FG-ConvNO: A Geometry-Aware Neural Operator for Propeller CFD Prediction

Yichen Di[1] Valentin Duruisseaux[2] Di Zhou[3] Xinyi Li[2] Daniel Leibovici[4]
Jean Kossaifi[4] Anima Anandkumar[2]

[1]Tsinghua University [2]California Institute of Technology [3]University of Tennessee [4]NVIDIA
{diyc041018@gmail.com, vduruiss@caltech.edu, xinyili@caltech.edu }

Abstract

Computational Fluid Dynamics (CFD) is essential in contemporary engineering design, yet it remains expensive to compute and can take thousands of CPU/GPU hours or more for a single high-fidelity simulation. Recent developments in machine learning for scientific modeling have led to powerful surrogate frameworks that can approximate flow fields and physical quantities by using a fraction of the computational cost. However, most of these methods require large-scale homogeneous datasets or simple geometries, exhibiting severe generalization gaps when applied to complex and data-scarce configurations. To address this, we propose the Factorized Geometry Convolutional Neural Operator (FG-ConvNO) for robust geometry-aware propeller pressure prediction under limited training data. We introduce geometry-aware encoding, efficient interpolation-based decoding, and auxiliary global quantity supervision. In addition, we incorporate DISCO blocks that use discrete continuous convolutions to make the model discretization agnostic. In our experiments, FG-ConvNO achieves the highest accuracy for both surface pressure and global thrust prediction on a procedurally generated propeller CFD dataset. Geometry-aware encoding and architectural simplification improve stability, while random vertex subsampling enhances robustness. Overall, adding geometric priors to grid-based convolutional operators reduces overfitting on complex propeller geometries and improves generalization in low-data regimes.

1 Introduction

Computational Fluid Dynamics (CFD) is the cornerstone of modern aerospace and engineering, enabling accurate prediction of aerodynamic and hydrodynamic behavior. However, traditional CFD solvers are computationally prohibitive: resolving turbulent 3D flows often requires thousands of GPU hours or more for a single geometry (Umetani and Bickel 2018). This cost severely limits rapid design iterations, parameter study, and uncertainty estimation for industrial applications. Consequently, there has been an increasing interest in machine-learning-based surrogates that efficiently emulate CFD solvers while preserving physical fidelity (Wang et al. 2024), using architectures ranging from convolutional and graph neural networks to neural operators.

Among them, neural operators are a class of methods that can learn mappings from boundary conditions and geometric representations to continuous solution fields (Kovachki et al. 2023; Azizzadenesheli et al. 2024). Architectures such as the Fourier Neural Operator (FNO) (Li et al. 2021), the Graph Neural Operator (GNO) (Li et al. 2020), and the Geometry-Informed Neural Operator (GINO) (Li et al. 2020) enable resolution-independent inference, learning PDE solution operators rather than fixed discretized mappings (Berner et al. 2025). Recent advances also include DISCO (Liu-Schiaffini et al. 2024), which replaces traditional convolution kernels with continuous kernels to help stabilize operator behavior under resolution changes. These models demonstrate exceptional performance on various benchmarks, including physical systems involving elasticity, plasticity, and turbulent flow systems such as airfoils and pipes. They also excel on aerodynamic datasets such as ShapeNet-Car (Chang et al. 2015), which feature smooth, structured geometries.

Even though these methods can deal with unstructured point clouds, they typically require access to extensive datasets to learn from. Their generalization capabilities to unseen shapes and geometries can deteriorate significantly when training data is scarce, especially for intricate designs like propellers. There, the base geometries are highly nonlinear with complex curvature, skew and pitch changes leading to very nonuniform pressure distributions. In these contexts, models trained on a few hundred objects often suffer from two competing failure modes: underfitting and overfitting.

Underfitting. In the limited data setting, diverse geometries often cause data-driven models to collapse toward nearly uniform surface predictions. Sharp variations at tips or hub regions become overly smoothed, indicating that the model fails to learn the nonlinear geometry–flow relationship and instead defaults to global averages.

Overfitting. With limited data, high-capacity models fit the training set quickly but fail to generalize. The test loss plateaus early, showing that the network memorizes sample-specific noise instead of learning stable flow patterns. This appears as noisy surface predictions where local flow features are memorized rather than generalized.

Proposed Approach. We propose a new architecture, the Factorized Geometry Convolutional Operator (FG-ConvNO), based on the Factorized Implicit Global Convolution (FIGConv) but reformulated for robustness and geometry-awareness under limited CFD data.

- **Simplified architecture:** FG-ConvNO retains the hierarchical design from FIGConv but reduces its depth and increases the number of blocks per level, avoiding overfitting while maintaining a sufficiently large receptive field. In addition, the grid-to-vertex decoding is reformulated via trilinear interpolation, yielding smoother feature reconstruction and lower memory consumption.
- **Geometry-aware encoding:** We combine signed distance functions (SDFs), surface normals, and boundary condition embeddings in the vertex–grid exchange layers. This coupling enables FG-ConvNO to utilize geometric context and flow boundary information more .
- **Integration of DISCO blocks:** In FG-ConvNO, we replace standard CNN blocks with DISCO blocks that perform discrete continuous convolution and maintain a consistent receptive field across different grid resolutions, enabling discretization invariance.
- **Global prediction objective:** The coarsest latent grid is pooled to regress global aerodynamic or hydrodynamic quantities, e.g., thrust that brings integral physical supervision from a local pressure fields to global performance metrics.

We also introduce a new (small) dataset of RANS simulations for diverse propeller geometries.

Summary of Results. Our numerical experiments on the newly introduced propeller CFD dataset demonstrates the effectiveness of the proposed modifications:

- **Superior surface accuracy and robustness:** FG-ConvNO achieves the lowest surface pressure relative error (24.98%), which is 3-5% better than GINO (29.57%), Transolver++ (28.11%), and GraphSAGE (42.60%). Performance improves steadily with training size, and random vertex subsampling enhances generalization in low-data settings by introducing geometric variability.
- **Physically consistent global prediction:** Explicit supervision of global quantities (e.g. total thrust and torque) yields the best thrust estimates, with a 2–4$\times$ reduction in global loss compared to GINO and Transolver++ models.
- **Decoder and SDF ablation:** Decoding using interpolation leads to more robust results, while the incorporation of signed distance functions (SDFs) improves geometric consistency. The combined setup achieves the best compromise between accuracy and physical consistency.

