^{d_k}$ and $\boldsymbol{v}_{ij} = \boldsymbol{W}_v \bar{\boldsymbol{s}}_{ij} \in \mathbb{R}^{d_v}$ (with the centers' local neighbors $j \in \mathcal{N}_i$). The local aggregation produces the head output $\boldsymbol{o}_i \in \mathbb{R}^{d_v}$ as

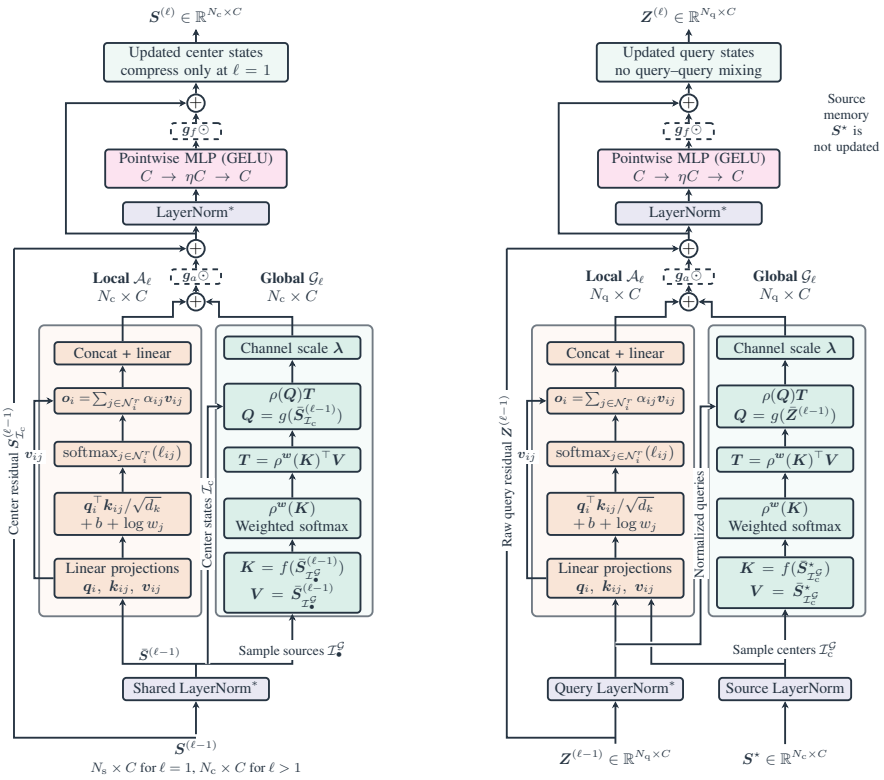
$$\ell_{ij} = \frac{\boldsymbol{q}_i^\top \boldsymbol{k}_{ij}}{\sqrt{d_k}} + b\left(\frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{r}, \frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{r}\right) + \log w_j, \tag{2}$$

$$\alpha_{ij} = \mathrm{softmax}_{j\in\mathcal{N}_i}(\ell_{ij}), \qquad \boldsymbol{o}_i = \sum_{j\in\mathcal{N}_i} \alpha_{ij} \boldsymbol{v}_{ij}, \tag{3}$$

where b is a two-layer MLP over relative displacement and distance. The log-weight term makes Equation 3 a normalized, quadrature-weighted aggregation rather than an unweighted average over the sampling distribution. It provides a discrete approximation to a normalized local integral, as detailed in Appendix B. Changing the ball radius r or cap K affects this approximation. The coordinate-based definition permits different discretizations without implying identical predictions across them.

Concatenating H heads and applying an output projection gives the local attention path's output $\mathcal{A}_\ell(\bar{\boldsymbol{S}}^{(\ell-1)}, \mathcal{N}^{(\ell)}, \boldsymbol{w}_{\mathcal{I}_\bullet}, \boldsymbol{c}) \in \mathbb{R}^{N_c \times C}$, with $\boldsymbol{w}_{\mathcal{I}_\bullet}$ the weights of source points $\boldsymbol{w}_{\mathcal{I}_s}$ in the first block, and the weights of center points $\boldsymbol{w}_{\mathcal{I}_c}$ in subsequent blocks. In the first block, the ball radius is set to $r = R_0$, and thus $\mathcal{N}^{(\ell=1)} = \{\mathcal{N}_i^{r=R_0}\}_{i=1}^{N_c}$. In subsequent blocks, further downsampling is achieved by only retaining center points, but not source points. Consequently, only centers remain available as neighbors. To maintain a similar number of neighbors within a neighborhood across blocks, the radius needs to be increased. For two-dimensional domains and surfaces, a density-based choice is $R_c = R_0\sqrt{N_s/N_c}$, which accounts approximately for the increased spacing after center selection. This is a geometric heuristic rather than a guarantee of coverage or unchanged neighborhood cardinality for which connectivity is given by $\mathcal{N}^{(\ell>1)} = \{\mathcal{N}_i^{r=R_c}\}_{i=1}^{N_c}$.

Note that center selection, radii, and neighborhoods remain fixed within each forward pass. Holding R_0 and R_c fixed across resolutions in a consistent coordinate system preserves the geometric extent of local neighborhoods at corresponding spatial locations. This makes the definition of locality resolution-independent, while the number of neighbors may vary with sampling density.



(a) Source encoder block $\mathcal{B}^\mathcal{E}_\ell$ (b) Query decoder block $\mathcal{B}^\mathcal{D}_\ell$

Figure 2: PLT blocks, read from bottom to top. The encoder retains only centers after its first block, and subsequent source blocks update these centers. Query blocks read fixed encoded states $\boldsymbol{S}^\star$. Starred normalizations and dashed gates $\boldsymbol{g}_a, \boldsymbol{g}_f$ indicate optional adaLN-Zero conditioning.

Global mixer $\mathcal{G}$. For G global tokens, MLPs $f, g\colon \mathbb{R}^C \to \mathbb{R}^G$ act pointwise, and we define

$$\boldsymbol{Q} = g\left(\bar{\boldsymbol{S}}^{(\ell-1)}_{\mathcal{I}_c}\right) \in \mathbb{R}^{N_c \times G}, \quad \boldsymbol{K} = f\left(\bar{\boldsymbol{S}}^{(\ell-1)}_{\mathcal{I}^\mathcal{G}_\bullet}\right) \in \mathbb{R}^{N^\mathcal{G}_\bullet \times G}, \quad \boldsymbol{V} = \bar{\boldsymbol{S}}^{(\ell-1)}_{\mathcal{I}^\mathcal{G}_\bullet} \in \mathbb{R}^{N^\mathcal{G}_\bullet \times C}, \tag{4}$$

where $\bar{\boldsymbol{S}}^{(\ell-1)}_{\mathcal{I}_c}$ denotes stacked source states of center points in the index set $\mathcal{I}_c$. In the first block, $\mathcal{I}^\mathcal{G}_\bullet$ and $N^\mathcal{G}_\bullet$ specify to $\mathcal{I}^\mathcal{G}_s \subseteq \mathcal{I}_s$ and $N^\mathcal{G}_s \le N_s$, respectively, and to $\mathcal{I}^\mathcal{G}_c \subseteq \mathcal{I}_c$ and $N^\mathcal{G}_c \le N_c$ in subsequent blocks. Thus, the global mixer pools information from $N^\mathcal{G}_s$ sampled source points in the first block and from $N^\mathcal{G}_c$ retained centers in subsequent blocks, forming global tokens that communicate this information to all N_c centers.

Subsequently, $\rho(\boldsymbol{Q})$ applies softmax across tokens for each center, while $\rho^{\boldsymbol{w}}(\boldsymbol{K})$ applies quadrature-weighted softmax across sampled source points for each token:

$$[\rho(\boldsymbol{Q})]_{it} = \text{softmax}_{t=1,\dots,G}(Q_{it}), \quad i \in \mathcal{I}_c \tag{5}$$

$$[\rho^{\boldsymbol{w}}(\boldsymbol{K})]_{jt} = \text{softmax}_{j \in \mathcal{I}^\mathcal{G}_\bullet}(K_{jt} + \log w_j), \quad t = 1 \dots G. \tag{6}$$

This factorization is not generally equivalent to dense dot-product softmax attention. It neither forms a dense source-to-query attention matrix nor applies self-attention between global tokens.

Following the factorized notation of Efficient Attention (Shen et al., 2021), we write the quadrature-weighted global mixing path's output as

$$\mathcal{G}_\ell(\bar{\boldsymbol{S}}^{(\ell-1)}, \boldsymbol{w}) = \rho(\boldsymbol{Q})\left[\rho^{\boldsymbol{w}}(\boldsymbol{K})^\top \boldsymbol{V}\right] \text{diag}(\boldsymbol{\lambda}) \in \mathbb{R}^{N_c \times C}, \tag{7}$$

where the bracketed term forms the token matrix $\boldsymbol{T} = \rho^{\boldsymbol{w}}(\boldsymbol{K})^\top \boldsymbol{V} \in \mathbb{R}^{G \times C}$ and the learned vector $\boldsymbol{\lambda} \in \mathbb{R}^C$ scales output channels.

Combined local attention and global mixing. Each attention block combines local attention $\mathcal{A}_\ell$, global mixing $\mathcal{G}_\ell$, and a pointwise residual MLP. The update of source states at center points is given by

$$\begin{aligned} \boldsymbol{U} &= \boldsymbol{S}_{\mathcal{I}_c}^{(\ell-1)} + \mathcal{A}_\ell(\bar{\boldsymbol{S}}^{(\ell-1)}, \mathcal{N}^{(\ell)}, \boldsymbol{w}, \boldsymbol{c}) + \mathcal{G}_\ell(\bar{\boldsymbol{S}}^{(\ell-1)}, \boldsymbol{w}) \in \mathbb{R}^{N_c \times C}, \\ \boldsymbol{S}_{\mathcal{I}_c}^{(\ell)} &= \mathcal{B}_\ell^{\mathcal{E}}(\boldsymbol{S}^{(\ell-1)}, \mathcal{N}^{(\ell)}, \boldsymbol{w}, \boldsymbol{c}) = \boldsymbol{U} + F_\ell^{\mathrm{MLP}}(\mathrm{LN}(\boldsymbol{U})) \in \mathbb{R}^{N_c \times C}, \qquad \ell = 1, \dots, L_s. \end{aligned} \tag{8}$$

The pointwise residual MLP F_ℓ^{MLP} has two affine layers with a GELU activation and expansion ratio η. The encoder returns $\boldsymbol{S}^\star = \boldsymbol{S}^{(L_s)} = \mathcal{E}(\boldsymbol{S}^{(0)}, \mathcal{N}^{(\ell=1)}, \mathcal{N}^{(\ell>1)}, \boldsymbol{w}, \boldsymbol{c})$.

