VortexNet: A Graph Neural Network-Based Multi-Fidelity Surrogate Model for Field Predictions

Yiren Shen*, Jacob T. Needels†, and Juan J. Alonso‡
Stanford University, Stanford, CA 94305, USA

Aircraft design relies on accurate aerodynamic predictions. High-fidelity (HF) methods, such as computational fluid dynamics (CFD), provide accurate aerodynamic analyses but are computationally expensive for early-stage design. Conversely, low-fidelity (LF) methods, such as the vortex lattice method (VLM), offer cost-effective solutions but struggle to capture complex flow phenomena, limiting their predictive accuracy. The conceptual design process thus presents a fidelity-cost trade-off, requiring a balance between high-fidelity (HF) and LF data in early design phases. This study introduces `VortexNet`, a graph neural network (GNN)-based surrogate model designed to bridge the fidelity gap between LF and HF aerodynamic predictions. `VortexNet` learns corrections to LF panel-wise local loading coefficient field data using data-driven insights from HF CFD simulations, enabling pressure coefficient field predictions across a range of Delta wing geometries and free-stream conditions. The model demonstrates strong prediction accuracy and generalizability across configurations, effectively capturing nonlinear flow features under geometric variations. A hyper-parameter sensitivity study and a preliminary prediction mechanism explanation, leveraging the latent space ablation technique, are conducted to rationalize the model's predictive capabilities and provide guidance for future improvements in `VortexNet`-like surrogate modeling. These results indicate that `VortexNet` has potential as a valuable tool for conceptual design in multidisciplinary design optimization (MDO), while emphasizing the need for further validation and refinement.

Nomenclature

AOA	=	Angle of Attack [degree]
b	=	span length [m]
c	=	chord length [m]
C_p	=	pressure coefficient
ΔC_p	=	local loading coefficient
ΔC_p^{HF}	=	high-fidelity (HF) ΔC_p
ΔC_p^{LF}	=	low-fidelity (LF) ΔC_p
ΔC_p^{pred}	=	surrogate model predicted ΔC_p
$\mathcal{F}$	=	a generic mapping function
Γ	=	vortex strength [m^2/s]
$G(V, E)$	=	a graph with nodes V and edges E
$\mathbf{h}$	=	a generic feature array in surrogate modeling
$\mathcal{L}$	=	a generic loss term
Ma	=	free-stream Mach number
p	=	pressure [Pa]
q_{ref}	=	reference pressure [Pa]
q^{LF}	=	low-fidelity (LF) field vector
q^{HF}	=	high-fidelity (HF) field vector
Re	=	Reynolds number
R^2	=	coefficient of determination
$\bar{v} = [u, v, w]^\intercal$	=	velocity vector [m/s]
x, y, z	=	Cartesian coordinate
y^+	=	dimensionless grid-to-wall spacing

I. Introduction

Aircraft design involves computational modeling to predict and optimize vehicle aerodynamic performance. Accurate simulation can be accomplished using high-fidelity (HF) computational fluid dynamics (CFD) codes to solve the Navier-Stokes equations, a set of coupled nonlinear partial differential equations (PDE), which govern viscous,

*Ph.D. Candidate, Department of Aeronautics and Astronautics, AIAA Student Member.
†Ph.D. Candidate, Department of Aeronautics and Astronautics, now Postdoctoral Appointee at Sandia National Laboratories, AIAA Member.
‡Vance D. and Arlene C. Coffman Professor, Department of Aeronautics and Astronautics, AIAA Fellow.

compressible flow. Despite significant progress in CFD techniques and high-performance computing hardware, CFD remains computationally expensive [1, 2], and this computational cost is exacerbated in many-query applications, such as design optimization, where a large number of flow-field evaluations across a wide range of flight conditions and vehicle configurations are required. Such costs generally prohibit relying exclusively on CFD for the conceptual design phase of an aircraft. During this phase, engineers and researchers frequently utilize low-fidelity (LF) methods, subject matter expert knowledge, and historical data to explore the design space and draft an initial design [3, 4]. As the aircraft configuration matures, the candidate designs are assessed and revised with higher fidelity tools, including CFD and wind tunnel testing. This process carries the risk that inaccuracies introduced by lower fidelity tools will result in a non-optimal configuration, requiring significant revision in later stages and adversely impacting development cost and schedule. Improved accuracy in conceptual level design tools has the potential to improve prediction quality of, and confidence in, configuration performance, and motivates the introduction of HF design tools earlier in the design cycle. However, constraints on computational cost necessitate the development of strategies which can accurately augment LF model predictions across the joint design-mission space while minimizing required computational cost.