2 Related Works

Graph and Grid-Based Neural Operators. Early CFD surrogates in geometric domains naturally adopted graph neural networks (GNNs) such as GraphSAGE (Hamilton, Ying, and Leskovec 2018) and MeshGraphNet (Pfaff et al. 2021). However, they rely on hard-coded adjacency, and denser sampling may cause message passing to degenerate into a single-point operator. The Graph Neural Operator (GNO) (Li et al. 2020) mitigates this by using radius-based aggregation to approximate local kernel integration in continuous space. For regular meshes, the Fourier Neural Operator (FNO) (Li et al. 2021) performs global mixing via FFTs and achieves strong resolution-invariant performance.

The Geometry-Informed Neural Operator (GINO) (Li et al. 2020) combines the strengths of GNO and FNO by mapping irregular meshes to a latent grid for spectral convolution and projecting the results back. Building on this idea, the Factorized Implicit Global Convolution (FIGConv) (Choy et al. 2025) factorizes the latent domain into anisotropic grids, reducing memory from cubic to quadratic order and replacing spectral layers with efficient 3D convolutions. This latent-space based strategy with domain factorization is particularly promising as the building block for scalable neural architectures for complex CFD applications.

Transformer-based architectures. Transolver (Wu et al. 2024) introduces physics-domain tokenization and self-attention over geometric tokens, while Transolver++ (Luo et al. 2025) improves efficiency through memory optimization and parallelism, supporting training on millions-scale datasets. AB-UPT (Alkin et al. 2025) adopts an anchor-based design, compressing spatial information through anchor points and using cross-attention between surface and volume samples for efficient coupling.

Physics-Informed Neural Operators and Learning-Augmented Integrators. Beyond purely data-driven surrogates, multiple works integrate physics or numerical priors into operator learning. Physics-Informed Neural Operators (PINOs) (Li et al. 2023; Lin et al. 2025; Ganeshram et al. 2025) extend the idea of Physics-Informed Neural Networks (PINNs) (Raissi, Perdikaris, and Karniadakis 2019) to the operator learning paradigm by fusing coarse-resoluton data with high-resolution PDE constraints, enabling super-resolution inference and better parametric stability. In parallel, the Neural Operator Element Method (NOEM) (Ouyang et al. 2025) embeds neural operators inside the FEM pipeline to reduce meshing cost in complex subdomains.

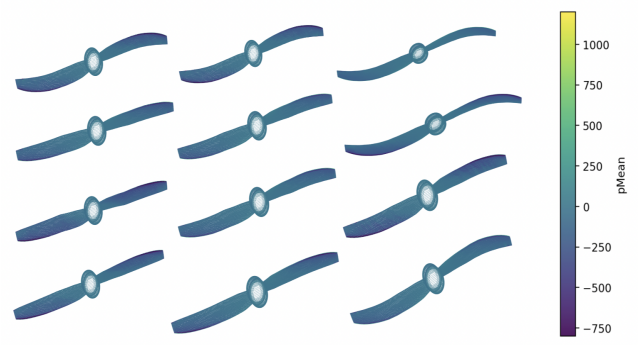

Figure 1: **Visualization of procedurally generated propeller geometries.** Each sample is colored by time-averaged pressure obtained from simulations, illustrating flow variations across different blade designs.

3 Methods

3.1 Dataset: Propeller CFD

The dataset consists of procedurally generated 3D propellers simulated using the open-source finite-volume solver OpenFOAM®. Each case is parameterized by the blades' pitch, skew and chord length at selected radial positions, from which a complete mesh is generated and evaluated under physical conditions defined by the inflow velocity and angular speed ω. To model rotating propellers, the Multiple Reference Frame (MRF) technique is employed, providing a computationally efficient way to account for the rotating frame of reference. High-performance parallel computing is used to perform Reynolds-Averaged Navier–Stokes (RANS) simulations for these propeller configurations, producing steady-state flow quantities including pressure, wall shear stress, and integrated forces (thrust and torque).

Each sample is represented as $\{v, c, u, g\}$, where $v \in \mathbb{R}^{n\times 3}$ are the spatial coordinates of the point cloud, $c \in \mathbb{R}^{d_c}$ encodes global boundary conditions, $u \in \mathbb{R}^{n\times d_u}$ denotes local quantities such as pressure and wall shear stress, and $g \in \mathbb{R}^{d_g}$ global quantities such as total thrust or torque. Surface prediction serves as the primary objective for capturing detailed flow behavior, while accurate global prediction is equally critical for downstream applications such as performance evaluation and geometry optimization. The dataset includes approximately 1000 propeller cases, each containing $\sim$70k surface points and million-scale volume points.

3.2 Factorized Implicit Global Convolutions

The Factorized Implicit Global Convolution (FIGConv) (Choy et al. 2025) is a CNN-based neural network designed to learn continuous 3D mappings. FIGConv creates a set of shared factorized latent grids $\{F_m\}_{m=1}^{M}$ for all training cases. Each grid is defined on two high-resolution axes and one low-resolution axis, effectively decomposing a dense 3D volume into several anisotropic 2.5D slices. For instance, F_1 may have dimensions $(r, W_{\max}, D_{\max})$, F_2 as $(H_{\max}, r, D_{\max})$, and F_3 as $(H_{\max}, W_{\max}, r)$, where $r \ll H_{\max}, W_{\max}, D_{\max}$. This factorization reduces voxels from cubic to roughly quadratic scale while preserving full 3D coverage. The vertex–grid encoder maps surface points onto these grids, each capturing structure along two dense axes, and their combined features reconstruct the full field.

Positional Encoding. FIGConv encodes all spatial coordinates using sinusoidal positional embeddings. These encodings provide translational and rotational consistency by representing relative offsets as frequency-domain phase shifts rather than absolute coordinates. Consequently, neighboring vertices and grid voxels exhibit smoothly varying embeddings, enabling coherent interpolation between the irregular vertex domain and the structured latent grids.

