

PARTNO: A Neural Operator based on Physically Adaptive Region Partitioning for Solving Large-Scale Partial Differential Equations

Jiyan Qiu, Wu Yuan and Jian Zhang

Computer Network Information Center, Chinese Academy of Sciences

University of Chinese Academy of Sciences

Beijing, China

qiujiyan@cnic.cn, yuanwu@sccas.cn, zhangjian@sccas.cn

Abstract—Neural operators have shown strong potential for solving partial differential equations (PDEs). Among them, transformer-based models often achieve high accuracy, yet they face efficiency bottlenecks in large-scale settings because mesh discretization combined with global attention incurs quadratic computational cost. To address this limitation, we propose PARTNO, an efficient PDE solver based on physically adaptive region partitioning. Instead of operating on the full mesh globally, PARTNO adopts a physics-informed region decomposition strategy that partitions the computational domain into physically similar and relatively independent subregions, and models region-specific properties. Within each subregion, we introduce a dedicated Halo-Attention mechanism to jointly model intra-region physical representations and interactions with neighboring subregions, while maintaining continuity of physical quantities across region boundaries. By restricting attention computation to within subregions, PARTNO reduces the overall complexity to linear, improves parallelism, and delivers excellent scalability for large-scale inference. Extensive experiments show that PARTNO achieves state-of-the-art performance on multiple benchmarks; in particular, it obtains the best results on DrivAerStar, the largest industrial-scale automotive dataset. We further demonstrate its practicality in real-world industrial simulations, including automotive and aerospace design.

Index Terms—Neural operator, partial differential equations, Transformer, domain decomposition, attention mechanism, large-scale simulation

I. INTRODUCTION

Solving partial differential equations (PDEs) is a central problem in scientific computing. Mathematical and numerical methods are routinely used to approximate solutions over multiple variables and their partial derivatives, with applications spanning physics [1, 2], engineering [3, 4], meteorology [5, 6], and finance [7, 8]. In practical simulations, continuous fields are discretized onto meshes [9], enabling iterative solvers to operate on finite sets of nodes.

High-fidelity simulations typically require fine meshes to resolve critical structures and multiscale effects [10]. While accuracy improves with resolution, the resulting growth in degrees of freedom makes large-scale computation increasingly expensive [11]. To mitigate this cost, learning-based surrogates have been actively explored [12–14]. Among them,

transformer architectures [15] provide strong representational capacity for approximating complex solution operators.

Despite their promise, transformers on PDE meshes remain limited by two fundamental challenges. First, global self-attention scales quadratically with the number of mesh nodes, causing compute and memory requirements to become prohibitive on large meshes [16, 17]. Second, PDE solutions are heterogeneous and multiscale, and robust accuracy often benefits from physics-aware inductive biases beyond generic sequence modeling [18, 19].

In classical large-scale PDE solvers, domain decomposition is a standard strategy for improving efficiency and scalability [12]. The computational domain is partitioned into subdomains, local computation is performed within each subdomain, and coupling is enforced through boundary exchanges. This view is particularly suitable for complex geometries, where physical fields exhibit pronounced regional variation while continuity across interfaces must still be preserved. These properties suggest that a scalable neural operator should allocate computation region-wise and explicitly model cross-region interactions [20, 21].

Motivated by these observations, we propose the PARTNO, a transformer-based solver that integrates physically adaptive region partitioning with explicit interface communication. The domain is decomposed into physically similar and relatively independent subdomains to enable modular computation and scalable inference. Within each subdomain, mesh nodes are organized into inner nodes and halo nodes: inner nodes are fully contained within a subdomain, whereas halo nodes lie on subdomain interfaces and mediate information exchange. We introduce a Halo-Attention mechanism that models representations locally within each subdomain while communicating neighbor interactions through halo nodes, thereby maintaining continuity of physical quantities across subdomain boundaries. By restricting attention computation to within subdomains, the overall complexity is reduced from quadratic to linear in the number of mesh nodes, and the resulting structure naturally supports batched and distributed execution.

In summary, the main contributions of this work are as follows:

- Introduces a transformer-based PDE solving framework in which physics-informed region partitioning and interface coupling enable scalable computation on large meshes.
- Develops PARTNO with Halo-Attention to couple independent subdomain computation via halo-node message passing.
- Demonstrates strong empirical results on DrivAerStar, an industrial-scale automotive dataset, achieving state-of-the-art performance.