A. Motivation

Recent interest in multidisciplinary design optimization (MDO) [5, 6] calls for a more closely integrated design-assessment cycle. Many conceptual aircraft design suites have been developed to address this need, such as `SUAVE` [7, 8]. These software suites provide aircraft design analysis and optimization frameworks that incorporate hierarchical multi-fidelity (MF) analyses across multiple disciplines. At the lowest fidelity, pressure distribution and force and moment coefficients are typically computed using a vortex lattice method (VLM) code [9–12] due to its relative simplicity and low computational cost. The VLM models the pressure distribution on the aircraft based on potential flow theory, neglecting compressibility, viscous effects, and aspects of geometry such as wing thickness. Despite limitations on accuracy, vehicle configurations, and free-stream condition trust region, the VLM has been demonstrated to be an accurate LF model for preliminary design configuration changes [13].

Despite the relative popularity of the VLM, its underlying assumptions make it incapable to model certain physical phenomena. For example, the VLM prediction of the pitching moment of a Delta wing at high angle of attack (AOA) can be unreliable because nonlinear effects, such as vortex lift, vortex-boundary layer interactions, and flow separation, are not captured by the VLM model. Under such flow conditions, the force and moment predictions from VLM are largely inaccurate. Relying on VLM for conceptual vehicle design may lead to a non-optimal or infeasible design for this phase of flights. An option to improve VLM prediction accuracy is to incorporate theoretical or semi-empirical models, such as the Polhamus correction [14], into VLM solvers. However, this approach has drawbacks regarding the accuracy and platform applicability: correction parameters need to be tuned for specific wing platforms and flow mechanism to produce accurate predictions [15, 16]. When a large design space is explored in a conceptual design task, validating the correction parameters themselves across the design space can be a complicated task. Alternatively, rather than adding additional correction terms to LF models, a MF, data-driven surrogate model can be used to correct correct LF predictions. Such frameworks are attractive for being extensible to different problems with accuracy comparable to the highest fidelity analysis tool, given sufficient samples and training.

B. Background

One common surrogate modeling approach is to learn corrections to scalar-valued aerodynamic force and moment coefficients through Bayesian frameworks, such as through MF Gaussian processes regression (GPR), to synthesize multiple data sources, allowing lower-fidelity solver results to be calibrated by higher-fidelity solver predictions with lower uncertainty [17]. Mukhopadhaya et al. demonstrated such approach to generate multi-fidelity probabilistic aerodynamic databases for simulation of commercial aircraft, employing data from panel codes, CFD, and wind tunnel experiments. `SUAVE` has a built-in module to accommodate MF data sources and build surrogate models for design optimization purposes, demonstrating a good balance between computational efficiency and accuracy [8]. However, standard GPR is limited to scalar-valued quantities of interest, making it challenging to apply to high-dimensional data like flow fields or surface pressures. In many applications, accurate prediction of field data is necessary, such as to location dependent phenomena, such as turbulent transition and flow separation, or to allow the integration of arbitrary Quantities of interest (QOI) unknown *a priori*. The ability of the surrogate to produce field data predictions also aids in the interpretability of results, improving understanding and confidence in predictions.

Popular techniques for surrogate modeling of aerodynamic field data frequently utilize proper orthogonal decomposition (POD) [19]. By decomposing the flow-field into a set of spatial modes, POD permits a parameterized model

of field variables based on the selection of modal frequencies for a truncated expansion. One popular methodology is POD-kriging, where modal coefficients are selected using GPR. This method has been previously applied to aerodynamic design optimization of passenger cars [20]. However, challenges remain for POD-based methods. A large number of modes are required to capture nonlinear phenomena, including moving coherent structures in the flow-field. Incorporation of MF data is also challenging, particularly when HF data is limited and contains distinct features not present at lower fidelities. While Mrosek et al. demonstrates the application of POD-kriging to design optimization, generalizability between configurations is a concern. This is particularly the case when the design space is large and geometric variations between designs are significant, which may limit the capability of POD-kriging to provide accurate predictions across geometries.