Vertex–Grid Encoder and Decoder. The encoder and decoder adopt a point convolution formulation for vertex–grid feature exchange. For encoding, each voxel center $p_{m,ijk}$ in grid F_m aggregates the features of its neighbors:

$$f_{m,ijk} = \sum_{v\in\mathcal{N}(p_{m,ijk})} \psi_m\big(\phi(v), \phi(p_{m,ijk}), \phi(v - p_{m,ijk})\big) \approx \int_{\mathcal{N}(p_{m,ijk})} \kappa_m(v, p_{m,ijk})\, \rho(v)\, dv.$$

Here,

$$\mathcal{N}(p_{m,ijk}) = \{\, v \mid \|\Sigma_m^{-1/2}(v - p_{m,ijk})\| < 1 \,\}$$

defines an ellipsoidal neighborhood centered on $p_{m,ijk}$ and determined by a covariance matrix Σ_m. This is similar to the radius-based aggregation used in GNO, but with an ellipsoidal (rather than spherical) region that aligns with the directional structure of each factorized grid. The function $\psi_m(\cdot)$ implements the kernel κ_m through an MLP, and $\rho(v)$ represents the density function of the vertex. This continuous formulation allows the encoder to integrate geometric information from irregular vertex distributions while preserving local spatial coherence. The decoder mirrors this process in the reverse direction, reconstructing vertex embeddings from nearby voxel features,

$$\hat{f}(v) = \sum_{p_{m,ijk}\in\mathcal{N}(v)} \psi'_m\big(\hat{f}_{m,ijk}, \phi(p_{m,ijk}), \phi(v - p_{m,ijk})\big).$$

This symmetric design ensures consistent information flow between the physical domain and the structured latent grid.

Latent Grid Convolution. All latent grids $\{F_m\}$ are jointly processed by a U-Net backbone composed of convolution, downsampling, and upsampling stages. Each 3D convolution applies a local kernel W_m to grid F_m, followed

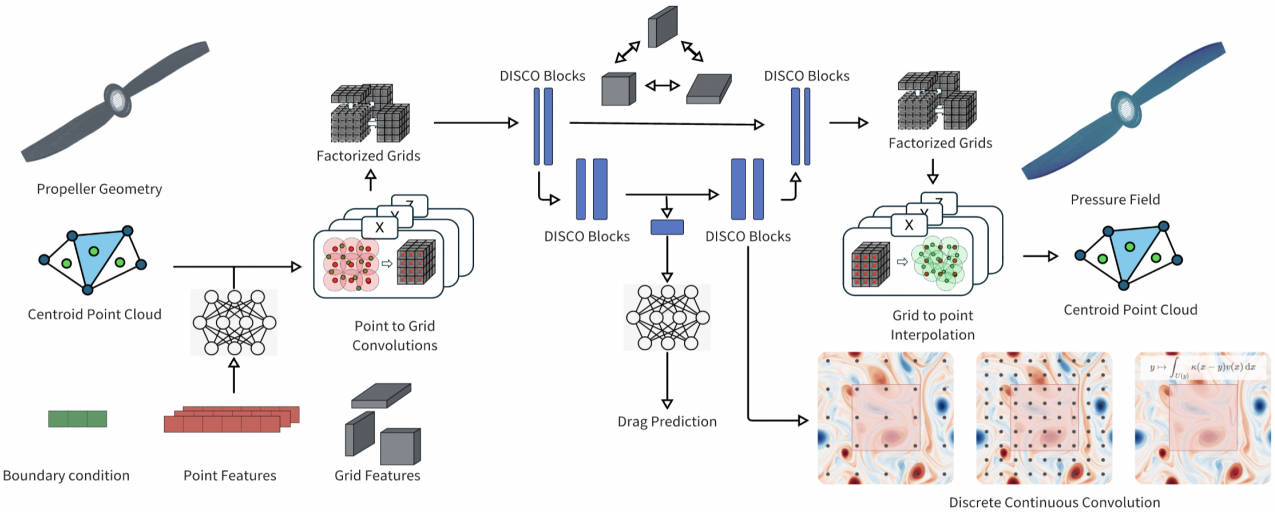

Figure 2: **The Factorized Geometry Convolutional Neural Operator (FG-ConvNO) for propeller pressure prediction.** Each propeller geometry is first represented as a centroid-based surface point cloud, where each centroid corresponds to a local patch on the propeller surface. FG-ConvNO then projects vertex features onto multiple latent grids, performs hierarchical convolutional aggregation across the grids, and decodes the resulting latent fields back to the vertex domain for pressure prediction. Compared with FIGConv, FG-ConvNO incorporates geometry-aware conditioning (using boundary conditions, SDFs, and normals), interpolation-based feature decoding, an explicit global prediction head to infer quantities such as thrust and torque, and hierarchical DISCO blocks for discretization convergence, which preserves consistent local support across different grid resolutions.

by inter-grid feature exchange that allows communication across different orientations:

$$\begin{aligned}\tilde{F}_m^{(l)} &= \mathrm{Conv3D}(F_m^{(l)}; W_m)\\ F_m^{(l+1)} &= \tilde{F}_m^{(l)} + \sum_{m' \neq m} \mathrm{Interp}\big(\tilde{F}_{m'}^{(l)}, \mathrm{coords}(F_m)\big).\end{aligned}$$

The fusion step enables the propagation of information across all factorized grids.

3.3 Factorized Geometry Convolutional Neural Operator

We extend FIGConv to the Factorized Geometry Convolutional Neural Operator (FG-ConvNO), designed to improve generalization under data scarcity and complex geometry. Its baseline configuration follows FIGConv, where only surface vertices v_s are used as input, and training is supervised by a ℓ_2 relative loss in surface pressure predictions.

Architectural Simplification. To alleviate overfitting and improve stability, FG-ConvNO reformulates FIGConv by simplifying both its hierarchical structure and the vertex–grid interaction mechanism. Specifically, we retain the multi-scale U-Net–like design from FIGConv but reduce its depth (n_{levels}) while increasing the number of convolution blocks per level (n_{down}, n_{up}). This heuristics is used to maintain the receptive field in the new configuration close to that of its original one at a lower information loss due to redundant downsampling. Second, we replace the grid-to-vertex point convolution in the decoder with an interpolation-based feature exchange implemented using trilinear sampling. For each output vertex v, the features are interpolated from all factorized grids through differentiable sampling:

$$\hat{f}(v) = \sum_{m=1}^{M} w_m \cdot \mathrm{GridSample}(F_m, \mathrm{proj}(v)),$$

where $\mathrm{GridSample}(\cdot)$ denotes trilinear interpolation, $\mathrm{proj}(v)$ maps v to the normalized coordinates of grid F_m, and w_m are learned per-grid weights. This interpolation avoids repeated kernel evaluations, significantly reducing memory consumption, and ensures spatial smoothness of the reconstructed pressure field since the interpolated features continuously vary with vertex position.