II. METHOD

A. Problem Settings

We consider learning a solution operator for PDEs defined on a domain Ω . In practice, Ω is discretized into N mesh nodes with geometric (or parameter) features $g \in \mathbb{R}^{N \times C_g}$. Given g and optional observed physical fields $u^{\text{in}} \in \mathbb{R}^{N \times C_u}$, the goal is to predict target fields $u \in \mathbb{R}^{N \times C_u}$ via an operator G , i.e.,

$$u = G(g, u^{\text{in}}). \quad (1)$$

To enable scalable computation on large meshes, we adopt a domain-decomposition view: Ω is partitioned into subdomains $\{\Omega_i\}$ that are coupled through interface continuity on $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$, requiring

$$u_i|_{\Gamma_{ij}} = u_j|_{\Gamma_{ij}}. \quad (2)$$

PARTNO follows this structure by computing representations locally within each subdomain and exchanging information only through boundary (halo) overlap.

B. Overview of PARTNO

Instead of modeling relationships across all grid points using global attention, which can be computationally prohibitive on large meshes, we propose PARTNO. It decomposes the discretized domain into M regions (*parts*) and performs computation primarily within each part, while coupling parts only through a lightweight interface exchange. This design mitigates the complexity challenges caused by the large number of mesh nodes, and is well suited for large-scale physical simulations and distributed execution.

PARTNO consists of two main components: Physically Aware Region Partitioning and Halo-Attention. The overall workflow is illustrated in Figure 1. Given a mesh input, an encoder first embeds the raw features into high-dimensional node representations. The physically aware partitioning then divides the mesh into M regions, each with an inner node set and a halo node set. During computation, the Inners-to-Halos step gathers halo nodes (interfaces) needed by each region from its neighbors. Halo-Attention then updates inner features by attending to the corresponding halo features, enabling explicit cross-region coupling while keeping computation region-local. After stacking multiple Inners-to-Halos and Halo-Attention layers, we merge region-wise features back to the original mesh order and apply task-specific decoders to predict physical quantities.

Formally, PARTNO can be written as $u = G_\theta(g, u^{\text{in}})$. The network consists of L stacked Halo-Attention layers. For layer $\ell \in \{1, \dots, L\}$, let $Z^\ell \in \mathbb{R}^{N \times C}$ denote the node features after the ℓ -th layer. The initial features are obtained by an MLP embedding of geometry and input observations. After the final layer, region merging restores the original node ordering, and task-specific MLP decoders map Z^L to the target physical fields u .

C. Halo-Attention

Given node features $x \in \mathbb{R}^{N \times C}$, we first partition the mesh into M regions using physically aware decomposition (Section II-D). For region i , let $\mathcal{M}_i$ denote its inner node index set and $\mathcal{E}_i$ denote its halo node index set. We gather region-local tensors by indexing:

$$\begin{aligned} \text{inner}_i &= \{z_j \mid j \in \mathcal{M}_i\} \in \mathbb{R}^{|\mathcal{M}_i| \times C}, \\ \text{halo}_i &= \{z_j \mid j \in \mathcal{E}_i\} \in \mathbb{R}^{|\mathcal{E}_i| \times C}. \end{aligned} \quad (3)$$

Here, inner nodes are fully contained within a region. Halo nodes are defined as the k -hop neighbors of the inner nodes on the mesh graph, which typically lie on or across region interfaces and provide the information needed for cross-region coupling. In Algorithm 1, this neighborhood width is controlled by the halo parameter k . In practice, $|\mathcal{E}_i|$ is typically much smaller than $|\mathcal{M}_i|$.

As shown in the right panel of Figure 1, Halo-Attention updates region- i inner features by attending to its halo features:

$$\begin{aligned} Q_i &= \text{Linear}_q(\text{inner}_i), \\ K_i, V_i &= \text{Linear}_{kv}(\text{halo}_i), \\ \tilde{Z}_i &= \text{Softmax}\left(\frac{Q_i K_i^T}{\sqrt{C}}\right) V_i. \end{aligned} \quad (4)$$

Here, $Q_i, \tilde{Z}_i \in \mathbb{R}^{|\mathcal{M}_i| \times C}$ and $K_i, V_i \in \mathbb{R}^{|\mathcal{E}_i| \times C}$. The updated inner features are written back to the corresponding indices of the global tensor, yielding $x' = \text{Halo-Attention}(x)$. Let $e = \max_{1 \leq i \leq M} |\mathcal{E}_i|$ be the maximum halo size across regions. The total compute complexity is $O(eCN)$:

$$\begin{aligned} O\left(NC + C \sum_{i=1}^M |\mathcal{M}_i| |\mathcal{E}_i|\right) &\leq O\left(NC + eC \sum_{i=1}^M |\mathcal{M}_i|\right) \\ &= O(NC + eCN) = O(eCN) \end{aligned} \quad (5)$$

Since halo sizes are much smaller than the full mesh ($e \ll N$), the overall complexity is effectively linear in N . This locality also enables efficient distributed training and inference: assigning regions to devices, communication is confined to exchanging halo features along region interfaces. We employ multi-head attention as in the Transformer [15], splitting channels into h heads of dimension C/h to improve representational capacity.