Rather than relying on linear projection for dimensionality reduction, other approaches utilize machine learning (ML) techniques for surrogate modeling. One popular approach is the use of deep neural network (DNN) to model aerodynamic QOIs based on MF data sources. This method has been applied to airfoil and wing sections design optimization and outperformed GPR-based approaches [21], but is limited to scalar-valued prediction. Flow-field prediction capability is also demonstrated by using a DNN to learn the discrepancy between LF Euler and HF Reynolds-averaged Navier-Stokes (RANS) flow fields [2], but for a fixed geometry.

To adequately extend VLM predictions for the case of interest, analysis and optimization of Delta wing configurations at high AOA, both a field prediction capability and strong geometric generalizability are required. To overcome limitations of previous methods, we explored approaches involving a lattice-wise vortex strength modification based on CFD results. Instead of relying on semi-empirical models, we propose using GNN to learn a mapping that calibrates LF predictions using HF CFD results. Recent studies show that GNN has strong potential for this type of mapping, offering superior performance compared to traditional methods [22–28]. Due to their graph nature, GNN are highly effective at integrating and learning mappings between different fidelity data sources, including field data sampled at different spatial locations [23, 29, 30]. Additionally, complex geometries and physical constraints can be efficiently represented in graph, even with underlying geometries variations, and thus GNN demonstrates superior generalizability to geometry changes, scalability, and efficiency [23, 24, 31]. These characteristics will be critical for MDO applications as the vehicle configuration, mesh discretization, free-stream conditions, and the QOIs may change during the design-assessment cycle.

C. Overview

Motivated by these research advancements, this study aims to enhance the accuracy of VLM in conceptual aircraft design by proposing a GNN-based mapping function to correct lattice vortex strength using HF CFD data. The proposed network will be tested with a benchmark case of a Delta wing at various AOAs. In Section II, an overview of the MF analysis tools to generate a training dataset are given, as well as the GNN architecture and training procedure used in this work. Section III presents network prediction quality for aerodynamic loading across a range of vehicle geometries. A comprehensive study of how input graph features are propagated through latent layers is conducted, to better understand the relation between network architecture and prediction quality. The paper concludes with an overview of results and directions for future work.

II. Methodology

To construct the proposed GNN-based MF surrogate model, pressure distribution data is used from two models of different fidelity. LF data is generated using the panel-wise pressure distribution computed by a vortex lattice method (VLM) solver integrated in the `SUAVE` software package. For HF data, we use the surface pressure coefficient computed by a computational fluid dynamics (CFD) Reynolds-averaged Navier-Stokes (RANS) solver available in `SU2`. A brief summary of the physical and numerical models for each mehod is presented in Section II.A for VLM and Section II.B for CFD. A detailed description of the dataset used to train the surrogate model, along with the dataset preparation procedure and the graph definition, is covered in Section II.C. Finally, the graph neural network (GNN) surrogate model's network architecture and associated training procedure is explained in Section II.D.

A. Low-Fidelity Solver: Vortex Lattice Method

The vortex lattice method models flow over an aircraft as a lattice of bound vortices to solve for the local loading on each panel by computing the vortex strengths that satisfy the boundary condition of zero normal flow on panels. In this study, a Python-based implementation of the `VORLAX` code [11], shipped with the `SUAVE` package [7, 8], is utilized. This

VLM uses infinitesimally thin, cambered or uncambered panels to represent the base geometry, and is applicable to "compressibility-corrected," inviscid, attached flows for both subsonic and supersonic conditions.