2.2 Query decoder

The decoder updates independently specified queries $\mathcal{X}_q = \{\boldsymbol{x}_i^q\}_{i=1}^{N_q} \subset \Omega$ using fixed encoded centers $\boldsymbol{S}^\star$ (Figure 2). An affine lift and optional positional embedding initialize $\boldsymbol{z}_i^{(0)} = \phi_q(\boldsymbol{x}_i^q) + p(\boldsymbol{x}_i^q) \in \mathbb{R}^C$, stacked into $\boldsymbol{Z}^{(0)} \in \mathbb{R}^{N_q \times C}$ and updated through L_q blocks:

$$\boldsymbol{Z}^{(\ell)} = \mathcal{B}_\ell^{\mathcal{D}}\left(\boldsymbol{Z}^{(\ell-1)}, \boldsymbol{S}^\star, \mathcal{N}^q, \boldsymbol{w}_{\mathcal{I}_c}, \boldsymbol{c}\right) \in \mathbb{R}^{N_q \times C}, \qquad \ell = 1, \dots, L_q. \tag{9}$$

Here, the connectivity is given by $\mathcal{N}^q = \{\mathcal{N}_i\}_{i=1}^{N_q}$, with $\mathcal{N}_i$ the local neighborhood of query point $\boldsymbol{x}_i^q$ with radius R_c, capped at K neighbors. Normalized query states $\boldsymbol{Z}^{(\ell-1)}$ provide attention queries, while source states $\boldsymbol{S}^\star$ provide local keys and values and global pooling inputs. Both paths return $\mathbb{R}^{N_q \times C}$ updates without query–query attention. Pointwise layer normalization and an affine projection map $\boldsymbol{Z}^{(L_q)}$ to output-field predictions.

For fixed conditions and time, let $\mathcal{S} = (\mathcal{X}_c, \boldsymbol{S}^\star, \boldsymbol{w}_{\mathcal{I}_c})$ and partition $\mathcal{X}_q = \bigsqcup_b \mathcal{X}_{q,b}$. In evaluation mode, with neighborhoods and global samples fixed, the decoder satisfies

$$\mathcal{D}(\mathcal{S}, \mathcal{X}_q) = \mathrm{concat}_b\, \mathcal{D}(\mathcal{S}, \mathcal{X}_{q,b}), \tag{10}$$

after restoring query order. During training, chunking still retains intermediate features for gradient computation; checkpointing source blocks and decoder chunks (Chen et al., 2016) reduces storage by recomputing these features during the backward pass. These execution options preserve the learned parameters and operator up to floating-point effects.

Local attention evaluates at most $K(L_s N_c + L_q N_q)$ edges, with $\mathcal{O}(D + C)$ arithmetic per edge. Global mixing costs $\mathcal{O}((N_\bullet^{\mathcal{G}} + N_p)GC)$ for a block updating N_p centers or queries in a source or query block, respectively. Lifting, source projections, pointwise MLPs, graph preparation, and checkpoint replay add costs, so we assess memory and latency for complete model execution in Section 4. Unlike dense self-attention, whose pairwise aggregation cost grows quadratically with the number of points, bounded local attention and fixed-size global mixing scale linearly with the participating point counts for fixed K and G.

3 Benchmark results

3.1 Plasticity

The Plasticity benchmark considers the time-dependent deformation of an elasto-plastic material block compressed by a rigid die with varying shape. The learning task is to predict the evolving mesh coordinates and displacement field from the die geometry, with the governing equations and dataset construction described by Li et al. (2023c).

PLT achieves a full-trajectory relative L_2 error of 6.44×10^{-4}, the lowest among the methods compared in Appendix D. This represents a 41.46% reduction relative to the strongest reported baseline, Transolver++ (Luo et al., 2025), whose error is 1.10×10^{-3}. The final-time predictions in Figure 3 closely reproduce the deformed geometries and displacement patterns of three test samples, complementing the trajectory-level comparison and demonstrating accurate prediction of nonlinear, time-dependent material response. We additionally evaluate geometry-dependent stress prediction on the Elasticity benchmark. PLT achieves a mean relative L_2 error of 5.08×10^{-3}, the lowest among the errors reported in the literature. The Elasticity benchmark is used for the ablation study in Appendix E, where we examine the contributions of the attention components and the effects of sampling budgets.

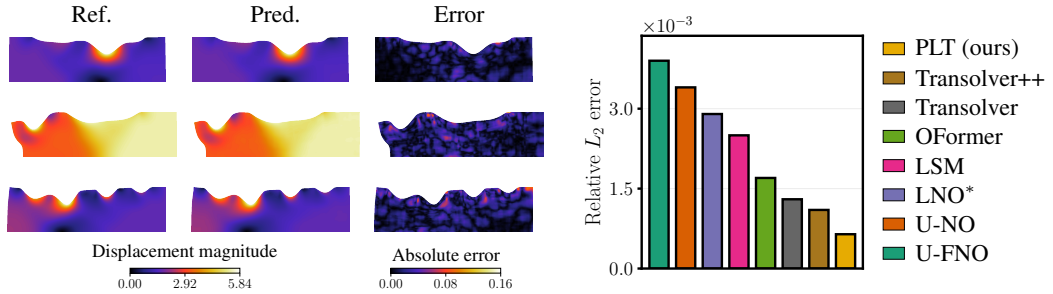


Figure 3: Plasticity predictions and benchmark comparison. Left, reference and predicted displacement magnitudes and absolute errors at the final time for three test samples. Right, full-trajectory relative L_2 errors for the eight best-performing models.

3.2 SuperWing surface-flow prediction

SuperWing (Yang et al., 2025) is a benchmark for transonic surface-flow prediction across varying wing geometries and operating conditions. Given surface coordinates, angle of attack, and Mach number, PLT predicts the pressure coefficient C_p and skin-friction components $C_{f,\tau}$ and $C_{f,z}$ at all 32,768 surface points. We compare with AeroJEPA and the FigConvUNet, GeoTransolver, and Transolver results reported by Giral et al. (2026). Figure 4 focuses on pressure prediction, while Appendix F.1 provides the comparison for all three fields and the dataset, training, and evaluation details.

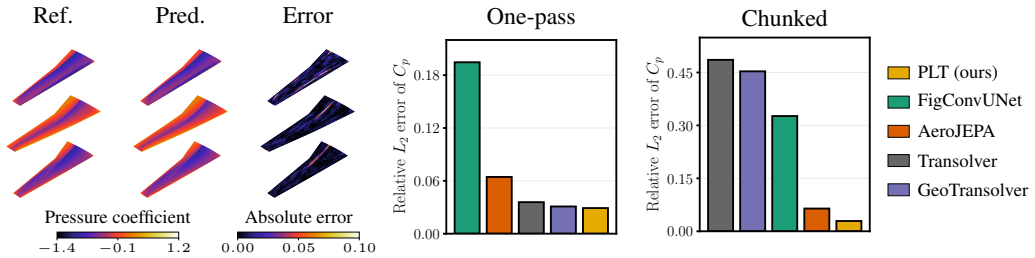


Figure 4: SuperWing pressure coefficient predictions for geometries 735, 2240, and 3053 (top to bottom), and full-field relative L_2 errors under one-pass and chunked inference.

Figure 4 shows close agreement between the predicted and reference pressure distributions on three held-out wing geometries. Across the full test set, PLT achieves a mean relative L_2 error of 2.91×10^{-2} for C_p, the lowest one-pass error among the compared models. This represents a 5.95% reduction relative to GeoTransolver, the strongest one-pass baseline, and an 18.82% reduction relative to Transolver. Under chunked inference, PLT achieves the lowest reported C_p relative L_2 error among all compared models, reducing the error by 54.87% relative to AeroJEPA, the strongest chunked baseline. Relative to FigConvUNet, GeoTransolver, and Transolver, the error reduction ranges from 91.10% to 94.02%. Both PLT and AeroJEPA preserve their one-pass accuracy, whereas the other three models exhibit substantial error increases under chunking. For PLT, partitioning the output queries leaves the encoded source field and each query's source dependencies unchanged, consistent with Equation 10. Dividing the full surface into four 8,192-query chunks reduces peak allocated GPU memory from 0.591 to 0.322 GiB, a 45.6% reduction, while predictions agree within numerical tolerance across all 2,871 test cases. This reduction comes with increased inference latency, providing a memory–latency trade-off without sacrificing accuracy.

Figure 5 examines the dependence of a selected pressure prediction on the lifted source features of each illustrated wing. Although each query directly attends to at most 48 processed centers, sensitivity extends beyond the region connected to the query through local attention alone. These examples illustrate how local attention and sampled global mixing together provide a spatially distributed receptive field without dense point-to-point attention. Definitions and diagnostic details are provided in Appendix F.2.

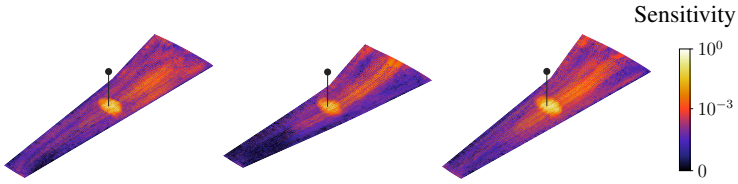


Figure 5: Source-feature sensitivity for one pressure query on each of geometries 735, 2240, and 3053, from left to right. Colors show gradient norms, normalized by a shared maximum across all three cases. The color scale is linear below 10^{-5} and logarithmic above. Thin pins indicate query locations.