Remark 1. Halo-Attention can achieve global consistency by exchanging information only through boundary overlap. During the training process of Halo-Attention, the boundary conditions of the subdomains are dynamically updated to maintain physical consistency. This updating is guided by the

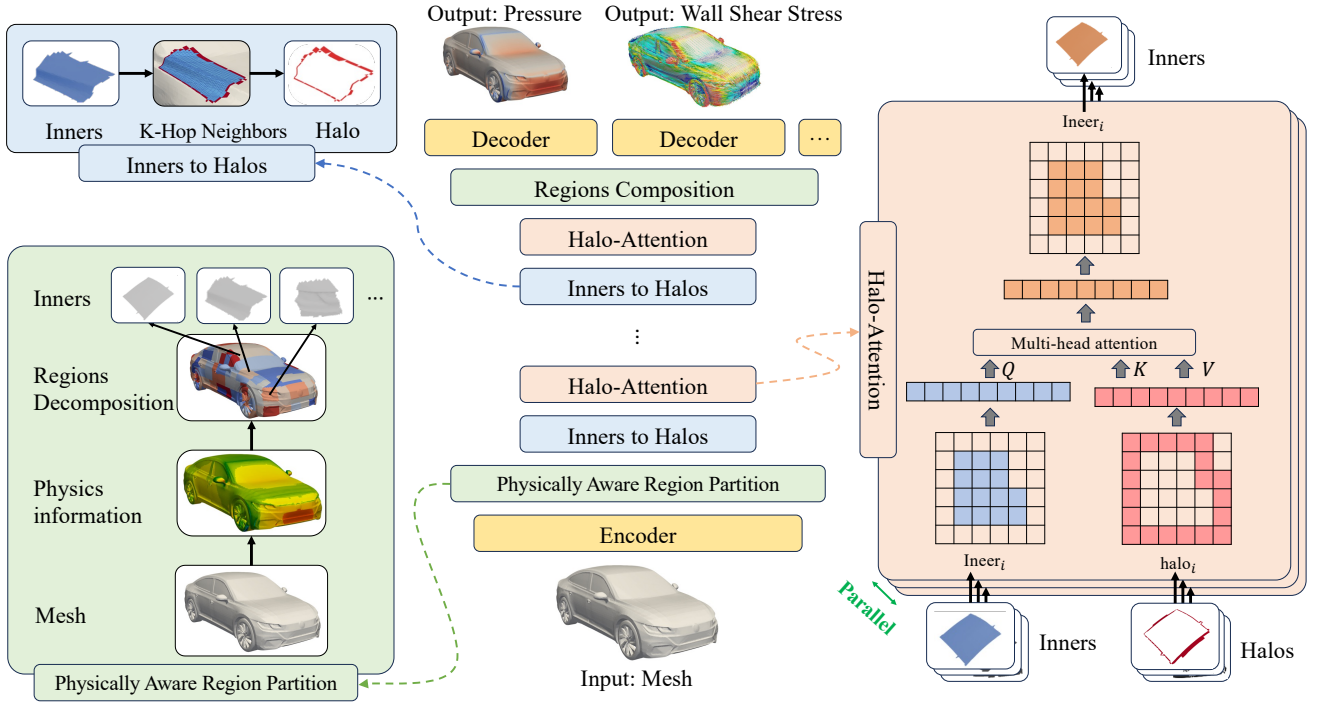


Fig. 1. Overview of the physics-aware regional Transformer architecture for fluid dynamics prediction on 3D automotive meshes. The framework first decomposes the input mesh into *inner* and *halo* node sets via physics-aware partitioning, where physical field information guides the division of sub-regions. An encoder extracts initial features from the mesh, which are then processed through stacked Halo-Attention and Inners-to-Halos mapping layers to enable cross-boundary information exchange between adjacent regions. The right panel details the Halo-Attention mechanism, which performs multi-head attention between inner and halo feature blocks to model boundary dependencies. Finally, a region composition module aggregates regional features, and task-specific decoders predict target physical quantities, including surface pressure and wall shear stress distributions.

physical information u_j of the neighboring subdomains Ω_j , ensuring continuity of the physical field across the intersection interface Γ_{ij} . This boundary adjustment strategy based on the physical information of the neighboring subdomains facilitates efficient exchange of physical information between the subdomains Ω_i and Ω_j , gradually refining the boundary conditions of each subdomain. Through iterative optimization, the deep learning operator $\mathcal{F}_i$ of each subdomain is further adjusted to approximate a solution u_i that aligns more closely with global physical laws. Ultimately, the physical information u_i of all subdomains will converge to a stable state where the physical field within each subdomain remains uniform and satisfies the global physical law $\mathcal{L}(u) = 0$ at the boundaries.