The computation of the local loading coefficient (ΔC_p), which is a normalized pressure difference between the lower and upper surfaces of a wing, begins with solving the linear system of filament vortex strength (Γ_i), geometric relationship among the lattice, and the boundary conditions of an oncoming flow:

$$AIC\,\mathbf{\Gamma} = RHS, \tag{1}$$

where Γ a vector of the vortex filament strength, AIC is the aerodynamic influence coefficient matrix representing the induced velocity contributions from each vortex filaments to each panel's control points, and RHS represents the free-stream flow contributions to the boundary conditions.

Once Γ is solved, the local induced velocities u and v can be solved using the Biot-Savart Law. The local loading coefficient is thus:

$$C_p = -\frac{2}{q_\infty^2}(U_\infty u + V_\infty v), \tag{2}$$

where U_∞ and V_∞ are the components of the free-stream velocity vector of magnitude q_∞, and u and v are the induced velocity components. C_p is the field pressure coefficient ($C_p = (P - P_\infty)/q_\infty$). In our study, we use infinitesimally thin panels, such that the pressure difference between the lower and upper wing surfaces, ΔC_p, is the C_p computed from Equation 2. In the current study, we take this ΔC_p as the pressure field predicted by LF method.

Total lift and induced drag can then be calculated by summing up all panels' contribution. The details of the VLM implementation, along with leading edge suction correction methods for Delta wing aircraft, can be found in the original `VORLAX` technical report [11].

B. High-Fidelity Solver: Computational Fluid Dynamics

`SU2` is a computational fluid dynamics (CFD) software used for the solution of partial differential equations (PDEs) and PDE-constrained optimization problems on unstructured numerical grids [32–34]. To simulate compressible flows, `SU2` solves the Navier-Stokes equations expressed in differential form as

$$\mathcal{R}(U) = \frac{\partial U}{\partial t} + \nabla \cdot \bar{F}^c(U) - \nabla \cdot \bar{F}^v(U, \nabla U) - S = 0. \tag{3}$$

The vector of conserved variables, U, can be expressed as

$$U = [\rho, \rho\bar{v}, \rho E]^\mathsf{T} \tag{4}$$

,

where ρ is the fluid density, $\bar{v} = [u, v, w]^\mathsf{T}$ is the flow speed in Cartesian system of reference, E is total energy per unit mass, and S is a generic source term (zero in the absence of external body forces). The convective and viscous fluxes are given by

$$\bar{F}^c = \begin{bmatrix} \rho\bar{v} \\ \rho\bar{v} \otimes \bar{v} + \bar{\bar{I}}p \\ \rho E\bar{v} + p\bar{v} \end{bmatrix} \tag{5}$$

and

$$\bar{F}^v = \begin{bmatrix} . \\ \bar{\bar{\tau}} \\ \bar{\bar{\tau}} \cdot \bar{v} + \kappa\nabla T \end{bmatrix} \tag{6}$$

respectively, where p is the static pressure, $\bar{\bar{\tau}}$ is the viscous stress tensor, T is the temperature, κ is the thermal conductivity, and μ is the viscosity. The viscous stress tensor can be expressed in vector notation as

$$\bar{\bar{\tau}} = \mu\left(\nabla\bar{v} + \nabla\bar{v}^T\right) - \mu\frac{2}{3}\bar{\bar{I}}\left(\nabla \cdot \bar{v}\right). \tag{7}$$

For a perfect gas with gas constant R and constant specific heat ratio γ, the ideal gas equation of state can be used to determine the pressure and temperature as $p = (\gamma - 1)\rho\,[E - 0.5(\bar{v} \cdot \bar{v})]$ and $T = p/(\rho R)$, respectively. Laminar viscosity, μ, is calculated using Sutherland's law and thermal conductivity can be computed as $\kappa = \mu c_p/Pr$, where c_p is the specific heat capacity at constant pressure and Pr is the Prandtl number, or set to a constant value. Steady, turbulent flows can be simulated in `SU2` by solving the Reynolds-averaged Navier-Stokes (RANS) equations. In accordance with the Boussinesq hypothesis, the effective viscosity can be computed as the sum of the laminar (dynamic) viscosity and turbulent viscosity as