Geometry-Aware Encoding. While FIGConv only uses positional encoding for grid feature information, FG-ConvNO pursues geometric and physical awareness by exploiting multiple conditioning signals such as boundary conditions, signed distance functions (SDFs) and surface normals. We integrate them through a learned feature fusion module $\mathrm{Fuse}(\cdot)$, which can be weighted summation or attention-base mixing. For each surface vertex v and grid point $p_{m,ijk}$, we first compute their respective embeddings and then aggregate them via the fusion module:

$$\begin{aligned}\tilde{\phi}(v) &= \mathrm{Fuse}\big(\phi(v), \gamma(c), \gamma(n_v)\big),\\ \tilde{\phi}(p_{m,ijk}) &= \mathrm{Fuse}\big(\phi(p_{m,ijk}), \gamma(\mathrm{SDF}(p_{m,ijk}))\big).\end{aligned}$$

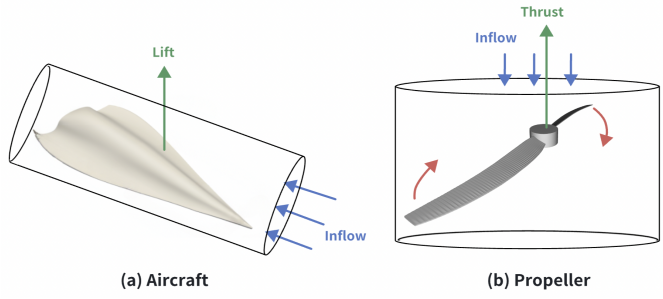

Figure 3: **Comparison of aerodynamic and propulsive forces.** (a) Aircraft wings generate lift perpendicular to the inflow. (b) Propellers produce thrust opposite to the inflow. This illustrates how global quantities such as thrust and drag integrate local pressure distributions.

This enhanced representation enables FG-ConvNO to capture geometry and boundary-aware correlations more effectively than FIGConv, improving feature consistency and generalization across diverse propeller geometries.

DISCO Blocks A standard convolution assumes that the kernel acts on grid indices rather than geometric coordinates. For a regular grid $D_h = \{z_i\}$ with spacing h, a discrete convolution takes the form

$$\mathrm{Conv}_K[v](z_i) = \sum_{t=1}^{S} K_t \, v(z_i + z_t),$$

where z_t denotes a fixed stencil offset such as $(\pm h, 0)$ or $(\pm h, \pm h)$. Both the neighbor set $\{z_i + z_t\}$ and the kernel coefficients K_t are tied to this discretization. As the grid is refined, these offsets shrink, the receptive field collapses, and the convolution converges to a pointwise operator, so CNN kernels depend inherently on the chosen resolution.

DISCO replaces this with a continuous formulation

$$(\kappa * g)(v) = \int_{B_r(v)} \kappa(u - v)\, g(u)\, du,$$

where neighbors come from a radius-based set $B_r(v)$ independent of discretization. The kernel is represented as

$$\kappa(z) = \sum_{\ell=1}^{L} \theta_\ell \, \phi_\ell(z),$$

using continuous basis functions ϕ_ℓ and trainable coefficients θ_ℓ. The integral is approximated by quadrature,

$$(\kappa * g)(v_i) \approx \sum_{u_j \in B_r(v_i)} \kappa(u_j - v_i)\, g(u_j)\, q_j,$$

so the operator evaluates continuous offsets rather than discrete grid steps. This yields a stable receptive field and makes the kernel reusable across resolutions.

In FG-ConvNO, every CNN block used in FIGConv is replaced with a DISCO block while keeping the same intra-grid communication. At each downsampling level, the latent grid resolution is halved, while the DISCO convolution radius is doubled, which ensures that each layer represents the same continuous operator despite operating on grids of different resolutions. Down blocks therefore consist of a DISCO convolution followed by average pooling for spatial halving, while up blocks apply bilinear upsampling before the DISCO convolution to restore spatial resolution. With this construction, the network maintains consistent local support and discretization invariance.

3.4 Global Prediction Objective

The local surface pressure is a measure of the fine grained flow behavior, but as far as engineering system design is concerned, the integrated global response variables (total thrust and total torque) matter the most, as they drive the performance of a propulsion device. These global forces are physically analogous to lift and drag on aircraft wings or automotive surfaces (Fig. 3), where distributed pressure fields integrate into macroscopic aerodynamic effects.

In FG-ConvNO, we predict such global quantities (e.g., thrust, torque, efficiency) directly, extending FIGConv with explicit global physical supervision. The lowest-resolution latent grid within the U-Net hierarchy serves as a compact representation of the entire flow field, from which the network infers global variables g summarizing the overall aerodynamic or hydrodynamic performance of the propeller. Formally, these quantities correspond to surface integrals of local flow variables,

$$g = \int_S \big(p(v)\, n_v + \tau(v)\big)\, dS,$$

where $p(v)$ is the surface pressure, n_v the normal vectors and $\tau(v)$ the shear-stress component. In practice, FIGConv does not perform this integration explicitly; instead, it learns to approximate this mapping directly using the feature maps of the coarsest latent grids $\{F_m^{(L)}\}$:

$$z_{\text{global}} = \mathrm{Pool}\big(\{F_m^{(L)}\}\big), \quad \hat{g} = \mathrm{MLP}(z_{\text{global}}).$$

The training loss combines global MSE $\mathcal{L}_{\text{global}} = \| \hat{g} - g \|_2^2$ and local surface supervision, i.e.,

$$\mathcal{L} = \mathcal{L}_{\text{rel}}(y_s, \hat{y}_s) + \lambda_g \| \hat{g} - g \|_2^2,$$

where both g and y_s are normalized with respect to their own statistical ranges during training to ensure scale consistency. This joint formulation links local pressure prediction with physically significant global quantities, improving overall physical consistency. This global supervision mechanism is model-agnostic and can be readily applied to other surrogate settings where dense fields must remain consistent with integral physical quantities.

4 Experiments

4.1 Experimental Setup

All experiments are conducted on the proposed propeller CFD dataset described in Section 3.1. Unless otherwise stated, the training set contains 750 samples and the test set 150 samples. Each sample includes surface vertices with the corresponding normals, local flow quantities, and global responses such as thrust and torque. Optimization uses Adam with a learning rate of 1×10^{-3}, a StepLR scheduler (step size 25, decay factor $\gamma = 0.2$), and batch size 1 for 200 epochs. We compare our method against representative neural operator architectures, including graph-based GraphSAGE (Hamilton, Ying, and Leskovec 2018), and two recent neural operator frameworks, GINO (Li et al. 2020; Kossaifi et al. 2024) and Transolver++ (Luo et al. 2025). All models receive identical geometric and boundary condition inputs and have comparable parameter counts for fair comparison.