D. Physically Aware Region Partitioning

To partition the entire physical region into distinct independent regions, we introduce a sub-algorithm: Physical Perception Region Decomposition outlined in Algorithm 1. This sub-algorithm is designed to acquire physical information and segment the entire solution domain into multiple independent regions. It guarantees the uniformity of partition outcomes and ensures that the grids within each region are densely interconnected without any presence of “outliers” or “noise” points.

1) *Extracting Physical Descriptors for Partitioning:* Let g denote the set of mesh nodes (with coordinates) and u^{in}

denote optional observed physical fields. We compute a per-node physical descriptor via a shallow graph encoder:

$$\begin{aligned} X_C^0 &= \text{Linear}_v(\text{Concat}(g, u^{\text{in}})), \\ e_{i,j} &= \text{Linear}_e(g_i - g_j), \\ X_{C_i}^{n+1} &= W_1 X_{C_i}^n + W_2 \sum_{j \in \mathcal{N}(i)} e_{i,j} \odot X_{C_j}^n, \\ X_Y &= \text{Linear}_y(X_C^{L_C}). \end{aligned} \quad (6)$$

We treat mesh nodes as a graph $G_C = (g, E_C)$. For point clouds or unstructured meshes, we build edges using a radius- R neighborhood; for structured grids, we use grid connectivity. We apply a shallow graph convolution stack with depth L_C (typically $L_C \leq 3$) to obtain $X_C^{L_C}$, and project it to a scalar descriptor $X_Y \in \mathbb{R}^N$ for partitioning.

2) *Partitioning Using Physical Information:* The physical descriptor is used to allocate each node to one of the M regions, as outlined in Algorithm 1.

The algorithm first assigns nodes to regions using a coarse voxel grid, then iteratively refines region boundaries by re-assigning halo candidates to neighboring regions with more similar physical descriptors. This produces physically coherent regions while keeping halo sets compact.

Remark 2. (Allocation balancing.) The refinement step only reassigns nodes on region interfaces (halo candidates), so region sizes stay close to the initial voxel-based allocation.

Algorithm 1 Physical Perception Region Decomposition

```
1: Input:  $X_Y$ : node-wise physical descriptor,  $G_C = (g, E_C)$ : graph,  $k$ : halo width,  $V$ : voxel grid
2: Output:  $\{\mathcal{M}_i\}_{i=1}^M$ : inner node sets;  $\{\mathcal{E}_i\}_{i=1}^M$ : halo node sets
3: Initialize partitions by voxel assignment:  $\mathcal{M}_i \leftarrow \{p \in g \mid \text{coord}(p) \in V_i\}$ 
4: for iterate  $T$  times to refine boundaries do
5:   Identify halo candidates for each region:
6:    $\mathcal{E}_i \leftarrow \{p \in \mathcal{M}_i \mid \exists q \notin \mathcal{M}_i, \text{dist}(p, q) \leq k\}$ 
7:   Compute region physical centroids:  $\mu_i \leftarrow \frac{1}{|\mathcal{M}_i|} \sum_{p \in \mathcal{M}_i} X_Y(p)$ 
8:   for each region  $i$  do
9:     for each  $p \in \mathcal{E}_i$  do
10:      Let  $k^* \leftarrow \arg \min_{k \in \mathcal{N}(i)} \|X_Y(p) - \mu_k\|_2$ 
11:      if  $\|X_Y(p) - \mu_{k^*}\|_2 < \|X_Y(p) - \mu_i\|_2$  then
12:        Move  $p$  from  $\mathcal{M}_i$  to  $\mathcal{M}_{k^*}$ 
13:      end if
14:    end for
15:  end for
16: end for
17: Return  $\{\mathcal{M}_i\}$  and  $\{\mathcal{E}_i\}$ 
```

As a simple sanity check, consider an idealized random-redistribution model with M regions and N nodes (initially $P = N/M$ per region). After k random transfers, for any fixed region i the node count $X_i^{(k)}$ satisfies $\mathbb{E}[X_i^{(k)}] = N/M$ (total mass is conserved and the process is symmetric across regions). Moreover, each transfer can change X_i by at most ± 1 , so $X_i^{(k)}$ behaves like a bounded-step random walk; around the balanced state one obtains the rough scaling $\text{Var}(X_i^{(k)}) \propto k$ (more precisely $\text{Var}(X_i^{(k)}) \approx 2k/M$), hence $X_i^{(k)}$ is often well-approximated by $\mathcal{N}(N/M, 2k/M)$. In our refinement, the effective k is small because only interface nodes are eligible to move; additionally, we cap the number of moved nodes per iteration to prevent degenerate partitions.