$$\mu = \mu_d + \mu_t, \tag{8}$$

where the prediction of μ_t is dependent on the particular choice of turbulence model used. In this work, the one-equation Spalart-Allmaras turbulence model with Rotation/Curvature correction is used to solve for the value of the turbulent viscosity [35, 36]. The turbulence model is selected to achieve a balance between the computational cost and adequate vortex effects quantification accuracy, as discussed in a previous study by Seraj and Martins [37]. All simulations employ the Jameson-Schmidt-Turkel scheme for convective discretization [38] and implicit time integration with an adaptive Courant–Friedrichs–Lewy number to converge the steady-state solution. The vehicle boundary condition is specified by a no-slip, adiabatic wall, and a characteristic-based free-stream is used for the far-field boundary. A Cauchy convergence criteria is used, such that the solver stops when the change in C_D for the wing over the previous 100 iterations becomes less than 1×10^{-4}.

C. Dataset

1. Dataset Overview

The dataset for surrogate model training and testing consists of 15 3-D Delta wing geometries at two levels of fidelity. The HF surface pressure field data is generated using the `SU2` CFD solver to simulate the steady-state aerodynamics with a RANS solver as described in Section II.B. The flow domain is discretized using `Pointwise`'s automatic unstructured volume mesh capability [39]. The outer mold line of each underlying geometry is accurately captured by enforcing a maximum surface mesh-to-geometry deviation of less than 1×10^{-5} [m]. Across all geometries, identical mesh property definitions are used to ensure consistency in global mesh properties and boundary conditions, including the T-Rex boundary layer mesh growth profile, leveraging `Pointwise`'s scripting capability. Due to variations in surface body geometry, the total number of elements in the fluid domain ranges from 4.89 to 19.47 million. The resultant mesh achieves $y^+ < 1$ for most parts of each wing, except near the wing tips at the trailing edge, satisfying the turbulence model's y^+ requirements.

The LF data is generated using the VLM implementation in `SUAVE`. For all geometries, a 30×30 lattice discretization is applied in the chord-wise and span-wise directions. The lattice is evenly distributed in the chord-wise direction and follows a cosine distribution in the span-wise direction.

The 15 Delta wing geometries are parameterized using two design variables: leading edge sweep angle and root airfoil shape. The leading edge sweep angles used in this study are 55°, 65°, and 75°. Across all geometries, the root chord length is kept constant at 0.65 [m], resulting in variations in span and reference area of the Delta wing. The root airfoil shapes are taken from NACA 4-digit airfoils, and in this study, NACA0010, NACA0016, NACA0024, NACA2416, and NACA4416 are used. All 15 combinations of the root airfoil shape and leading edge sweep angle are generated using `SUAVE`.

Each geometry is then simulated under 40 different free-stream conditions, varying AOA, Mach number (Ma), and Reynolds number (Re). The AOA is sampled from 0° to 20°, the Ma from 0.35 to 0.5, and the Re from 6.5×10^6 to 1×10^7. A Latin Hypercube Sampling (LHS) technique is used to generate samples across the three free-stream variables. It is worth noting that the LF VLM simulation does not utilize the Reynolds number in its simulation. In total, 600 test samples are generated across the geometry design space and the free-stream variable space.

The dataset is generated on the Sherlock Cluster at the Stanford Research Computing Center (SRCC), utilizing a 64-core 2.4 GHz partition.

2. Dataset Preparation

The HF and LF datasets differ significantly in their dimensions. While the LF data resides in $\mathbb{R}^{30\times30}$, the HF data dimension is defined by the number of surface mesh nodes, which varies based on the geometry. Although GNN-based

architectures can accommodate mapping between these different graph structures, training such surrogate models can be computationally expensive and is not the focus of this work.