3.3 HiLiftAeroML surface-pressure prediction

HiLiftAeroML (Ashton et al., 2026) provides wall-modeled large-eddy simulations of 180 high-lift aircraft geometries at ten angles of attack. We evaluate the official single-angle benchmarks at $4°$, $12°$, and $22°$, each comprising 126 training, 18 validation, and 36 test geometries. Evaluation follows the official benchmark protocols, with error metrics computed using the prescribed surface-area weighting on the native mesh. Separate PLT models predict the time-averaged surface pressure coefficient C_p from geometry alone, assessing generalization to unseen aircraft geometries at fixed operating conditions. To our knowledge, these are the first full-field surface-pressure prediction results reported for the official HiLiftAeroML single-angle benchmark protocols. Table 1 shows mean

Table 1: HiLiftAeroML full-field C_p errors and peak GPU memory. Errors are surface-area-weighted and averaged over the 36 test geometries at each angle of attack, while memory is the maximum peak allocation across these geometries. Relative errors are ratios rather than percentages. Regime labels follow Ashton et al. (2026).

Model	AoA	Regime	Surface points (millions)	Relative L_2	Relative L_1	RMSE	MAE	Peak GPU memory (GiB)
PLT (ours)	$4°$	Pre-stall	126.1–140.4	0.065	0.034	0.030	0.011	1.262
PLT (ours)	$12°$	Mid-range	137.0–150.6	0.081	0.041	0.062	0.019	1.231
PLT (ours)	$22°$	Post-stall	135.6–150.6	0.193	0.108	0.127	0.049	1.234

area-weighted relative L_2 errors below 0.09 at $4°$ and $12°$. Figure 6 shows close agreement between the reference and predicted pressure patterns for the illustrated aircraft at $12°$, together with the error distributions across all 36 test geometries at each angle. At $22°$, the benchmark probes a challenging post-stall regime in which many configurations exhibit large-scale flow separation (Ashton et al., 2026). The higher prediction errors extend across the test geometries rather than arising solely from isolated outliers, highlighting the difficulty of accurate prediction in this regime. Across all three benchmarks, chunked inference predicts C_p at every native surface point, covering 126–151 million output locations per case, with less than 1.30 GiB of peak allocated GPU memory. These output counts are distinct from the sampled geometry inputs used by the encoder. The results demonstrate full-native-surface prediction at industrial resolution with modest GPU memory requirements.

All three single-angle PLT models in Table 1 yield smaller C_p RMSE than the values converted from pressure errors reported for AeroJEPA and its baselines (Giral et al., 2026), including the challenging post-stall case at $22°$. PLT's mean area-weighted C_p RMSE values are 0.030, 0.062, and 0.127 at $4°$, $12°$, and $22°$, respectively, compared with a converted RMSE of 0.207 for AeroJEPA, the best-performing reported model. These error magnitudes are obtained under different evaluation protocols, with differences in data splits, evaluation locations, and metric weighting. Appendix G describes these differences and provides training settings, metric definitions, and evaluation details.

4 Computational scaling

We evaluate the training cost of PLT, UPT (Alkin et al., 2024), Transolver (Wu et al., 2024a), Transolver++ (Luo et al., 2025), and AeroJEPA (Giral et al., 2026), as the number of input points increases.

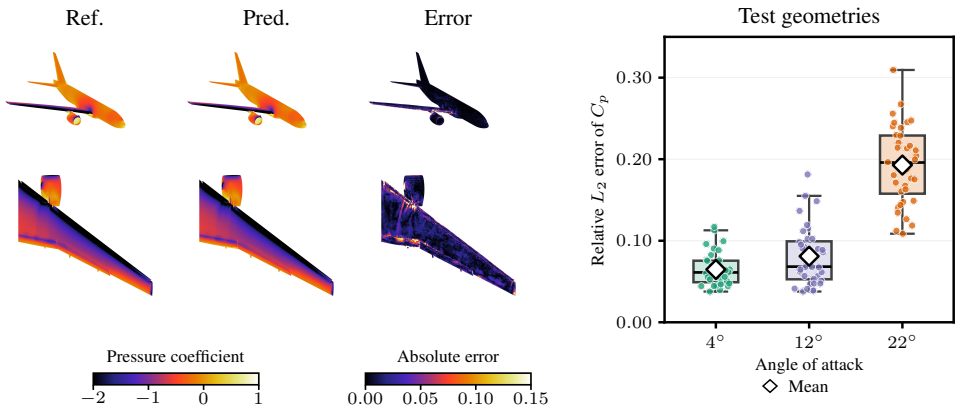


Figure 6: HiLiftAeroML surface predictions and full-field accuracy. Left: Reference, prediction, and absolute C_p error for test case LHC009 at 12°, showing the available half-aircraft and wing detail. Right: Area-weighted relative L_2 errors for all 36 test geometries at 4°, 12°, and 22°, with diamonds marking the means. PLT achieves consistently low errors across all three angles, including the challenging post-stall regime at 22°.

This experiment measures memory, arithmetic cost, and latency on synthetic data, independently of the predictive-accuracy benchmarks in Section 3.

Profiling protocol. We generate quasi-uniform irregular point clouds in the unit square with $N_s = N = 32{,}768 \cdot \xi$ points, where the scale factor $\xi \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$. All models use batch size one and mean-squared-error supervision at the same $N_q = 8{,}192$ sampled query points. AeroJEPA additionally retains its coupled latent-learning losses. Runs use a single NVIDIA A100 80-GB PCIe GPU with single-precision tensors. Each of three isolated runs performs one warm-up step followed by three timed forward–backward–optimizer steps. We report the median peak allocated GPU memory and step time across runs. Offline center sampling and neighbor search are excluded from timing, while online geometry preparation is included. We count floating-point operations (FLOPs) for forward and backward matrix arithmetic, including fused local attention and checkpoint recomputation. Appendix H provides the aggregation and counting conventions, optimizer settings, and additional model details.

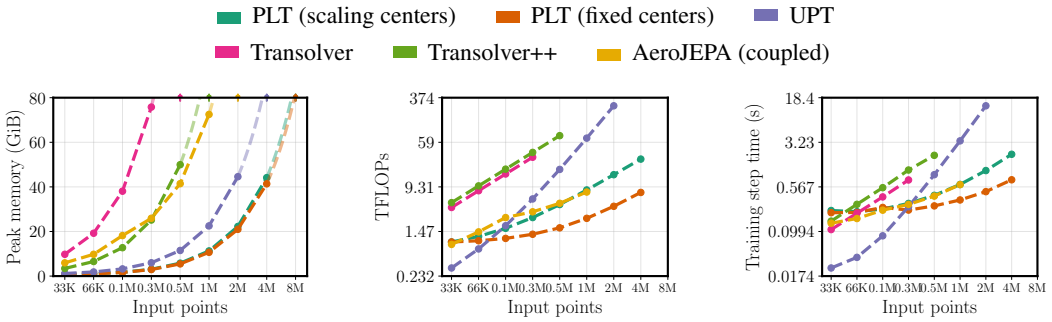


Figure 7: Training scalability on irregular point clouds with 8,192 supervised queries. From left to right, peak allocated GPU memory, counted training FLOPs, and step time for two PLT sampling policies and four baselines. Faded extensions show extrapolated memory at out-of-memory (OOM) scales, not measured allocations, with arrows marking estimates beyond the plotted range.

Models and execution policies. We compare two PLT variants with different compression policies. The scaling-center variant uses $N_c = 2{,}048 \cdot \xi$ with an input-neighborhood cap of $K = 32$, whereas the fixed-center variant uses $N_c = 2{,}048$ with $K = 32 \cdot \xi$. Both retain fixed neighborhood radii and $N_g = 2{,}048$ sampled sources per active global mixer. At the base scale $\xi = 1$, PLT, UPT, Transolver, Transolver++, and AeroJEPA have 19.6M, 20.1M, 20.0M, 18.2M, and 20.3M trainable

parameters, respectively. UPT uses an unconditioned configuration with $2{,}048 \cdot \xi$ supernodes and $512 \cdot \xi$ learned latent tokens, so its parameter count grows with ξ. AeroJEPA uses $2{,}048 \cdot \xi$ context tokens with fixed 32-point clusters, and its parameter count includes the target encoder. PLT, UPT, and AeroJEPA decode the sampled queries, whereas both Transolver variants compute all N outputs before selecting the loss points. PLT combines active-source compaction and query chunking with activation checkpointing (Chen et al., 2016). Transolver++ retains its released attention and MLP checkpointing, while UPT, Transolver, and AeroJEPA use no activation checkpointing. Detailed architectures and execution settings are provided in Appendix H.

Results. Figure 7 shows that both PLT variants use less peak GPU memory than the baselines at shared feasible input sizes, including $\xi \in \{1, 2, 4, 8\}$ where every model runs successfully. Both PLT variants support training with 4.19M input points ($\xi = 128$), twice the largest successful input size of any tested baseline, attained by UPT at $\xi = 64$. At $\xi = 128$, the scaling-center variant uses 44.1 GiB and 2.02 s per step, while the fixed-center variant uses 41.4 GiB and 0.75 s per step. The latter requires approximately one quarter of the counted training FLOPs of scaling centers and achieves a $2.71\times$ step-time speedup, although fine-input processing still increases memory use as N grows. Transolver, Transolver++, AeroJEPA, and UPT first run out of memory at $\xi = 16, 32, 64, 128$, respectively, while both PLT variants first run out of memory at $\xi = 256$. The memory advantage does not imply lower latency at small scales. At $\xi = 1$, PLT takes 207–225 ms per step compared with 24 ms for UPT, but both PLT variants become faster than UPT at $\xi = 16$. PLT demonstrates strong computational scaling, supporting high-resolution physical-field prediction in large-scale scientific and industrial applications.