Both FG-ConvNO variants achieve lower surface error than the graph-based baseline (GraphSAGE) and the recent operator models (GINO, Transolver++), while explicit global supervision substantially improves the accuracy of global quantity prediction. All models are implemented in PyTorch and trained on NVIDIA A100 GPUs (32GB memory).

4.2 Experimental results

We compare FG-ConvNO with three representative baselines: GraphSAGE, GINO, and Transolver++. All models are trained using only the relative surface loss $\mathcal{L}_{\text{rel}}$ unless otherwise noted, and their global quantities are computed by integrating the predicted surface pressure.

$$\hat{g} = \int_S \hat{p}(v)\, n_v \, dS.$$

For FG-ConvNO, we evaluate two variants: (1) *surface-only* training using $\mathcal{L}_{\text{rel}}$, and (2) *joint* training with both $\mathcal{L}_{\text{rel}}$ and $\mathcal{L}_{\text{global}}$. All variants include geometry-aware inputs (normals, boundary conditions, and grid SDFs).

Table 1: **Performance comparison with existing architectures on the propeller dataset.** Both FG-ConvNO variants achieve lower surface error than the graph-based baseline (GraphSAGE) and the recent operator models (GINO and Transolver++), while explicit global supervision substantially improves the accuracy of global quantity prediction.

Model	Surface ↓	Global ↓
GraphSAGE	42.47%	8.32×10^{-3}
Transolver++	27.96%	2.41×10^{-3}
GINO	29.16%	3.15×10^{-3}
FG-ConvNO (surface only)	**23.68%**	1.76×10^{-3}
FG-ConvNO (surface + global)	24.07%	$\mathbf{3.43 \times 10^{-4}}$

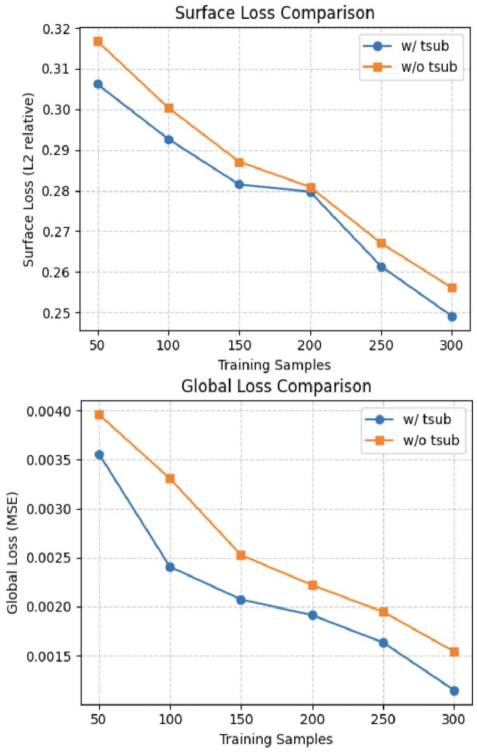

Figure 4: **Ablation on dataset size and random vertex subsampling for training (tsub)**. Increasing training samples reduces both surface and global losses, confirming the importance of data scale. Random subsampling consistently improves global prediction and provides stronger gains on surface accuracy when training data are limited.

Table 1 shows that FG-ConvNO consistently outperforms other recent architectures (GraphSAGE, GINO and Transolver++) in surface prediction, and achieves the lowest overall global loss when trained with the global objective. This shows that incorporating the global loss $\mathcal{L}_{\text{global}}$ effectively couples local pressure estimation with integral physical consistency.

4.3 Ablation Studies

To gain better insight into the effect of each design component, we perform comprehensive ablation studies to investigate the roles played by data scale, random vertex subsampling, and network architecture choice. This provides further insights into the robustness and generalization properties of the proposed approach in different data regimes.

Data Size and Random Sampling. We train FG-ConvNO (surface + global) using different dataset sizes (50–300 samples), with and without random vertex subsampling. Fig. 4 shows that both local and global losses decrease monotonically as the size of training set increases, indicating that the scale of dataset plays a crucial role in learning with complicated meshing geometries. Such larger datasets benefit the

model by making overfitting harder as well as enabling it to more effectively learn higher order geometric and physical relationships that are not well sampled in small data regimes. Random vertex sub-sampling contributes to better generalization by introducing the model to various geometries and point distributions during training iterations. This encourages the network to rely on physically meaningful flow patterns rather than memorizing mesh-specific correlations. Together, these results highlight that increasing geometric diversity is essential for robust and physically consistent surrogate modeling in data-scarce CFD settings.

U-Net Depth and Block Count. We investigate how hierarchical depth (n_{levels}) and intra-stage convolutional capacity ($n_{\text{up}}, n_{\text{down}}$) affect performance. Figure 5 presents quantitative comparisons of surface loss, global loss, and GPU memory usage across six configurations, as well as qualitative visualizations of surface pressure prediction and error maps. here, all models share the same training configuration and input encoding. Across all tested configurations, the $(1, 2)$ setting offers the best balance between accuracy and efficiency. Shallower models lack contextual coverage, while deeper hierarchies oversmooth local features and incur unnecessary cost. A shallow hierarchy with richer intra-level blocks therefore provides a more effective inductive bias under limited data. These results indicate that increasing hierarchical depth beyond a moderate level does not yield consistent gains in accuracy but instead promotes underfitting due to excessive downsampling.

This aligns with our broader finding that a shallow hierarchy with richer intra-level capacity provides a more effective inductive bias for learning complex propeller flows under limited data. This configuration maintains a good balance between local precision and global consistency while remaining efficient and stable during training.

Decoder and Grid SDF Variants. We further evaluate the influence of decoder formulation and grid SDF conditioning. Three configurations are compared: (1) point convolution decoder with the SDF input, (2) interpolation-based decoder without SDFs, and (3) the proposed model combined with interpolation and SDF. All models are trained using $\mathcal{L}_{\text{global}}$. The results are shown in Table 2.

Table 2: **Ablation on decoder formulation and grid SDF conditioning.** Interpolation enhances smoothness and stability, while SDF features improve geometric consistency.

Configuration	Surface ↓	Global ↓
PointConv (w/ Grid SDF)	45.07%	7.67×10^{-3}
Interp (w/o Grid SDF)	26.85%	9.72×10^{-4}
Proposed	**23.68%**	$\mathbf{3.43 \times 10^{-4}}$

The interpolation-based decoder substantially improves training stability and surface smoothness compared to the original Point Convolution. Point-based decoding is highly sensitive to irregular neighborhood patterns, which vary sharply across propeller blades and often introduce inconsistent or noisy reconstructions. Our decoder, by contrast, provides a stable and geometry-agnostic mapping whose output changes smoothly with vertex position, making it inherently more robust for transferring latent grid features back to complex surfaces.