In this study, the goal is to demonstrate the capability of utilizing such architectures to build surrogate models that enhance conceptual vehicle design at a relatively low computational cost. To achieve this, the authors decided to standardize the HF and LF graphs. One approach is to project the HF surface pressure field onto the lattice panels defined in the VLM. As VLM computes the local loading coefficient $\Delta C_p^{LF} = C_{p,\text{lower}} - C_{p,\text{upper}}$, it was decided that the standardization step for the HF data should project the surface pressure onto the lattice panels to generate the HF local loading coefficient, ΔC_p^{HF}. This process is described by the following equations:

$$\Delta C_p^{HF} = C_{p,\text{lower}}^{HF} - C_{p,\text{upper}}^{HF}, \tag{9}$$

where C_p^{HF} is a vector in $\mathbb{R}^{30x30}$. The pressure coefficient for each side can be calculated by:

$$C_{p,\text{side}}^{HF} = \frac{1}{q_{\text{ref}}} \left[\frac{\sum_{i \in \mathcal{I}} \delta_{\text{side}}(i) P_i A_i}{\sum_{i \in \mathcal{I}} \delta_{\text{side}}(i) A_i} \right]_{\mathcal{I}=1}^{N}, \tag{10}$$

where P_i is the pressure, A_i is the cell area at HF node i, $\mathcal{I}$ is the set of all surface nodes whose (x_i, y_i) coordinates lie within the boundary of a lattice panel, indexed from 1 to N, under vertical projection, and q_{ref} is the reference pressure computed from free-stream conditions. $\delta_{\text{side}}(i)$ is an upper or lower surface selection mask that identifies whether a node belongs to the lower or upper surface based on its z_i coordinate (applicable for the root airfoils used in the current study). For example, the $\delta_{\text{lower}}(i)$ mask can be defined as:

$$\delta_{\text{lower}}(i) = \begin{cases} 1 & \text{if } z_i < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

3. Node and Edge Features

As discussed in Section II.C.1, the VLM simulation is Reynolds number agnostic. To enrich the aerodynamic features used in the surrogate model, free-stream conditions, including free-stream AOA, Ma, and Re are incorporated alongside the local loading coefficient ΔC_p^{LF}. Because the proposed surrogate model is intended to augment aerodynamic prediction for design space exploration in conceptual design, it is essential to include geometric properties that influence aerodynamics in the surrogate model input. These geometric properties include wing thickness, surface curvatures, lattice panel control point coordinates, and surface slopes. All geometric features are defined and evaluated at the nodal level.

At the edge level, the Euclidean distance between nodes is used to represent nodal connectivity patterns and spatial relationships among nodes. For each node, the Euclidean distances to its four direct neighboring nodes are included in the edge feature. The combination of nodal and edge features allows the `VortexNet` surrogate model to capture the spatial and aerodynamical information for aerodynamic characterization.

The nodal and edge features used by the `VortexNet` surrogate model, along with their intended purposes, are summarized in Table 2.

Once these node and edge features are assembled, a Hyperbolic Tangent (Tanh) standardization [40] is performed for each feature to scale and normalize all feature values to a range between −1 and 1. After these dataset preparation steps, the LF graph and HF training label can be feed into the surrogate for training.

D. Surrogate Modeling

A MF surrogate model is used to establish a mapping function, $\mathcal{F}$, between the LF field variables q^{LF} into their HF counter part q^{HF}. In general, such a mapping can be expressed as

$$q^{HF} \sim \mathcal{F}\left(q^{LF}, \omega_{\mathcal{L}}\right), \tag{12}$$

where $\omega_{\mathcal{L}}$ represent the weights and biases of the network under a specific loss function $\mathcal{L}$. Such a mapping exists if the LF field variables q^{LF} exhibit a one-to-one correspondence with the HF field variables q^{HF}, a hypothesis that has not been tested in this study. To construct such a surrogate model, a GNN based U-net like ML autoencoder is used. The network architecture and the training pipeline are presented in detail in the following sections.

Feature Name	Dimension	Purpose
ΔC_p^{LF}	$[N_{\text{nodes}} \times 1]$	VLM local loading coefficient
AOA $[rad]$	$[N_{\text{nodes}} \times 1]$	free-stream AOA
Ma_∞	$[N_{\text{nodes}} \times 1]$	free-stream Mach number
Re_∞	$[N_{\text{nodes}} \times 1]$	free-stream Reynolds number
Thickness $[m]$	$[N_{\text{nodes}} \times 1]$	wing thickness at lattice panel control points
Upper Surface Curvature $[m^{-1}]$	$[N_{\text{nodes}} \times 1]$	wing upper surface curvature at lattice panel control points
Lower Surface Curvature $[m^{-1}]$	$[N_{\text{nodes}} \times 1]$	wing lower surface curvature at lattice panel control points
Upper surface slope	$[N_{\text{nodes}} \times 1]$	wing upper surface chord-wise slope at lattice panel control points
Lower surface slope	$[N_{\text{nodes}} \times 1]$	wing lower surface chord-wise slope at lattice panel control points
x_c $[m]$	$[N_{\text{nodes}} \times 1]$	lattice panel control points x-coordinates
y_c $[m]$	$[N_{\text{nodes}} \times 1]$	lattice panel control points y-coordinates
Euclidean distance $[m]$	$[4N_{\text{nodes}} \times 1]$	Euclidean distances from one lattice panel control points to its four direct neighbors'