5 Conclusion

Accurate prediction of high-resolution physical fields requires models that accommodate irregular geometries while controlling computational cost. PLT addresses this challenge through quadrature-weighted local attention, sampled global communication, and separate source encoding and query decoding. The architecture retains a geometrically anchored representation while making the requested output resolution independent of the number of input points. This provides a flexible basis for learning on point sets and evaluating physical fields wherever predictions are needed. The benchmark results establish the accuracy of this approach across solid mechanics and aerodynamics. PLT achieves the lowest errors among the compared methods on Plasticity and Elasticity benchmark tasks, capturing both time-dependent material deformation and geometry-dependent stress fields. The Elasticity ablations further support the complementary roles of local aggregation and global communication. On the SuperWing benchmark, PLT achieves the lowest pressure coefficient error among the compared models in both one-pass and chunked inference. Its ability to reduce inference memory while preserving predictions shows that computational constraints can be addressed without compromising the accuracy of the selected model. Scalability is demonstrated at both the input and output stages. In the controlled scaling tests, the fixed-center PLT variant supports training with 4.19 million input points using 41.4 GiB of peak allocated memory on a single GPU. PLT uses less peak memory than the evaluated baselines at shared feasible input sizes. Separately, in the HiLiftAeroML benchmark, inference evaluates 126–151 million native surface output points per aircraft using less than 1.30 GiB of peak allocated GPU memory, with sampled geometry inputs and chunked query evaluation. Separating input representation from output evaluation thus provides a practical way to control GPU memory without restricting predictions to a coarse output mesh.

Limitations. PLT's accuracy depends on the sampled input representation and the center and neighborhood budgets. Dense output evaluation alone therefore does not guarantee that fine-scale features are accurately resolved. The higher errors for post-stall HiLiftAeroML cases identify strongly separated flows as a remaining challenge. Low peak GPU memory also does not imply low end-to-end cost, as query chunking increases inference latency.

Outlook. The combination of predictive accuracy, geometric flexibility, and memory-efficient execution makes PLT a promising approach for industrial applications requiring repeated field predictions on complex geometries. Beyond the benchmarks considered here, the results motivate the use of local geometric aggregation, compact global communication, and independently queryable outputs as building blocks for large-scale scientific machine learning.

Reproducibility statement

Code and pretrained checkpoints are provided in the supplementary materials and will be made publicly available on GitHub.

AI use statement

Generative AI tools were used to assist with code development, literature organization, and language editing. All AI-assisted code, experiments, citations, and text were reviewed by the authors, who take responsibility for the final content.

References

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2402.12365.

Neil Ashton, Adam Clark, Liam Heidt, Christopher Ivey, Sanjeeb Bose, Rahul Agrawal, Konrad Goc, Rishi Ranade, Corey Adams, Peter Sharpe, Sheel Nidhan, Semit Akkurt, Daniel Leibovici, and Jean Kossaifi. HiLiftAeroML: High-fidelity computational fluid dynamics dataset for high-lift aircraft aerodynamics. *arXiv preprint arXiv:2605.19565*, 2026. URL https://arxiv.org/abs/2605.19565.

Shuhao Cao. Choose a transformer: Fourier or galerkin. In *Advances in Neural Information Processing Systems*, 2021.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. URL https://arxiv.org/abs/1604.06174.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2205.14135.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.11929.

Francisco Giral, Abhijeet Vishwasrao, Andrea Arroyo Ramo, Mahmoud Golestanian, Federica Tonti, Adrian Lozano-Durán, Steven L. Brunton, Sergio Hoyas, Hector Gomez, Soledad Le Clainche, and Ricardo Vinuesa. AeroJEPA: Learning semantic latent representations for scalable 3D aerodynamic field modeling. *arXiv preprint arXiv:2605.05586*, 2026. URL https://arxiv.org/abs/2605.05586.

Jan Hagnberger and Mathias Niepert. SMART: Scalable mesh-free aerodynamic simulations from raw geometries using a transformer-based surrogate model. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2601.18707.

Zhongkai Hao, Chengyang Ying, Zhengyi Wang, Hang Su, Yinpeng Dong, Songming Liu, Ze Cheng, Jun Zhu, and Jian Song. GNOT: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, 2023.

Benjamin Holzschuh, Georg Kohl, Florian Redinger, and Nils Thuerey. P3D: Highly scalable 3D neural surrogates for physics simulations with global context. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8UdCE5nhFl.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M.

Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=fILj7WpI-g.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023a.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for PDE surrogate modeling. In *Advances in Neural Information Processing Systems*, 2023b.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Zongyi Li, Daniel Z. Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24 (388):1–26, 2023c.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Animashree Anandkumar. Geometry-informed neural operator for large-scale 3D PDEs. In *Advances in Neural Information Processing Systems*, volume 36, 2023d. URL https://arxiv.org/abs/2309.00583.

Xinliang Liu, Bo Xu, and Lei Zhang. HT-Net: Hierarchical transformer-based operator learning model for multiscale PDEs. *arXiv preprint arXiv:2210.10890*, 2022.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. URL https://arxiv.org/abs/2103.14030.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *International Conference on Machine Learning*, 2025.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4055–4064, 2018. URL https://proceedings.mlr.press/v80/parmar18a.html.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. URL https://arxiv.org/abs/2212.09748.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017a. URL https://arxiv.org/abs/1612.00593.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, 2017b. URL https://arxiv.org/abs/1706.02413.

Md Ashiqur Rahman, Zachary E. Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2 edition, 2003. doi: 10.1137/1.9780898718003. URL https://www-users.cse.umn.edu/~saad/IterMethBook_2ndEd.pdf.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3531–3539, 2021. URL https://arxiv.org/abs/1812.01243.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2019. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *International Conference on Learning Representations*, 2023.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse PDE problems. In *Advances in Neural Information Processing Systems*, 2024.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M. Benson. U-FNO: An enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional PDEs with latent spectral models. In *International Conference on Machine Learning*, 2023.

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*, 2024a.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point Transformer V3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4840–4851, 2024b. URL https://arxiv.org/abs/2312.10035.

Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. In *International Conference on Machine Learning*, 2024.

Yunjia Yang, Weishao Tang, Mengxin Liu, Nils Thuerey, Yufei Zhang, and Haixin Chen. SuperWing: A comprehensive transonic wing dataset for data-driven aerodynamic design. *arXiv preprint arXiv:2512.14397*, 2025. URL https://arxiv.org/abs/2512.14397.

Yunjia Yang, Babak Gholami, Caglar Gurbuz, Mohammad Rashed, and Nils Thuerey. Towards a foundation-model paradigm for aerodynamic prediction in three-dimensional design. *arXiv preprint arXiv:2604.18062*, 2026. URL https://arxiv.org/abs/2604.18062.

Maksim Zhdanov, Max Welling, and Jan-Willem van de Meent. Erwin: A tree-based hierarchical transformer for large-scale physical systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78023–78040, 2025. URL https://proceedings.mlr.press/v267/zhdanov25a.html.

APPENDIX CONTENTS

A Related work **13**
B Local operator and block details **14**
C Quadrature weights **15**
D Standard benchmarks **15**
D.1 Elasticity 16
D.2 Plasticity 16
E Ablation study on Elasticity **18**
E.1 Contributions of the attention components 18
E.2 Effect of the center budget 18
E.3 Neighborhood and global-token budgets. 19
F SuperWing protocol and full-field evaluation **19**
F.1 Model architecture and evaluation protocol 19
F.2 Feature sensitivity and receptive-field tracing 22
G HiLiftAeroML protocol and full-field evaluation **22**
H Computational scaling details **24**

A RELATED WORK

Attention-based neural operators. Galerkin Transformer (Cao, 2021) and GNOT (Hao et al., 2023) develop attention mechanisms for operator learning without relying on standard dense point-to-point attention. OFormer (Li et al., 2023a) combines attention-based input encoding with cross-attention at separately specified query coordinates, accommodating irregular sampling of both inputs and outputs. Transolver (Wu et al., 2024a) learns soft assignments from points to a smaller set of physical slices. It aggregates point features into slice states, applies self-attention between those states, and distributes the updated information through the learned assignments. Transolver++ (Luo et al., 2025) addresses homogeneous slice assignments through pointwise adaptive temperatures and Gumbel-Softmax reparameterization. It also reduces redundant projections and distributes slice aggregation across devices to support larger inputs. SMART (Hagnberger & Niepert, 2026) is a mesh-free aerodynamic surrogate that uses dense cross-attention over sampled geometry points to construct compact, condition-dependent latent representations. Its decoder attends to the intermediate representation of each corresponding encoder block, allowing independent query evaluation and chunked full-field inference.

Point-set learning. PointNet (Qi et al., 2017a) operates directly on point sets using shared feature maps and symmetric pooling. PointNet++ (Qi et al., 2017b) adds hierarchical sampling and radius-based grouping, followed by local PointNet aggregation. Its center-selection and grouping operations directly motivate PLT's neighborhoods. PLT uses quadrature-weighted attention to aggregate fine-source information into selected centers in its first source block. Only the updated centers are retained for subsequent source blocks and query decoding, rather than propagating the full fine-source state through the encoder.

Local attention. Image Transformer (Parmar et al., 2018) is an early example of local attention for image generation. Vision Transformer (Dosovitskiy et al., 2021) reduces sequence length through patch tokenization, while Swin (Liu et al., 2021) combines a hierarchy of features with shifted local windows. P3D (Holzschuh et al., 2026) combines convolutional encoding, unshifted window attention, and a separate global-context model for patch-wise 3D prediction. AeroTransformer (Yang et al., 2026) applies mesh patching and shifted windows on structured wing-surface layouts. These layouts can represent varying geometries, but their patch organization differs from grouping arbitrary unordered coordinates. Point Transformer V3 (Wu et al., 2024b) organizes point clouds using space-filling-curve serialization and applies self-attention within fixed-size patches. Varying the serialization order exchanges information across patches without requiring exact nearest-neighbor search. Such methods demonstrate that local attention is not restricted to structured grids; PLT differs by defining neighborhoods through physical radii rather than contiguous groups in a serialized sequence. Erwin (Zhdanov et al., 2025) also targets prediction of physical systems on irregular point sets. It applies self-attention within fixed-cardinality ball-tree groups, combines hierarchical coarsening and refinement, and uses rotated tree partitions to communicate across groups. Its local attention cost is linear in the number of points for fixed group size. PLT instead uses potentially

overlapping radius neighborhoods and $1 \times K$ center-to-neighborhood cross-attention, rather than $K \times K$ within-group self-attention, with sampled global mixing providing long-range communication. Whereas Erwin's refinement restores features on the input tree, PLT evaluates independently specified output queries from its retained center states.