III. RELATED WORK

Transformer-based neural operators have emerged as a powerful family of models for learning PDE solution operators on complex geometries. Our work is most closely related to efficient attention mechanisms for large discretizations and operator learning on irregular meshes.

A. Efficient attention for PDE solvers

The quadratic cost of dot-product attention becomes a major bottleneck when the number of mesh nodes is large. A broad line of work proposes approximate or structured attention to reduce complexity, including linear-time variants and locality-structured designs [22, 23]. In the PDE context, Transolver [18] reduces attention computation by grouping mesh nodes into slices and performing attention at the slice level. Other approaches factorize attention for multidimensional physical systems or time-dependent settings, such as FactFormer [24] and IFactFormer [25]. IPOT [26] performs

attention on a set of inducing points in latent space to avoid operating directly on large input discretizations.

B. Operator learning on irregular meshes

Several transformer-based neural operator frameworks are designed to handle irregular domains and arbitrary sampling. OFormer [27] separates the representation of input functions from query points, and GNOT [28] extends this idea to multiple input functions via conditional embeddings and cross-attention. CORAL [29] improves flexibility for varying geometry sampling.

IV. EXPERIMENTS

A. Experimental Setup and Evaluation Protocol

1) *Dataset and baselines:* We use the DrivAerStar dataset [30], an industrial-grade CFD benchmark for vehicle aerodynamic optimization generated with Star-CCM+ on high-quality hexahedral-dominant meshes. It contains 12,000 simulations across three canonical DrivAer vehicles, with about 2×10^5 surface points per case.

We compare our method with three representative baselines: GNOT [28], Transolver [18], and PointNet [31]. These baselines include strong Transformer-based PDE solvers and a widely used point-cloud backbone.

We follow the official DrivAerStar benchmark and evaluate all methods under two settings: *single-vehicle specialization* (train/test on one configuration) and *multi-vehicle generalization* (train jointly on multiple configurations and test across them). For each case, the model predicts surface pressure P , wall shear stress (WSS), and the drag coefficient C_D .

2) *Evaluation Protocol and Hyperparameters:* We evaluate all methods using the mean relative ℓ_2 error. Let u_i and $u'_i \in \mathbb{R}^n$ denote the ground-truth and predicted solutions for the i -th sample, and let D be the dataset size. The mean relative ℓ_2 error is defined as

$$\varepsilon = \frac{1}{D} \sum_{i=1}^D \frac{\|u'_i - u_i\|_2}{\|u_i\|_2}. \quad (7)$$

We keep training budgets and model scales comparable. All models are trained for 500 epochs using AdamW [32] with either a cyclic learning-rate schedule [33] or exponential decay, and repeated three times on NVIDIA A100 GPUs (80GB). We use the following default configuration unless otherwise stated: $L=4$, hidden width $C=64$, and $h=4$ attention heads, with MLP ratio 1, dropout 0.0, gradient clipping threshold 0.1, downsampling factor 5, and weight decay 10^{-4} . The number of partitions $M \in \{32, 64, 128\}$ is selected to balance accuracy and efficiency.

B. DrivAerStar benchmark results

We report validation errors on DrivAerStar under the two official settings described above (single-vehicle specialization and multi-vehicle generalization). We evaluate Transolver, GNOT, and PointNet following the benchmark protocol and report PARTNO under the same configurations; results are summarized in Table I.

TABLE I
DRIVAErSTAR BENCHMARK: VALIDATION ERROR. MEAN RELATIVE ℓ_2
 ERRORS FOR P AND WSS; C_D REPORTS RELATIVE ERROR.
 $S \in \{400, 800, 1200\}$ IS THE TOTAL NUMBER OF TRAINING SAMPLES.

Model	S	Single-vehicle			Multi-vehicle		
		P	WSS	C_D	P	WSS	C_D
Transolver	400	0.2173	0.7889	0.0314	0.2194	0.8049	0.0375
	800	0.2086	0.7837	0.0283	0.2075	0.8170	0.0335
	1200	0.1996	0.7953	0.0266	0.2074	0.7871	0.0286
GNOT	400	0.3250	0.9619	0.1144	0.2122	0.7795	0.0344
	800	0.2254	0.7986	0.0516	0.2002	0.7920	0.0309
	1200	0.2087	0.8217	0.0357	0.1964	0.8124	0.0315
PointNet	400	0.2991	0.6547	0.0502	0.3201	0.6767	0.0536
	800	0.2787	0.6769	0.0499	0.3007	0.7041	0.0653
	1200	0.2984	0.6273	0.0436	0.3030	0.6560	0.0757
Ours	400	0.2542	0.7823	0.0385	0.2695	0.7887	0.0292
	800	0.1920	0.7715	0.0308	0.2243	0.6792	0.0275
	1200	0.1787	0.6621	0.0252	0.2001	0.6384	0.0261

TABLE II
ABLATION ON PARTITIONING AND HALO EXCHANGE (VALIDATION).
 ALL ENTRIES ARE MEAN RELATIVE ℓ_2 ERRORS; C_D USES RELATIVE
 ERROR.