Table 2 List of the nodal and edge features used by the `VortexNet`, along with their dimension and intended purposes.

1. Machine Learning Architecture

A graph consists of a set of nodes and edges $G = (V, E)$, where $V = \{1, ..., n\}$ is a set of nodes and E is a set of edges between nodes $i, j \in V$. A GNN is thus a model that performs machine learning on graphs to incorporate information about the structure of the graph to understand the relationship among nodes and edges [41]. This capability is classified into three prediction tasks: node, edge, and graph predictions. The node prediction capability is of particular interest to the authors, as this type of prediction uses graph information to generate predictions on nodal values through message passing rounds. In each message passing round, each node updates its nodal embedding using information aggregated from its neighbors' embeddings. This type of prediction integrates well with mesh-based simulations, such as VLM or RANS solvers used in this study, as the simulation results' data structure can be easily represented by a graph G [42, 43].

Recently, multiple studies have leveraged GNN architectures to predict physical fields, including across varying fidelities. Pfaff et al. proposed `MeshGraphNets`, which use message-passing mechanisms to learn mesh-based simulation with noticeable efficiency and accuracy for both static and dynamic systems [25]. Their paper also highlighted the capability of `MeshGraphNets` to be discretization independent by training on highly irregular grids. This feature is desirable for the surrogate modeling task in this paper, as the surrogate model must not be sensitive to spatial discretization changes, such as mesh morphing and mesh topology changes, caused by geometry variation in the MDO cycle. Shao et al. proposed an auto-encoder GNN surrogate called `PIGNN-CFD` for rapid prediction of urban wind fields based on irregular unstructured mesh data from CFD simulations [26]. In their work, they added the Reynolds stress tensor residual to the loss function, thereby implicitly building physical constraints into the surrogate model. When applied to multi-scale and multi-fidelity applications, U-net [29] types of architectures are commonly used. Deshpande et al., Yang et al., and Wei and Freris demonstrated the effectiveness of U-net type GNNs in physical simulations, particularly in the domains of finite element method (FEM)-based structural simulations and smooth particle hydrodynamics (SPH)-based fluid simulations [27, 30, 31]. Their work highlighted the superior performance of graph U-Nets in multi-scale representation learning, accommodating unstructured mesh data, modeling long-range interactions, and being mesh topology-agnostic. Finally, Liu et al. used a graph attention network [44]-based surrogate for fluid simulations and found that the attention mechanism helps describing physical relationships among graph nodes. This mechanism enables the network to learn from different representation subspaces, significantly enhancing its aggregation capability compared to traditional GNNs and allowing it to capture more complex fluid patterns [28].

The `VortexNet` network proposed in this paper has an architecture inspired by the aforementioned works. Specifically, it utilizes a multi-head message aggregation and passing layer using Graph Attention Network v2 (`GATv2`) [45] and incorporates U-net type skip connections between encoders and decoders [29]. The use of multi-head attention enables

the network to attend to richer feature subspaces by concurrently building multiple regression models between certain node to its neighbors, thereby enhancing its feature-capturing capability. Additionally, as the network may require multiple layers of message aggregation and passing, U-net type skip connections are employed to short-circuit deep encoders, improving the network's efficiency in capturing low-frequency features. Overall, the network is designed to handle graph sizes typically used in VLM simulations, which contain hundreds to thousands of nodes. It aims to capture nonlinear flow patterns in RANS simulations and augment VLM results, providing a tool to increase design fidelity during the conceptual design phase.