Compressed representations. GINO (Li et al., 2023d) uses local graph integral operators to map irregular geometry samples onto a regular latent grid and decode predictions at requested output locations. Fourier neural operator layers provide global processing on this grid, with signed-distance features encoding geometry. It thus combines geometry-dependent local aggregation with efficient global computation in a structured latent space. UPT (Alkin et al., 2024) aggregates radius-graph messages onto sampled physical supernodes and processes their features before Perceiver pooling (Jaegle et al., 2022) into learned latent tokens. A transformer approximator operates on the latent representation, and a Perceiver decoder evaluates requested output coordinates. Physical supernodes and learned latent queries therefore have distinct roles. AeroJEPA (Giral et al., 2026) encodes subsampled geometry and flow point clouds through local aggregation and learns a condition-dependent predictor between their embeddings. An optional implicit decoder reconstructs physical fields. This joint-embedding objective differs from direct field supervision. PLT also uses a compressed representation and shares geometric sampling and independent query evaluation with these approaches. The distinction is not compression itself, but the retained representation. PLT processes states tied to sampled physical centers without an intermediate regular grid or Perceiver pooling into learned latent queries. These center states supply both local attention and global pooling in its query decoder.

B Local operator and block details

Continuous interpretation. For one head of the local operator $\mathcal{A}_\ell$, the discrete output approximates the normalized local integral

$$\boldsymbol{o}_i \approx \frac{\displaystyle\int_{\mathcal{B}_R(\boldsymbol{x}_i)\cap\Omega} \exp(\tilde{\ell}_{ij})\boldsymbol{v}_{ij}\,\mathrm{d}\mu(\boldsymbol{x}_j)}{\displaystyle\int_{\mathcal{B}_R(\boldsymbol{x}_i)\cap\Omega} \exp(\tilde{\ell}_{ij})\,\mathrm{d}\mu(\boldsymbol{x}_j)}, \tag{11}$$

where $\mathcal{B}_R(\boldsymbol{x}_i)$ denotes the geometric ball centered at the updated point, with $R = R_0$ in the first encoder block and $R = R_c$ in subsequent encoder blocks and the decoder. Scores and values are interpreted as continuous fields. At source points $\boldsymbol{x}_j$, $\tilde{\ell}_{ij} = \ell_{ij} - \log w_j$, with ℓ_{ij} defined in Equation 2. Since $\exp(\ell_{ij}) = w_j \exp(\tilde{\ell}_{ij})$, the additive log-weight term in the discrete softmax supplies the quadrature factor for the surface or volume measure μ. Concatenating the head outputs and applying the output projection gives $\mathcal{A}_\ell$. Quadrature weighting makes aggregation reflect the physical measure represented by each point rather than sampling density alone. Different discretizations can therefore approximate the same continuous operator.

Position and conditioning. The optional fixed sinusoidal positional embedding $p(\boldsymbol{x})$ encodes normalized coordinates and is added to the source and query lifts. For conditioned problems, adaLN-Zero (Peebles & Xie, 2023) uses an embedding $\boldsymbol{e}$ of $\boldsymbol{c}$ or t, as appropriate, to modulate a hidden state $\boldsymbol{h} \in \mathbb{R}^C$ according to

$$\mathrm{AdaLN}(\boldsymbol{h}, \boldsymbol{e}) = \big(\mathbf{1} + \gamma(\boldsymbol{e})\big) \odot \mathrm{LN}(\boldsymbol{h}) + \beta(\boldsymbol{e}). \tag{12}$$

Here, LN acts across channels without learned affine parameters, $\mathbf{1}$ is the all-ones vector, and the C-dimensional scales $\gamma(\boldsymbol{e})$ and shifts $\beta(\boldsymbol{e})$ are block- and branch-specific. Encoder blocks modulate attention and MLP normalizations. Decoder blocks modulate query and MLP normalizations while applying ordinary LN to $\boldsymbol{S}^\star$. Separate condition-dependent gates scale the combined local and global update and the MLP update. The modulation output layers are zero-initialized, making both residual updates initially zero.

Fused sparse attention. We store the selected neighborhoods in compressed sparse row (CSR) format (Saad, 2003). For each center or query point and attention head, a fused Triton kernel (Tillet et al., 2019) loads only neighboring keys and values. It combines score computation, relative-position bias, log quadrature weights, softmax normalization, and value aggregation without materializing edge-wise key and value tensors in global memory. During backward evaluation, attention probabilities are recomputed from saved log normalizers and projected features rather than stored for every edge, following the memory-saving strategy of FlashAttention (Dao et al., 2022).

C QUADRATURE WEIGHTS

The quadrature weights w_j represent the volume or area associated with each source point in the domain or on its surface. For Elasticity, we estimate relative areas from the local sampling density, assigning weights proportional to the squared distance to the eighth nearest neighbor and normalizing them to sum to one. These are approximate area weights rather than exact mesh-cell areas. For Plasticity, each quadrilateral's area is distributed equally among its four vertices on the fixed reference grid, independently of the predicted deformation. For SuperWing, each point represents a surface cell whose area is computed as the sum of two triangle areas. The Plasticity and SuperWing weights are also normalized to sum to one. For HiLiftAeroML, we use the surface dual areas supplied with the dataset.

Both local attention and global pooling incorporate $\log w_j$ in their source-normalization step. A common positive rescaling of the weights cancels in the normalized aggregation. Sampled sources and retained centers inherit their weights from the original discretization.

D STANDARD BENCHMARKS

We evaluate PLT on the Elasticity and Plasticity benchmarks introduced by Li et al. (2023c), covering geometry-dependent stress prediction and time-dependent material deformation. Table 2 compares PLT with all methods in the standard-benchmark table of Luo et al. (2025), retaining only these two tasks. PLT achieves the lowest reported errors in this comparison. The main Plasticity results and visualizations are discussed in Section 3.1.

Table 2: Relative L_2 errors on Elasticity and Plasticity. Baseline values are taken from Table 5 of Luo et al. (2025). The asterisk denotes their reproduced LNO results, and a slash retains an unavailable entry in their comparison. Bold entries indicate the lowest errors.

Model	Elasticity	Plasticity
U-FNO (2022)	0.0239	0.0039
Geo-FNO (2023c)	0.0229	0.0074
U-NO (2023)	0.0258	0.0034
F-FNO (2023)	0.0263	0.0047
LSM (2023)	0.0218	0.0025
LNO* (2024)	0.0069	0.0029
Galerkin (2021)	0.0240	0.0120
HT-Net (2022)	/	0.0333
OFormer (2023a)	0.0183	0.0017
GNOT (2023)	0.0086	0.0336
FactFormer (2023b)	/	0.0312
ONO (2024)	0.0118	0.0048
Transolver (2024a)	0.0064	0.0013
Transolver++ (2025)	0.0052	0.0011
PLT (ours)	**0.00508**	**0.000644**

Evaluation and comparison. Errors are computed in physical units after reversing target normalization and are averaged equally across test cases. Elasticity uses the discrete relative L_2 error over the complete stress field. Plasticity computes one relative L_2 error jointly over all spatial points, 20 times, and four output channels for each trajectory. These metrics use equal point weights, while quadrature weights enter the attention operations as described in Appendix C. The PLT values are single-run results, not averages across training seeds. We report errors from the last-epoch checkpoints after 2,000 training epochs.

Both benchmarks retain all source points as centers and use the complete source set for global pooling. Thus, $N_c = N_s$ and these small experiments do not reduce the source point counts. Local and global paths are active in every block, with fixed sinusoidal positional embeddings.

D.1 ELASTICITY

The task maps the geometry of a hyperelastic unit cell with a varying central void to its stress field under fixed tensile loading (Li et al., 2023c). Each geometry is represented by 972 two-dimensional points, with no additional input field or operating-condition features. We use the first 1,000 samples for training and the last 200 for testing in the released arrays. The scalar stress target is standardized using training-set statistics. Training minimizes relative L_2 error on 256 uniformly sampled queries per field, while evaluation covers all 972 points. Table 3 gives the settings of the reported run. Figure 8 compares reference and predicted stress fields for six test geometries.

Table 3: PLT architecture and training settings for Elasticity. Point counts are per geometry.

Setting	Value
Architecture	
Parameters	0.70M
Encoder and decoder depth $(L_{\mathrm{s}}, L_{\mathrm{q}})$	$(4, 4)$
Channel width C	128
Attention heads H	8
Total query/key width D	64
MLP expansion	1
Relative-bias hidden width	32
Conditioning	None
Global tokens G per block	16
Sampling	
Collocation points N	972
Source points N_{s}	972
Center points N_{c}	972 (all source points)
Radius R_0 in the first block	$R_0 = 0.15$
Subsequent encoder and decoder radius R_{c}	$R_{\mathrm{c}} = 0.15$
Maximum neighbors K	48
Global sources per block	All 972 source points
Training queries	256
Evaluation queries	972 (full field)
Optimization	
Optimizer	AdamW
Training epochs	2,000
Field batch size	4
Learning rate (initial, peak, minimum)	$(10^{-4}, 10^{-3}, 10^{-6})$
Schedule	OneCycleLR, cosine, 20% increasing phase
Weight decay	10^{-5}
Evaluation interval	25 epochs
Reported checkpoint epoch	2,000 (last epoch)
Random seed	47

D.2 PLASTICITY

The input is the shape of a rigid die compressing an elasto-plastic block (Li et al., 2023c). The one-dimensional die profile is repeated along the height of a fixed 101×31 reference grid, providing a scalar source feature at each material point. PLT directly predicts two evolving mesh-coordinate channels and two displacement channels at the requested time, without autoregressive rollout. Time is mapped to $[0, 1]$ and supplied through sinusoidal features and adaLN-Zero conditioning. The first 900 trajectories are used for training and the last 80 for testing, leaving the seven intervening trajectories unused. Input and channelwise output normalization use training-set statistics. Each training batch contains four trajectories and all 20 times, with 512 independently sampled spatial queries per trajectory and time. Evaluation covers all 3,131 points at every time. Table 4 summarizes the settings.