Variant	P	WSS	C_D	Notes
Full model	0.179	0.662	0.025	phys-aware + halo
w/o physics-aware partition	0.413	0.842	0.320	plane-cut partition
w/o halo exchange	0.508	0.915	0.529	no halo exchange
Small $M = 32$ (coarse partition)	0.216	0.701	0.031	fewer / larger regions
Large $M = 128$ (fine partition)	0.221	0.786	0.037	more / smaller regions

Overall, increasing the training budget S generally improves accuracy, suggesting that the benchmark is primarily data-limited in the low- S regime. Across both settings, Transformer-based solvers (Transolver and GNOT) outperform PointNet on (P, WSS, C_D) , which is expected since the task requires modeling long-range interactions and coupled surface quantities. The multi-vehicle setting is more challenging due to cross-configuration distribution shift; this gap is most visible on C_D , where small local errors can accumulate into larger coefficient deviations.

C. Ablation study

We ablate key components that enable accurate and scalable inference, including physics-aware partitioning driven by the physical descriptor, the number of partitions M (controlling the boundary-to-interior ratio), and halo exchange (enabling cross-region coupling). Unless otherwise specified, ablations are conducted on the single-vehicle specialization setting (Estimateback) with $S=1200$, and we keep backbone depth/width and the training protocol fixed. We report the same validation metrics as in the benchmark. Ablation results are reported in Table II.

Table II highlights that the two core components: physics-aware partitioning and halo exchange. Replacing the physically adaptive partitioning with a geometry-only plane cut causes a pronounced degradation across all targets, indicating that respecting physical similarity within each region is critical for learning locally coherent dynamics. Removing halo exchange

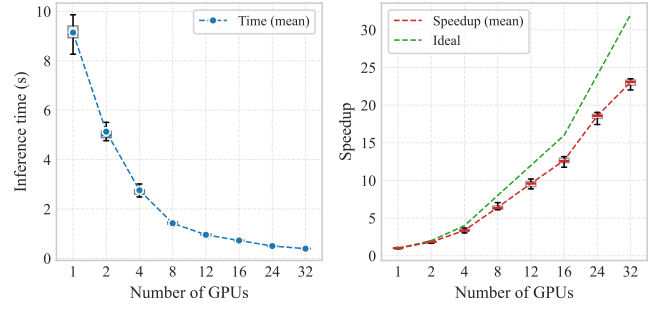


Fig. 2. **Distributed inference scaling.** Inference time and speedup versus the number of GPUs on large DrivAerStar cases. The measured speedup stays close to the ideal linear scaling.

leads to an even larger drop, which is consistent with the role of halo nodes in enforcing cross-region coupling and continuity; without explicit interface communication, errors accumulate at region boundaries and propagate to global quantities such as drag. In contrast, varying the number of partitions M mainly trades off region size and boundary-to-interior ratio: overly coarse or overly fine partitions remain workable but are consistently worse than the full model, supporting the choice of a moderate M that balances locality with sufficient context.

D. Distributed parallel inference performance

We exploit the region structure to parallelize inference with bounded communication. Given a partition into M regions, we assign regions to GPUs by balancing point counts and boundary sizes, and execute region-level operators in parallel. Cross-region coupling is implemented as halo exchange on boundary points; communication is sparse and localized to adjacent regions, so the total transferred volume scales with boundary size rather than the full mesh size. In practice, we overlap communication with computation by launching halo packing/unpacking on the GPU, using NCCL collectives or point-to-point exchanges depending on the adjacency pattern, and pipelining region batches so that earlier layers can start on one region while later layers communicate halos from another. Mixed precision and activation checkpointing are used to reduce memory pressure for large surfaces.

Figure 2 reports inference time and speedup versus the number of GPUs on large DrivAerStar cases. Note that training follows the 4-GPU setup described above; the scaling study varies the GPU count only for distributed inference. We observe strong scaling behavior with speedup close to the ideal linear trend, indicating that the region-wise computation dominates and the halo exchange overhead remains modest for these large cases.