Meanwhile, incorporating grid SDF features introduces richer geometric context into the latent representation. Unlike positional encodings that only describe relative coordinates, SDF features offer a continuous measure of distance to the surface, helping differentiate near-surface regions from free space and aligning pressure predictions with blade geometry. This lightweight geometric prior improves physical consistency.

Thus, combining interpolation-based decoding with SDF-aware conditioning yields smoother, more coherent predictions that remain geometrically and physically aligned with the underlying flow structure.

Super-Resolution Inference. To evaluate whether FG-ConvNO inherits the resolution-agnostic behavior introduced by DISCO layers. We train the model with the default factorized grid configuration (3×100×200),(150×2×200),(150×100×4), and evaluate it at inference time with a 1.5× finer resolution (applied only to the uncompressed axes)

The resulting surface error differs by only 0.1% from the native-resolution evaluation, indicating that DISCO blocks maintains stable behavior across grid scales. This resolution robustness is expected to become even more useful for larger datasets or geometries with stronger multiscale features, where higher-resolution latent grids may capture finer flow variations without sacrificing stability.

5 Conclusion

We proposed the Factorized Geometry Convolutional Neural Operator for propeller CFD prediction under limited data. FG-ConvNO simplifies the Convolution hierarchy and the decoding process, incorporates geometry aware features such as SDFs and normals, introduces a global prediction head for thrust and torque, and replaces resolution dependent CNN kernels with discrete continuous convolutions. These components jointly improve stability, discretization consistency and generalization.

Across experiments, FG-ConvNO achieves the best accuracy on both surface pressure and global thrust prediction, while exhibiting stable training behavior. Ablation studies further show that geometric conditioning, interpolation-

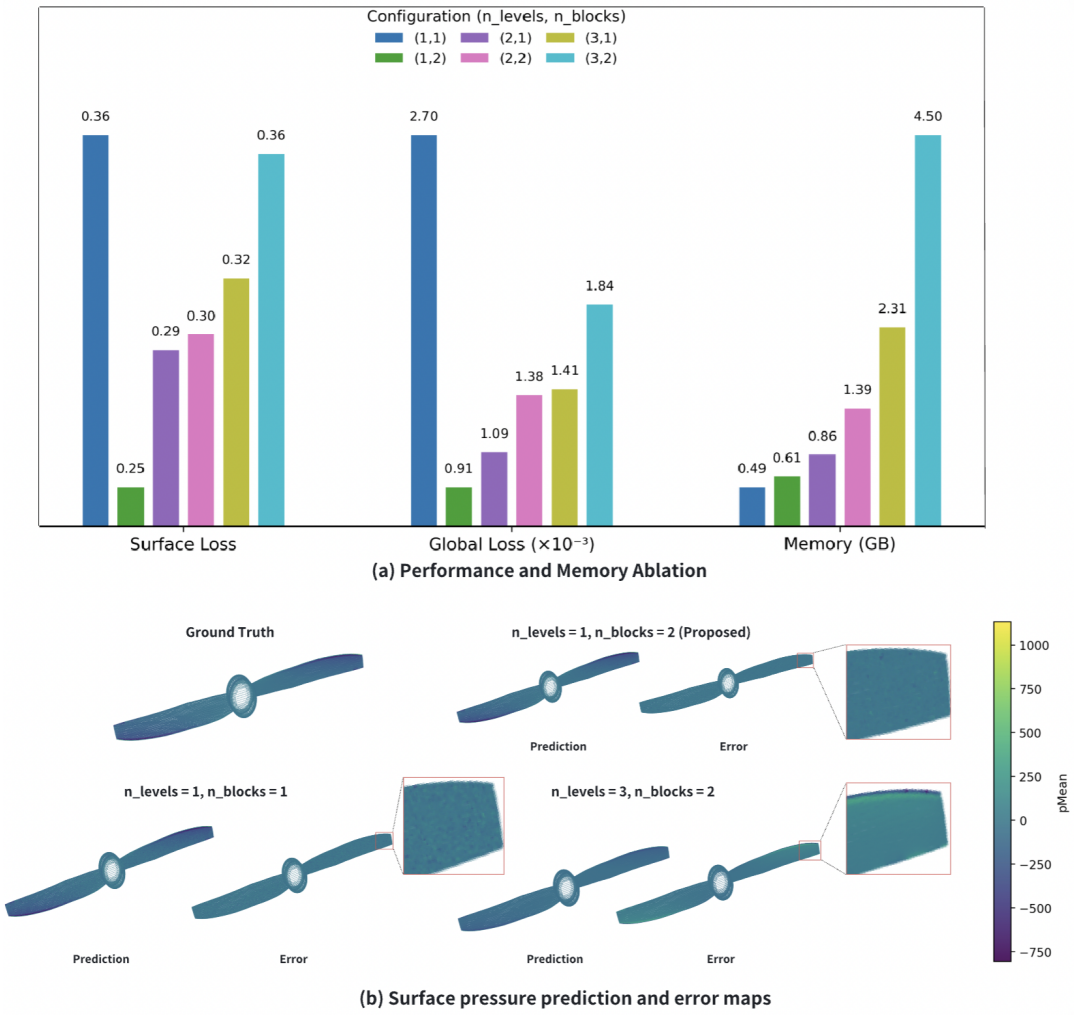

Figure 5: **Quantitative and qualitative ablation of U-Net hierarchy.** (a) Comparison of surface and global losses and GPU memory usage across six configurations of n_{levels} and block counts $(n_{\text{up}}, n_{\text{down}})$. (b) Example predictions and error maps for representative models $(1, 1)$, $(1, 2)$, and $(3, 2)$.

based decoding, and balanced hierarchical depth collectively contribute to robust learning in complex, high-curvature geometries. Together, these design choices establish a coherent principle: coupling lightweight architectures with geometry-informed representations yields robust and generalizable CFD surrogates, even under limited data availability.

Future Directions. Currently, the dataset size remains relatively limited, constraining the full exploration of complex geometric–physical correlations. We intend to produce a significantly larger and more diverse dataset of propeller geometries, boundary conditions, and operation regimes, so we can better evaluate model performance and the extrapolation capability at different levels of generality.