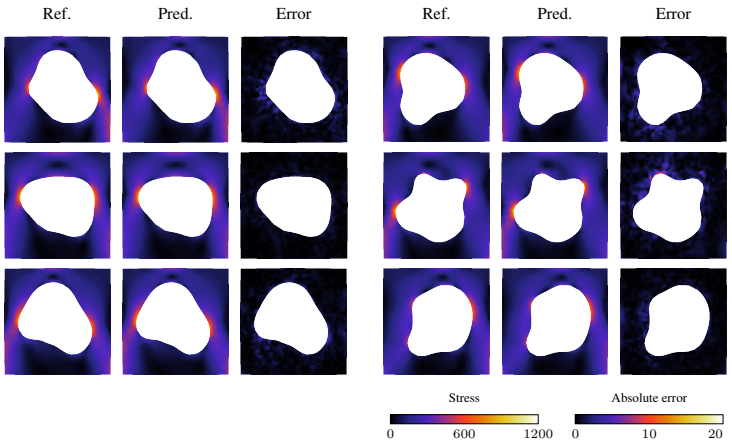


Figure 8: Elasticity stress predictions for six held-out geometries. Each block shows three samples with reference, prediction, and absolute error in columns. Filled triangular contours mask the central void and share the same stress and error scales across all samples.

Table 4: PLT architecture and training settings for Plasticity. Spatial point counts are per trajectory.

Setting	Value
Architecture	
Parameters	1.94M
Encoder and decoder depth $(L_{\mathrm{s}}, L_{\mathrm{q}})$	$(4, 4)$
Channel width C	128
Attention heads H	8
Total query/key width D	64
MLP expansion	1
Relative-bias hidden width	32
Conditioning	Time via adaLN-Zero, embedding width 192
Time embedding	32 sinusoidal frequencies, maximum 32
Global tokens G per block	16
Sampling	
Collocation points N	3,131
Source points N_{s}	3,131
Center points N_{c}	3,131 (all source points)
Radius R_0 in the first block	$R_0 = 0.075$
Subsequent encoder and decoder radius R_{c}	$R_{\mathrm{c}} = 0.075$
Maximum neighbors K	48
Global sources per block	All 3,131 source points
Training queries	512 per time
Times per trajectory per training batch	20 (all times)
Evaluation queries	$3{,}131 \times 20$ (full trajectory)
Optimization	
Optimizer	AdamW
Training epochs	2,000
Trajectory batch size	4
Learning rate (initial, peak, minimum)	$(10^{-4}, 10^{-3}, 10^{-6})$
Schedule	OneCycleLR, cosine, 20% increasing phase
Weight decay	10^{-5}
Evaluation interval	25 epochs
Selected checkpoint epoch	2,000
Random seed	0

E ABLATION STUDY ON ELASTICITY

We study the contributions of the attention components and sampling budgets using the Elasticity benchmark protocol in Appendix D.1. Unless stated otherwise, all settings follow Table 3, including $N_c = 972$, $R_0 = R_c = 0.15$, $K = 48$, and $G = 16$. We report the mean full-field relative L_2 error over 200 test geometries at the final epoch of each 2,000-epoch run. All runs use the same training seed, without averaging across independent training runs.

E.1 CONTRIBUTIONS OF THE ATTENTION COMPONENTS

Table 5 defines the four attention variants. Removing either the local attention or global mixing path applies to both the encoder and decoder. Geometric bias denotes the learned relative-position term b in Equation 2, which adjusts local attention scores using radius-normalized displacement and distance. The no-weights/bias variant jointly removes quadrature weighting from both paths and this bias from local attention, while retaining radius neighborhoods, coordinate embeddings, and learned global pooling.

Table 5: Ablation study of attention-block variants on the Elasticity benchmark. Quadrature indicates the paths using quadrature weights. Geometric bias denotes the local relative-position bias term b.

Variant	Local	Global	Quadrature	Geometric bias
PLT	Yes	Yes	Local + global	Yes
Local only	Yes	No	Local	Yes
Global only	No	Yes	Global	NA
No weights/bias	Yes	Yes	None	No

Figure 9(a) compares each variant's test error across its evaluated center budgets, using 128 centers for local-only and 972 for the other variants. The no-weights/bias variant was evaluated only at 972 centers. Full PLT achieves the lowest error. Global-only and local-only errors are approximately 1.88 and 2.76 times the full-model error, respectively. Jointly removing quadrature weighting and geometric bias increases the error by a factor of 1.63. These results support the complementary roles of local aggregation and global communication for this geometry-dependent stress-prediction task.

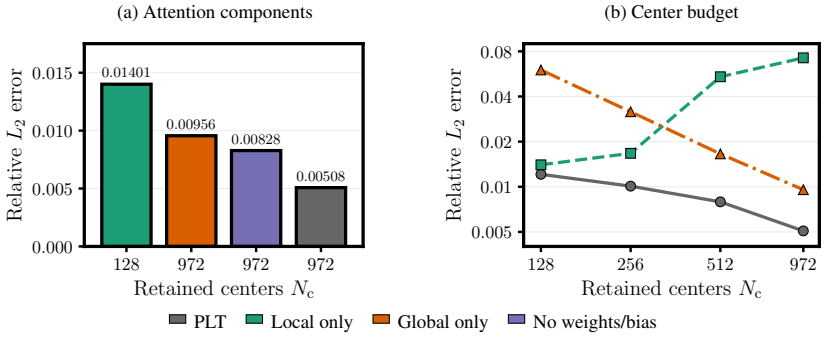


Figure 9: Elasticity benchmark ablations using full-field relative L_2 errors. (a) Lowest errors across evaluated center budgets, with the selected N_c beneath each bar. (b) Center budgets for full PLT and single-path variants. Colors identify the same variants in both panels.

E.2 EFFECT OF THE CENTER BUDGET

We vary $N_c \in \{128, 256, 512, 972\}$ while keeping the initial aggregation radius at $R_0 = 0.15$. Subsequent encoder and the decoder blocks use $R_c = R_0\sqrt{N_s/N_c}$, where N_s is equal to 972 for Elasticity. The number of global-source points equals N_c when the global path is present. Figure 9(b) shows that full PLT has the lowest error at every tested center budget and improves as more centers are retained. Local-only errors instead decrease with fewer centers, consistent with the larger physical neighborhoods improving spatial communication in the absence of a global path. Because center

count, radius, and global-source budget are coupled, this trend does not establish a benefit from compression alone.

E.3 Neighborhood and global-token budgets

Figure 10 varies the radius, neighbor cap, and number of global tokens separately, with all 972 points retained as centers. Errors decrease as each budget increases over the tested values. Eight global tokens recover most of the improvement over local-only attention, while increasing to sixteen further reduces the error.

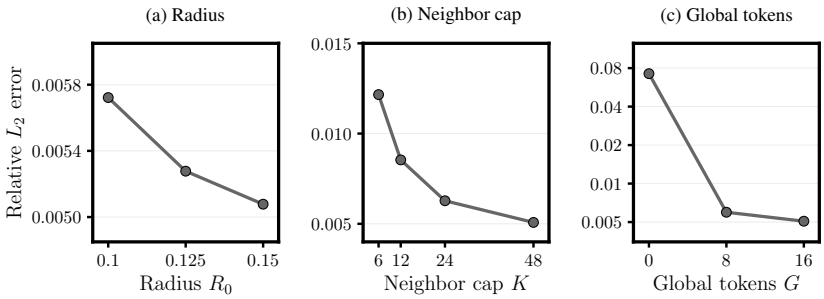


Figure 10: Relative L_2 errors for Elasticity benchmark test, with 972 centers. Each panel varies only the named setting. Vertical ranges differ, with a logarithmic error axis in (c).

F SuperWing protocol and full-field evaluation

F.1 Model architecture and evaluation protocol

Table 6: SuperWing full-field relative errors, counted forward TFLOPs per field, and peak GPU memory (GPU mem., GiB). Baseline results are from Giral et al. (2026). For chunked inference, the 32,768 queries are partitioned into four chunks of 8,192 while holding the checkpoint, conditions, centers, global samples, weights, and neighborhoods fixed.

	Relative L_2			Relative L_1				
Model	C_p	$C_{f,\tau}$	$C_{f,z}$	C_p	$C_{f,\tau}$	$C_{f,z}$	TFLOPs	GPU mem.
One-pass								
FigConvUNet	0.1946	0.1429	0.2912	0.1580	0.0863	0.2267	0.27	—
GeoTransolver	0.0309	0.0280	**0.0529**	0.0225	**0.0146**	**0.0374**	0.45	—
Transolver	0.0358	0.0324	0.0606	0.0269	0.0182	0.0459	0.35	—
AeroJEPA	0.0644	0.0548	0.1084	0.0473	0.0302	0.0768	0.32	—
PLT (ours)	**0.0291**	**0.0273**	0.0638	**0.0217**	0.0154	0.0507	**0.0345**	**0.59**
Chunked								
FigConvUNet	0.3265	0.2240	0.4909	0.2679	0.1410	0.3984	0.57	—
GeoTransolver	0.4533	0.3587	0.6990	0.3578	0.2376	0.6132	0.45	—
Transolver	0.4858	0.3858	0.8247	0.4135	0.2773	0.7352	0.35	—
AeroJEPA	0.0644	0.0548	0.1084	0.0473	0.0302	0.0768	0.32	—
PLT (ours)	**0.0291**	**0.0273**	**0.0638**	**0.0217**	**0.0154**	**0.0507**	**0.0345**	**0.32**

AeroJEPA reports a single set of results for either inference mode.

Data and representation. SuperWing (Yang et al., 2025) comprises 28,856 transonic flow solutions over 4,239 wing geometries. Given surface coordinates, angle of attack, and Mach number, PLT predicts the pressure coefficient C_p and skin-friction components $C_{f,\tau}$ and $C_{f,z}$ across the complete surface. The official shape-disjoint split assigns 25,985 flow cases to training and 2,871 to testing. All conditions of a held-out wing remain outside training. The 128-by-256 surface cell centers are

supplied as 32,768 three-dimensional coordinates, with cell-area quadrature weights derived from the supplied surface mesh. PLT consumes this grid as a coordinate point cloud rather than as image windows. Coordinates use a training-derived isotropic unit-box normalization. Angle of attack [°] and Mach number [–] are normalized to $[-1, 1]$ using the fixed bounds $[2, 12]$ and $[0.75, 0.9]$, respectively. Both conditions are embedded for adaLN-Zero conditioning in every encoder and decoder block, rather than concatenated with the source coordinates.