E. Visualization

We visualize surface pressure and wall shear stress predictions on DrivAerStar in Fig. 3. Qualitatively, PARTNO recovers smooth fields with no obvious partition seams. This suggests that the halo exchange effectively propagates cross-

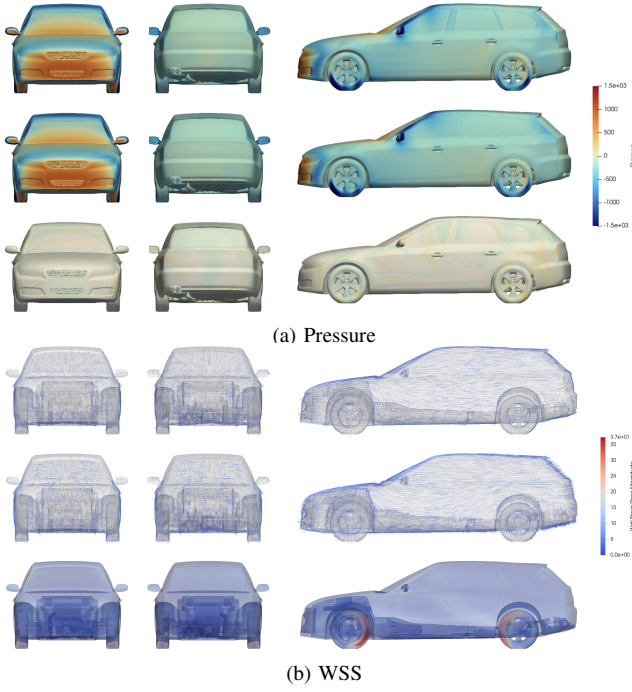


Fig. 3. **Quantitative comparison of aerodynamic pressure and WSS on vehicle geometry.** (a) Pressure field; (b) WSS field. Each subfigure shows ground truth, prediction, and absolute error (rows).

region information and preserves global consistency, while still enabling region-wise parallelism.

Figure 3 provides a detailed comparison of aerodynamic pressure and WSS. For pressure (top rows, units: Pa), the ground truth shows positive pressure concentrated on the front bumper and hood and negative pressure on the rear and underbody, consistent with aerodynamic theory. The prediction matches the global pressure trend but slightly underestimates peak positive pressure at the leading edge, while the error map highlights larger discrepancies at the front corners and rear spoiler where separation and vortex dynamics are complex. For WSS (bottom rows, units: Pa), the ground truth displays strongest shear along the wheel wells and weaker shear on the roofline; the prediction captures the main direction and magnitude trends but loses detail in high-curvature regions (e.g., wheel arches). The WSS error is largest in the wheel-well and rear-wake regions, reflecting the difficulty of modeling unsteady flow features.

V. CONCLUSIONS AND FUTURE WORK

We presented PARTNO, a transformer-based neural operator for large-scale PDE simulation on complex meshes. PARTNO integrates physically adaptive region partitioning with Halo-Attention to perform computation locally within subdomains while preserving cross-subdomain continuity through halo-node communication. This design reduces the cost of attention from global quadratic scaling to region-local linear scaling in the mesh size, and naturally enables batched and distributed inference. Extensive experiments demonstrate strong accuracy across benchmarks, state-of-the-art performance on

the industrial-scale DrivAerStar dataset, and practical strong-scaling behavior with speedup close to the ideal trend.

Several directions remain for future work. First, more principled partitioning objectives that jointly optimize accuracy, communication volume, and load balance could further improve scalability. Second, extending PARTNO to broader PDE families, higher-order boundary conditions, and coupled multi-physics settings would strengthen generality. Third, incorporating uncertainty quantification and data-driven error indicators may improve robustness for out-of-distribution geometries and operating conditions.

ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Program of China (No. 2025YFB3003504).

REFERENCES

- [1] A. H. Ganie, L. H. Sadek, M. M. Tharwat, M. A. Iqbal, M. M. Miah, M. M. Rasid, N. S. Elazab, and M. S. Osman, “New investigation of the analytical behaviors for some nonlinear PDEs in mathematical physics and modern engineering,” *Partial Differential Equations in Applied Mathematics*, vol. 9, p. 100608, Mar. 2024.
- [2] A. Ivagnes, M. Khamlich, P. Siena, and G. Rozza, “in computational fluid dynamics: An overview of methods,” *Emerging Technologies in Computational Sciences for Industry, Sustainability and Innovation: Math to Product*, p. 1, 2025.
- [3] H. Sharma, M. Patil, and C. Woolsey, “A review of structure-preserving numerical methods for engineering applications,” *Computer Methods in Applied Mechanics and Engineering*, vol. 366, p. 113067, 2020.
- [4] A. Habib, A. A. Hourri, M. T. Junaid, and S. Barakat, “A systematic and bibliometric review on physics-based neural networks applications as a solution for structural engineering partial differential equations,” in *Structures*, vol. 69. Elsevier, 2024, p. 107361.
- [5] K. Döös, P. Lundberg, and A. A. Campino, *Basic Numerical Methods in Meteorology and Oceanography*. Stockholm University Press, Dec. 2022.
- [6] E. H. Müller and R. Scheichl, “Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction,” *Quarterly Journal of the Royal Meteorological Society*, vol. 140, no. 685, pp. 2608–2624, 2014.
- [7] J. Cruz, M. Grossinho, D. Sevcovic, and C. I. Udeani, “Linear and nonlinear partial integro-differential equations arising from finance,” *arXiv preprint arXiv:2207.11568*, 2022.
- [8] T. Liu, M. Z. Ullah, S. Shateyi, C. Liu, and Y. Yang, “An efficient localized rbf-fd method to simulate the heston–hull–white pde in finance,” *Mathematics*, vol. 11, no. 4, p. 833, 2023.
- [9] A. Loseille and F. Alauzet, “Continuous mesh framework part ii: validations and applications,” *SIAM Journal on Numerical Analysis*, vol. 49, no. 1, pp. 61–86, 2011.