Exploring richer geometric encodings, such as curvature-based descriptors or implicit surface representations, could further improve the model's understanding of complex 3D structures. Additionally, integrating physics-informed regularization or differentiable solver coupling offers a potential path toward physically constrained and transferable neural operators applicable to a broader class of CFD problems.

In this paper, we consider only steady-state propeller flows and an extension of FIGConv to unsteady or turbulent ones is an interesting future direction. Temporal coupling through recurrent or transformer-based operators could enable dynamic flow prediction and wake evolution modeling, bridging the gap between steady and time-resolved behavior.

Finally, while the presented approach shows promising performance for complex propeller geometries in the small-data regime, future work will involve benchmarking with more canonical aerodynamic and hydrodynamic shapes, such as car bodies and airfoils, to further test generalization and robustness across geometric domains.

Acknowledgments

This work was supported in part by the Bren Endowed Chair at Caltech, NVIDIA research support, and computing resources provided by the Caltech High-Performance Computing Center. We also thank NVIDIA for additional GPU resources used in this work.

References

Alkin, B.; Bleeker, M.; Kurle, R.; Kronlachner, T.; Sonnleitner, R.; Dorfer, M.; and Brandstetter, J. 2025. AB-UPT: Scaling Neural CFD Surrogates for High-Fidelity Automotive Aerodynamics Simulations via Anchored-Branched Universal Physics Transformers. arXiv:2502.09692.

Azizzadenesheli, K.; Kovachki, N.; Li, Z.; Liu-Schiaffini, M.; Kossaifi, J.; and Anandkumar, A. 2024. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 1–9.

Berner, J.; Liu-Schiaffini, M.; Kossaifi, J.; Duruisseaux, V.; Bonev, B.; Azizzadenesheli, K.; and Anandkumar, A. 2025. Principled Approaches for Extending Neural Architectures to Function Spaces for Operator Learning.

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; and Yu, F. 2015. ShapeNet: An Information-Rich 3D Model Repository. arXiv:1512.03012.

Choy, C.; Kamenev, A.; Kossaifi, J.; Rietmann, M.; Kautz, J.; and Azizzadenesheli, K. 2025. Factorized Implicit Global Convolution for Automotive Computational Fluid Dynamics Prediction. arXiv:2502.04317.

Ganeshram, A.; Maust, H.; Duruisseaux, V.; Li, Z.; Wang, Y.; Leibovici, D.; Bruno, O.; Hou, T.; and Anandkumar, A. 2025. FC-PINO: High Precision Physics-Informed Neural Operators via Fourier Continuation. arXiv:2211.15960.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2018. Inductive Representation Learning on Large Graphs. arXiv:1706.02216.

Kossaifi, J.; Kovachki, N.; Li, Z.; Pitt, D.; Liu-Schiaffini, M.; Duruisseaux, V.; George, R. J.; Bonev, B.; Azizzadenesheli, K.; Berner, J.; and Anandkumar, A. 2024. A Library for Learning Neural Operators. arXiv:2412.10354.

Kovachki, N.; Li, Z.; Liu, B.; Azizzadenesheli, K.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2023. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89): 1–97.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2020. Neural Operator: Graph Kernel Network for Partial Differential Equations. arXiv:2003.03485.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2021. Fourier Neural Operator for Parametric Partial Differential Equations. arXiv:2010.08895.

Li, Z.; Zheng, H.; Kovachki, N.; Jin, D.; Chen, H.; Liu, B.; Azizzadenesheli, K.; and Anandkumar, A. 2023. Physics-Informed Neural Operator for Learning Partial Differential Equations. arXiv:2111.03794.

Lin, R. Y.; Berner, J.; Duruisseaux, V.; Pitt, D.; Leibovici, D.; Kossaifi, J.; Azizzadenesheli, K.; and Anandkumar, A. 2025. Enabling Automatic Differentiation with Mollified Graph Neural Operators. arXiv:2504.08277.

Liu-Schiaffini, M.; Berner, J.; Bonev, B.; Kurth, T.; Azizzadenesheli, K.; and Anandkumar, A. 2024. Neural Operators with Localized Integral and Differential Kernels. arXiv:2402.16845.

Luo, H.; Wu, H.; Zhou, H.; Xing, L.; Di, Y.; Wang, J.; and Long, M. 2025. Transolver++: An Accurate Neural Solver for PDEs on Million-Scale Geometries. arXiv:2502.02414.

Ouyang, W.; Shin, Y.; Liu, S.-W.; and Lu, L. 2025. Neural-operator element method: Efficient and scalable finite element method enabled by reusable neural operators. arXiv:2506.18427.

Pfaff, T.; Fortunato, M.; Sanchez-Gonzalez, A.; and Battaglia, P. W. 2021. Learning Mesh-Based Simulation with Graph Networks. arXiv:2010.03409.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378: 686–707.

Umetani, N.; and Bickel, B. 2018. Learning three-dimensional flow for interactive aerodynamic design. *ACM Trans. Graph.*, 37(4).

Wang, H.; Cao, Y.; Huang, Z.; Liu, Y.; Hu, P.; Luo, X.; Song, Z.; Zhao, W.; Liu, J.; Sun, J.; Zhang, S.; Wei, L.; Wang, Y.; Wu, T.; Ma, Z.-M.; and Sun, Y. 2024. Recent Advances on Machine Learning for Computational Fluid Dynamics: A Survey. arXiv:2408.12171.

Wu, H.; Luo, H.; Wang, H.; Wang, J.; and Long, M. 2024. Transolver: A Fast Transformer Solver for PDEs on General Geometries. arXiv:2402.02366.

A Additional Dataset and Simulation Details

Each propeller case was simulated using the open-source CFD solver OpenFOAM® under steady RANS conditions with the k–ω SST turbulence model. The inflow velocity ranged from 3–10 m/s, and the angular velocity ω was fixed to 4,000 rpm. Meshes contained approximately 100k–150k cells, with refined boundary layers near the blade surface to ensure accurate wall-shear stress resolution. Training, validation, and test splits followed a 4:1 ratio over 900 total cases.

For data preprocessing, all surface vertex coordinates were normalized independently along each spatial dimension using the global bounding box of the entire dataset, mapping each axis range to $(0, 1)$. This ensures consistent geometric scaling across different propeller configurations and stabilizes the vertex–grid interpolation during training. Surface pressure values were standardized by Gaussian normalization, where both the mean and standard deviation were calculated over the training set.