Model and training. Table 7 summarizes the model and training settings. The first encoder block compresses the full source field into retained centers, which provide the source states for subsequent encoder blocks and the query decoder. Training minimizes MSE on the scaled channels $(C_p, 150C_{f,\tau}, 300C_{f,z})$ at independently sampled queries. We evaluate the final epoch-200 checkpoint at all surface points, with fixed FPS centers and global samples. Training uses no dropout, activation checkpointing, or decoder chunking.

Table 7: PLT architecture and training settings for SuperWing. Point counts are per geometry.

Setting	Value
Architecture	
Parameters	1.31M
Encoder and decoder depth $(L_\mathrm{s}, L_\mathrm{q})$	$(4, 4)$
Channel width C	128
Attention heads H	8
Total query/key width D	64
MLP expansion	1
Relative-bias hidden width	32
Conditioning	adaLN-Zero, embedding width 96
Global tokens G per block	16
Sampling	
Collocation points N	32,768
Source points N_s	32,768 (full surface)
Center points N_c	8,192 selected by FPS
Radius R_0 in the first block	0.01
Subsequent encoder and decoder radius R_c	$R_\mathrm{c} = R_0\sqrt{N_\mathrm{s}/N_\mathrm{c}} = 0.02$
Maximum neighbors K	48
Global sources in the first block	8,192 sampled fine sources
Global sources in subsequent blocks	Retained centers
Training queries	2,048
Evaluation queries	32,768 (full surface)
Optimization	
Optimizer	Adam
Training epochs	200
Field batch size	4
Learning rate (initial, peak, minimum)	$(10^{-4}, 10^{-3}, 10^{-6})$
Schedule	10 warmup epochs, then cosine decay
Weight decay	0
Random seed	0

Comparison scope. The FigConvUNet, GeoTransolver, Transolver, and AeroJEPA results in Table 6 are taken from Table 2 of Giral et al. (2026). We retain their published one-pass and chunked values, repeating AeroJEPA's single reported set of errors in both groups. Dashes denote unreported GPU memory. Baseline chunking is reported as evaluated in that work and should not be interpreted as fixed-source query partitioning for every model. Training budgets and hardware are not controlled across the published and locally evaluated results.

Metrics and units. We follow the metric definitions in the released AeroJEPA evaluator[1]. Errors use equal point weights within each field and are averaged equally over the 2,871 test cases. Quadrature weights enter the model, not the reporting metrics. RMSE and MAE use raw coefficient units after

[1]NVIDIA PhysicsNeMo, AeroJEPA SuperWing metric implementation.

undoing the storage factors $(1, 150, 300)$. Literature comparisons use relative errors, which are invariant to these scaling factors.

Table 8: PLT SuperWing errors, reported as mean $\pm$ population standard deviation across 2,871 test fields, with identical displayed values for one-pass and 8,192-query chunked inference. Relative errors are ratios rather than percentages, while RMSE and MAE use raw coefficient units.

Metric	C_p	$C_{f,\tau}$	$C_{f,z}$
Relative L_2	0.0291 ± 0.0134	0.0273 ± 0.0184	0.0638 ± 0.0338
Relative L_1	0.0217 ± 0.0084	0.0154 ± 0.0079	0.0507 ± 0.0201
RMSE	0.0132 ± 0.0072	$(7.943 \pm 5.010)\ 10^{-5}$	$(3.765 \pm 2.742)\ 10^{-5}$
MAE	0.0076 ± 0.0032	$(3.867 \pm 1.689)\ 10^{-5}$	$(2.032 \pm 1.111)\ 10^{-5}$

Surface visualizations. Figure 11 shows twelve additional held-out geometries, selected uniformly without replacement with seed zero, followed by one randomly selected operating condition per geometry. Selection does not use prediction errors. Predictions use the same epoch-200 checkpoint and training normalization as the aggregate evaluation. Values are rendered on the original surface cells without smoothing, with shared pressure and error scales across all panels.

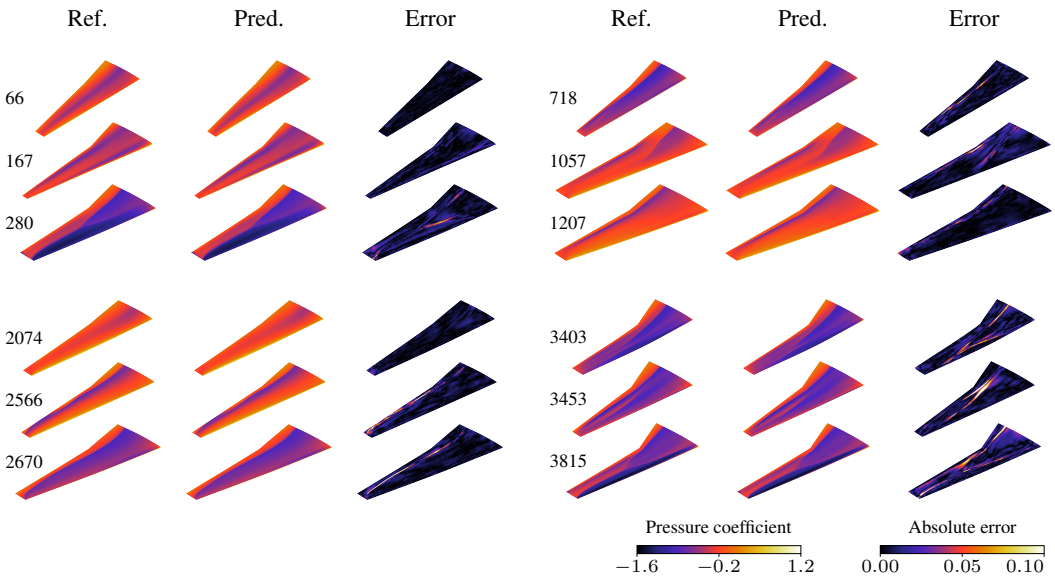


Figure 11: SuperWing surface pressure predictions for twelve held-out test geometries, identified by their geometry IDs. Each of the four panels shows three cases with reference, prediction, and absolute C_p error in columns. All panels share color scales, with error colors saturating at 0.1.

Computational cost at inference. Both PLT inference modes use an NVIDIA A100 80GB PCIe GPU, PyTorch 2.10.0 with CUDA 12.8, single-precision tensors with TF32 matrix multiplication enabled, and batch size one, without activation checkpointing. Accuracy and prediction-agreement tests cover all 2,871 cases, whereas profiling selects 16 evenly spaced cases with three warm-up and five measured forwards per case. The PLT forward includes active-source compaction, lifting, conditioning, all source and decoder blocks, and complete output assembly, with inputs and graphs already on the device. Peak memory is the maximum allocation across the profiled forwards and includes model parameters, resident inputs, outputs, temporary tensors, and PLT's graphs, but excludes CUDA driver and context memory. Table 9 summarizes these measurements.

Forward FLOPs count matrix arithmetic with two operations per multiply–add, including $2E(D + C)$ operations for each fused local-attention invocation with E neighbor interactions. Counts exclude backward computation, neighbor search, FPS, indexing, normalization, and softmax. They describe counted arithmetic rather than hardware throughput. Hardware and profiler coverage are not matched to the published baselines.

Table 9: Complete-field PLT inference on an A100 80GB GPU, including source processing once and decoding all 32,768 queries. Memory is the maximum allocated peak over 80 measured forwards, TFLOPs are means over 16 cases, and forward times are medians.

Inference	Peak (GiB)	TFLOPs	Forward (ms)
One-pass	0.591	0.03450	36.70
8,192-query chunks	0.322	0.03451	51.36

Query-partition verification and trade-off. All 2,871 fields pass an elementwise comparison with absolute tolerance 10^{-5} and relative tolerance 10^{-4}, applied to the stored output channels $(C_p, 150C_{f,\tau}, 300C_{f,z})$. The largest absolute difference between one-pass and chunked inference is $4.77 \cdot 10^{-7}$, confirming Equation 10 to numerical precision for the evaluated fields rather than bitwise identity. The resulting 45.6% memory reduction is accompanied by 39.9% higher median forward latency and approximately unchanged counted arithmetic, so chunking provides a memory–latency trade-off rather than a speedup.

F.2 Feature sensitivity and receptive-field tracing

For the same epoch-200 checkpoint and three held-out cases as Figure 4, we select one visible query near the middle of each projected wing and differentiate its predicted pressure with respect to the initial source states. For source point j, the sensitivity is

$$\left\| \nabla_{\boldsymbol{s}_j^{(0)}} \tilde{C}_p(\boldsymbol{x}_i) \right\|_2, \tag{13}$$

where $\boldsymbol{s}_j^{(0)}$ is the lifted state after positional features are added and before the first source block, as defined in Section 2. Geometry, neighborhoods, quadrature weights, centers, global samples, conditions, and model parameters remain fixed, while all global mixers remain enabled. Figure 5 divides these norms by their maximum over all source points and all three cases, rendering the original surface cells without smoothing or masking zero values. The shared color scale is linear on $[0, 10^{-5}]$ and logarithmic above this interval.

The local-only support is traced backward from the decoder's neighboring centers through the center-based encoder blocks and the initial aggregation from fine source points, retaining residual paths. Omitting global edges from this structural trace does not disable the global modules in the sensitivity calculation. Table 10 reports the resulting supports and the fraction of the summed gradient norms outside the local-only support. These fractions measure feature sensitivity, not physical attribution, explained pressure, or predictive accuracy, and the three selected queries do not establish a test-set-wide effect.

Table 10: Selected-query receptive-field diagnostics on SuperWing. Query coordinates use the training-derived isotropic unit-box normalization. "Direct" counts neighboring processed centers, while "Local-only" and "Nonzero" count initial source points in the local-only support and with a nonzero feature gradient, respectively. "Outside" is the percentage of summed gradient norms outside the local-only support, without area weighting. Each field contains 32,768 source points.