- [10] C. Burstedde, O. Ghattas, G. Stadler, T. Tu, and L. C. Wilcox, "Towards adaptive mesh pde simulations on petascale computers," *Proceedings of Teragrid*, vol. 8, 2008.
- [11] T. Schneider, Y. Hu, X. Gao, J. Dumas, D. Zorin, and D. Panozzo, "A large-scale comparison of tetrahedral and hexahedral elements for solving elliptic pdes with the finite element method," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 3, pp. 1–14, 2022.
- [12] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, "Combining machine learning and domain decomposition methods for the solution of partial differential equations—a review," *GAMM-Mitteilungen*, vol. 44, no. 1, p. e202100001, 2021.
- [13] J. Blechschmidt and O. G. Ernst, "Three ways to solve partial differential equations with neural networks—a review," *GAMM-Mitteilungen*, vol. 44, no. 2, p. e202100006, 2021.
- [14] C. Beck, M. Hutzenthaler, A. Jentzen, and B. Kuckuck, "An overview on deep learning-based approximation methods for partial differential equations," *arXiv preprint arXiv:2012.12348*, 2020.
- [15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.
- [16] X. Ma, X. Kong, S. Wang, C. Zhou, J. May, H. Ma, and L. Zettlemoyer, "Luna: Linear unified nested attention," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2441–2453, 2021.
- [17] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.
- [18] H. Wu, H. Luo, H. Wang, J. Wang, and M. Long, "Transolver: A fast transformer solver for pdes on general geometries," *arXiv preprint arXiv:2402.02366*, 2024.
- [19] S. L. Brunton and J. N. Kutz, "Promising directions of machine learning for partial differential equations," *Nature Computational Science*, pp. 1–12, 2024.
- [20] P. Sousa, C. V. Rodrigues, and A. Afonso, "Enhancing cfd solver with machine learning techniques," *Computer Methods in Applied Mechanics and Engineering*, vol. 429, p. 117133, 2024.
- [21] B. Haddadi, C. Jordan, and M. Harasek, "Cost efficient cfd simulations: Proper selection of domain partitioning strategies," *Computer Physics Communications*, vol. 219, pp. 121–134, 2017.
- [22] Z. Shen, M. Zhang, H. Zhao, S. Yi, and H. Li, "Efficient attention: Attention with linear complexities," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3531–3539.
- [23] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *ICLR*, 2020.
- [24] Z. Li, D. Shu, and A. Barati Farimani, "Scalable transformer for pde surrogate modeling," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [25] H. Yang, Z. Li, X. Wang, and J. Wang, "An implicit fac-torized transformer with applications to fast prediction of three-dimensional turbulence," *Theoretical and Applied Mechanics Letters*, p. 100527, 2024.
- [26] S. Lee and T. Oh, "Inducing point operator transformer: A flexible and scalable architecture for solving pdes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 153–161.
- [27] Z. Li, K. Meidani, and A. B. Farimani, "Transformer for partial differential equations' operator learning," *arXiv preprint arXiv:2205.13671*, 2022.
- [28] Z. Hao, C. Ying, Z. Wang, H. Su, Y. Dong, S. Liu, Z. Cheng, J. Zhu, and J. Song, "Gnot: A general neural operator transformer for operator learning," *ICML*, 2023.
- [29] L. Serrano, L. Le Boudec, A. Kassai Koupai, T. X. Wang, Y. Yin, J.-N. Vittaut, and P. Gallinari, "Operator learning with neural fields: Tackling pdes on general geometries," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [30] J. Qiu, L. Kuang, G. Wang, Y. Xu, L. Cui, S. Fu, Y. Zhu, and R. Zhang, "DrivAerStar: An Industrial-Grade CFD Dataset for Vehicle Aerodynamic Optimization," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, Oct. 2025.
- [31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.
- [32] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.
- [33] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006. SPIE, 2019, pp. 369–386.