During inference, both model outputs and ground-truth pressures were de-normalized before computing evaluation losses to ensure physical comparability. The same normalization and de-normalization procedures were applied to global quantities such as thrust and torque to maintain consistency between local and integral supervision.

B Additional Network Architectures Details

We summarize the architectural configurations of all compared models, including the proposed FG-ConvNO, implemented uniformly in PyTorch for fair comparison. All models share the same input normalization, conditioning pipeline, and supervision structure, with local (pressure) and global (thrust/torque) targets. Although the compared methods differ in how spatial dependencies are modeled, the proposed FIGConv introduces a hybrid vertex–grid interaction that bridges continuous and discretized representations.

B.1 Common Design Philosophy.

All networks take as input the normalized vertex coordinates (and optional condition scalars) and output the predicted surface pressures. This consistent design ensures that the differences in accuracy originate purely from architectural capacity, rather than differences in feature design or supervision.

B.2 Model Configurations.

Below we provide concise configuration-style summaries for the principal models evaluated in this study. Each follows a common notation to facilitate direct comparison.

GraphSAGE serves as a lightweight graph-based baseline using mean aggregation. Its neighborhood sampling of $[20, 10, 5]$ provides a moderate receptive field suitable for smooth CFD fields.

```
class GraphSAGEModelConfig:
    model_arch: str = "graphsage"
    in_channels: int = 3
    out_channels: int = 1
    hidden_channels: List[int] = [128, 128, 64]
    num_layers: int = 3
    aggr_type: str = "mean"
    normalize: bool = True
    dropout: float = 0.1
    activation: str = "relu"
    num_neighbors: List[int] = [20, 10, 5]
    use_self_loops: bool = True
    residual: bool = True
    global_pool: str = "mean"
    global_head_dim: int = 64
```

GINO combines graph operators with Fourier-based global mixing layers. The low latent resolution (24^3) ensures memory efficiency while retaining expressiveness.

```
class GINOModelConfig:
    in_channels: int = 3
    out_channels: int = 1
    in_gno_radius: int = 3
    out_gno_radius: int = 3
    fno_hidden_channels: int = 64
    fno_n_layers: int = 4
    fno_lifting_channel_ratio: int = 2
    latent_resolution: (24, 24, 24)
```

Transolver++ adopts a transformer-style attention mechanism over discretized 3D slices. This architecture is powerful for capturing long-range dependencies.

```
class TransolverPlusModelConfig:
    space_dim: int = 3
    n_layers: int = 5
    n_hidden: int = 256
    n_head: int = 8
    mlp_ratio: int = 1
    act: str = "gelu"
    slice_num: int = 64
```

FG-ConvNO introduces a hybrid feature interaction mechanism between irregular vertex sets and structured latent grids. Unlike conventional message-passing GNNs, FIGConv performs bidirectional vertex–grid convolutions, enabling efficient spatial reasoning in both local and global contexts.

```
class FIGConvModelConfig:
    out_channels: int = 1
    kernel_size: int = 5
    hidden_channels: [96, 128]
    num_levels: int = 1
    vertex_hidden_dim: int = 96
    feature_dim: int = 3
    feature_hidden_dim: int = 32
    condition_dim: int = 1
    condition_hidden_dim: int = 32
    grid_extra_dim: int = 1
    grid_extra_hidden_dim: int = 64
    pos_encode_dim: int = 32
    num_down_blocks: int = 2
    num_up_blocks: int = 2
    radius_cutoff: float = 0.1
    mlp_channels: [256, 256]
    neighbor_search_radius: float = 1.71
    to_point_sample_method: "interp"
    geometry_fusion_type: "weighted_sum"
```

B.3 Latent Grid Decomposition.

The latent grid configuration for all FIGConv variants followed a factorized anisotropic design to align with the spatial characteristics of propeller flows. Specifically, the resolution–format pairs were defined as (3, 100, 200), (150, 2, 200) and (150, 100, 4), representing three orthogonally oriented grids, each emphasizing two high-resolution axes and one compressed axis. This anisotropic decomposition reflects the physical density distribution of the flow, which is denser near the blades and more uniform along the rotation axis, enabling the model to allocate more representational capacity to regions with higher flow complexity. Empirically, this setup achieves an optimal trade-off between spatial expressiveness and computational cost, effectively preserving local vortical structures while maintaining global coherence.

B.4 Vertex–Grid Encoder and Neighborhood Search.

Feature exchange between the vertex domain and the latent grids is implemented through a continuous kernel-based formulation. For each grid center $p_{m,ijk}$ in grid F_m, the encoded feature is

$$f_{m,ijk} = \sum_{v \in \mathcal{N}(p_{m,ijk})} \psi_m\big(\phi(v), \phi(p_{m,ijk}), \phi(v - p_{m,ijk})\big)$$

where $\mathcal{N}(p_{m,ijk}) = \{v \mid \|v - p_{m,ijk}\|_2 < r_{\text{search}}\}$ defines a spatial neighborhood of radius r_{search}. Setting $r_{\text{search}} = 1.71$ allows each grid cell to access all vertices within the surrounding $2 \times 2 \times 2$ voxel cube, which corresponds to roughly eight adjacent cells, achieving smooth yet localized aggregation. This radius offers an empirically and physically grounded balance between spatial coverage and locality: smaller radii produce fragmented representations, while larger values over-smooth high-frequency flow features.

C Training and Optimization Settings

All models were trained under identical optimization schedules for consistent comparison. The optimizer was Adam with an initial learning rate of 10^{-3}, decayed by a factor of 0.2 every 25 epochs using the `StepLR` scheduler. Weight decay was set to 10^{-5}. Each model trained for 200 epochs with batch size 1 on a single NVIDIA A100 (32GB) GPU. The global supervision weight λ_g for integral losses was fixed to 1.0. Training typically required 10–12 hours per model depending on architecture complexity.

During training, both local and global objectives were optimized jointly:

$$\mathcal{L} = \mathcal{L}_{\text{local}} + \lambda_g \, \mathcal{L}_{\text{global}},$$

where $\mathcal{L}_{\text{local}}$ denotes vertex-wise mean-squared error on normalized pressures, and $\mathcal{L}_{\text{global}}$ measures error on de-normalized integral quantities (e.g., thrust and torque). This hybrid objective stabilizes training and ensures consistency between local flow reconstruction and global aerodynamic performance.

Data augmentation was applied by random vertex subsampling (ratio 0.6–0.8) to improve robustness under sparse or noisy geometric sampling. All experiments were repeated with three random seeds to report mean performance and standard deviation.