Wing	$\boldsymbol{x}_i$	Direct	Local-only	Nonzero	Outside (%)
735	(0.3165, 0.0950, 0.4629)	28	2,304	32,768	20.7
2240	(0.3348, 0.0536, 0.3481)	48	3,030	32,768	26.3
3053	(0.2977, 0.0810, 0.3870)	45	2,980	32,768	19.2

G HiLiftAeroML protocol and full-field evaluation

Data and prediction task. The official single-angle splits at $4°$, $12°$, and $22°$ share the same geometry partition of 126 training, 18 validation, and 36 test cases each (Ashton et al., 2026). Separate models predict $C_p = (p - p_{\text{ref}})/q_{\text{ref}}$ from geometry alone at each angle, using time-averaged pressure and

the dataset's reference pressure p_{ref} and reference dynamic pressure q_{ref}. We use the provided half-aircraft surface, not the volumetric flow domain. Coordinates are translated and scaled isotropically by each geometry's bounding box. Source points are sampled uniformly without replacement from positive-area native points, with quadrature weights as described in Appendix C.

Model and training. Table 11 summarizes the settings shared by all three models. Training minimizes the area-weighted relative L_2 loss on independently sampled queries. Checkpoints are selected by the mean validation relative L_2 error on fixed sampled queries, giving epochs 920, 940, and 650 for 4°, 12°, and 22°, respectively.

Table 11: PLT architecture and training settings shared by the HiLiftAeroML benchmarks at 4°, 12°, and 22°. Point counts are per geometry.

Setting	Value
Architecture	
Parameters	4.07M
Encoder and decoder depth $(L_{\mathrm{s}}, L_{\mathrm{q}})$	$(6, 6)$
Channel width C	256
Attention heads H	8
Total query/key width D	128
MLP expansion	1
Global tokens G per block	16
Sampling	
Collocation points N	126-151 million
Source points N_{s}	1,048,576
Center points N_{c}	8,192 selected by FPS
Radius R_0 in the first block	0.004
Subsequent encoder and decoder radius R_{c}	$R_{\mathrm{c}} = R_0\sqrt{N_{\mathrm{s}}/N_{\mathrm{c}}} \approx 0.0453$
Maximum neighbors K	48
Global sources in the first block	8,192 sampled fine sources
Global sources in subsequent blocks	Retained centers
Training queries	8,192
Validation queries	65,536
Optimization	
Optimizer	AdamW
Training epochs	1,000
Field batch size	1
Learning rate (initial, peak, minimum)	$(10^{-4}, 10^{-3}, 10^{-6})$
Schedule	5 warmup epochs, then cosine decay
Weight decay	10^{-5}
Validation interval	10 epochs
Random seed	0

Full-field metrics. For native surface point j, let w_j denote its dual area and $e_j = \tilde{C}_{p,j} - C_{p,j}$, where $\tilde{C}_{p,j}$ and $C_{p,j}$ are the predicted and reference pressure coefficients, respectively. The area-weighted errors for one geometry are

$$\begin{aligned} \text{Rel. } L_2 &= \left(\frac{\sum_j w_j e_j^2}{\sum_j w_j C_{p,j}^2}\right)^{1/2}, \quad \text{Rel. } L_1 = \frac{\sum_j w_j |e_j|}{\sum_j w_j |C_{p,j}|}, \\ \text{RMSE} &= \left(\frac{\sum_j w_j e_j^2}{\sum_j w_j}\right)^{1/2}, \qquad \text{MAE} = \frac{\sum_j w_j |e_j|}{\sum_j w_j}. \end{aligned} \tag{14}$$

All native points receive predictions, including zero-area points, which do not contribute to these weighted metrics. We accumulate sums in double precision and average the resulting per-geometry metrics with equal weight across the 36 test cases. Table 12 reports variation across geometries, not uncertainty across training seeds.

Table 12: HiLiftAeroML full-field C_p errors as mean $\pm$ population standard deviation across the 36 test geometries in each split.

Metric	AoA 4°	AoA 12°	AoA 22°
Relative L_2	0.065 ± 0.020	0.081 ± 0.035	0.193 ± 0.047
Relative L_1	0.034 ± 0.009	0.041 ± 0.013	0.108 ± 0.025
RMSE	0.030 ± 0.011	0.062 ± 0.027	0.127 ± 0.040
MAE	0.011 ± 0.003	0.019 ± 0.006	0.049 ± 0.012

Full-field inference. Each geometry is encoded once and decoded in 8,192-query chunks with fixed source states, streaming the native field rather than holding it on the GPU. The peak allocations in Table 1 include source encoding and query decoding.

Visualization and comparison. Figure 12 shows the selected held-out geometry LHC009 at all three angles using the validation-selected checkpoints. LHC009 provides a qualitative illustration rather than a representative mean case. Reference fields, predictions, and native absolute errors are resampled onto the supplied display mesh for visualization only. All reported metrics use the complete native surface. In Figure 6, boxes span the inter-quartile range, lines mark medians, and whiskers extend to the most extreme observations within 1.5 interquartile ranges of the box. All test cases are shown as points, and diamonds mark means.

The bars in Figure 12 compare mean C_p RMSE with values converted from the pressure errors reported for AeroJEPA and its baselines (Giral et al., 2026). The conversion is $\mathrm{RMSE}(C_p) = \mathrm{RMSE}(p)/q_{\mathrm{ref}}$, using the dataset's reference dynamic pressure $q_{\mathrm{ref}} = 4.937856$ in the same pressure units as the reported errors. PLT values are area-weighted native-surface errors averaged over 36 cases per angle. AeroJEPA reports 205 training and 50 test cases without their identifiers. The unmatched splits, evaluation locations, and metric weighting make this a contextual comparison of error magnitudes rather than a direct benchmark ranking.

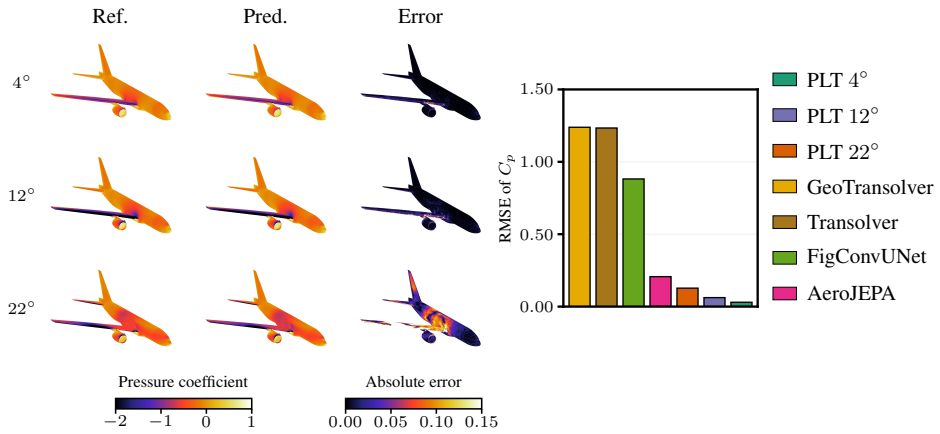


Figure 12: HiLiftAeroML test geometry LHC009 at 4°, 12°, and 22° (rows). Left, reference, prediction, and absolute C_p error (columns). Right, mean C_p RMSE for PLT and converted literature values from Giral et al. (2026), sorted in decreasing order. Protocol differences preclude a direct benchmark ranking.

H Computational Scaling Details

Synthetic inputs. The point clouds are jittered two-dimensional grids with four synthetic input channels and a scalar target. PLT uses uniform quadrature weights $w_j = 1/N$. Queries and centers are sampled uniformly, rather than using FPS. The scaling-center PLT, UPT supernodes, and AeroJEPA context tokens share center indices, while the fixed-center PLT uses a 2,048-point subset. These sampling choices are specific to the scaling study.

Model configurations. Table 13 summarizes the architectures. Both PLT variants use total query/key width $D = 192$ and MLP expansion four. The input-aggregation radius is 0.04, and subsequent encoder and decoder blocks use radius 0.16. Only the first-block neighborhood cap grows in the fixed-center variant, while subsequent blocks retain a cap of 32. Global mixing is applied by pooling 2,048 sampled sources into 16 tokens. Both Transolver variants use 32 slices and MLP expansion four. UPT's learned latent array increases its parameter count to 32.5M at $\xi = 64$. AeroJEPA includes a 4.3M-parameter target encoder, leaving 16.0M parameters on the prediction path. Its target encoder uses only the common 8,192 supervised points. The coupled latent MSE/cosine and target regularization losses are retained in addition to query MSE.

Table 13: Architectures used in the scaling study. Parameters are counted at $\xi = 1$. UPT's count grows with scale, and AeroJEPA includes its target encoder. UPT widths and heads follow encoder/latent/decoder order.

Model	Widths	Blocks by stage	Heads	Params. (M)
PLT (both variants)	384	6 encoder, 6 decoder	6	19.56
UPT	192/384/192	4 encoder, 4 latent, 4 decoder	3/6/3	20.09
Transolver	384	12 attention blocks	6	19.99
Transolver++	384	12 attention blocks	6	18.25
AeroJEPA	224	6 per encoder, 6 predictor pairs 4 decoder attention, 6 MLP layers	8	20.30

Execution and measurement. Runs use PyTorch 2.10.0 and CUDA 12.8 with single-precision tensors, TF32 matrix multiplication enabled, and no compilation. AdamW uses learning rate 10^{-4}, zero weight decay, $(\beta_1, \beta_2) = (0.9, 0.999)$, and $\epsilon = 10^{-8}$. PLT checkpoints source and query blocks and uses 1,024-query chunks. AeroJEPA also uses 1,024-query chunks, without checkpointing. Data loading and offline graph construction are excluded from timing. Online gathering, pooling, positional features, relative geometry, and CSR conversion are included. Peak allocated memory includes resident inputs and graphs, model and optimizer states, and activations retained for backward computation.

Aggregation and arithmetic counts. Step time is summarized by the median of three steps within each run, then by the median across three runs. Reported memory is the median of the three per-run peaks. FLOP counts cover forward and backward matrix arithmetic, including fused attention and checkpoint recomputation, with two operations per multiply–add. Optimizer updates, softmax, and other elementwise operations are excluded from FLOPs but included in step time. The faded OOM extensions in Figure 7 extrapolate the last two successful memory measurements linearly in input point count. They are estimates rather than measured allocations.