Figure 1 provides a high-level schematic illustration of the proposed `VortexNet` network. In this figure, the computational graph before and after each major block is shown in black boxes. The graph consists of 4 nodes, arranged in a Delta wing layout, and 5 black solid lines connecting them represent edges in the graph. The authors want to emphasize that the edges and nodes shown here are just for illustration purposes and do not represent the actual graph topology of the input data. The purple vectors (represented by purple boxes) beside each node are the corresponding nodal feature arrays, and the yellow vectors (represented by yellow boxes) beside each edge are the edge feature arrays. As the graph passes through the network, the nodal feature array and edge array may change sizes, which is indicated by the lengthening or shortening of the boxes representing the array. Hence, each individual graph in the boxes represents a snapshot of the graph's status. In this chart, a black arrow indicates a direct connection among stages, and a blue arrow represents a U-net type skip connection that will be explained in the following sections.

The input graph first goes through an Input Encoder, which maps the input features to a latent dimension. The encoded graph then goes through a series of Convolutional Blocks. These blocks use multi-head attention mechanisms, and hence the feature array size may be multiplied by the number of heads. In this network, the first half of the Convolutional Blocks serves as the encoder, and the second half of the Convolutional Blocks serves as the decoder. The decoder blocks receive information from both their previous blocks and the graph information passed on by skip connections. Finally, an Output Decoder, which utilizes an architecture similar to the Convolutional Block but with optional fully connected layers to adjust the output dimension, is used to produce nodal predictions for a field value of interest, such as pressure or velocity fields. The following sections describe the key components of `VortexNet`.

Input Encoders

The Input Encoder comprises two components: a node encoder and an edge encoder. The objective of including this input encoder is to enrich the input nodal features and edge features. Such enrichment is needed because the nodal feature array usually has limited dimension. For example, in the current work, an 11-dimensional nodal feature array is used. Similarly, the edge feature array only contains the distance between connected nodes. Hence, a feature encoding to expand the input feature array to higher dimensional space is beneficial for the latter stage of the Convolutional Blocks, where latent relationship will be identified. A linear transformation is used for both the node encoder and the edge encoder, such that

$$\mathbf{h}_i^{'} = \mathbf{W}_n \mathbf{h}_i + \mathbf{b}_n, \tag{13}$$

where $\mathbf{h}_i^{'}$ is the encoded feature, $\mathbf{h}_i$ is the input feature, $\mathbf{W}_n$ are the learnable weights and $\mathbf{b}_n$ are the learnable bias. The node encoder encodes nodal feature to a dimension of [`hidden_feature` $\times$ `hidden_feature`], with `hidden_feature` being a hyper-parameter used by `VortexNet`, and the edge encoder encodes edge feature to a dimension of `hidden_feature`. After the linear layer, a dropout layer is used for the node feature array.

Convolutional Block

In each Convolutional Block, the primary mechanism for updating node features based on graph structure is by using `GATv2Conv` layers [45]. Specifically, the `GATv2Conv` computes a weighted average of the transformed featured of the neighborhood nodes for each nodes, expressed as,

$$\mathbf{h}_i^{'} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \mathbf{h}_j\right), \tag{14}$$

where $\mathbf{h}_i^{'}$ is the new feature representation at node i, $\mathcal{N}_i$ is the set of neighboring nodes connected to node i by direct edges, α_{ij} and $\mathbf{W}$ are the learned weights, $\mathbf{h}_j$ is the input feature array at node j, and σ is an activation function. The attention weights α_{ij} indicate the importance of node j's features to node i and allow every node to attend to every other node connected by the graph, thereby effectively disregarding the structural information of the graph [44]. This property can help the convolution block to behave in a mesh-discretization-independent manner when the node and edge features are designed properly. The actual attention function is defined as:

$$\alpha_{ij} = \text{softmax}_j\left(e(\mathbf{h}_i, \mathbf{h}_j)\right) = \text{softmax}_j\left(\mathbf{a}^T \text{LeakyReLU}(W \cdot [\mathbf{h}_i][\mathbf{h}_j])\right), \tag{15}$$

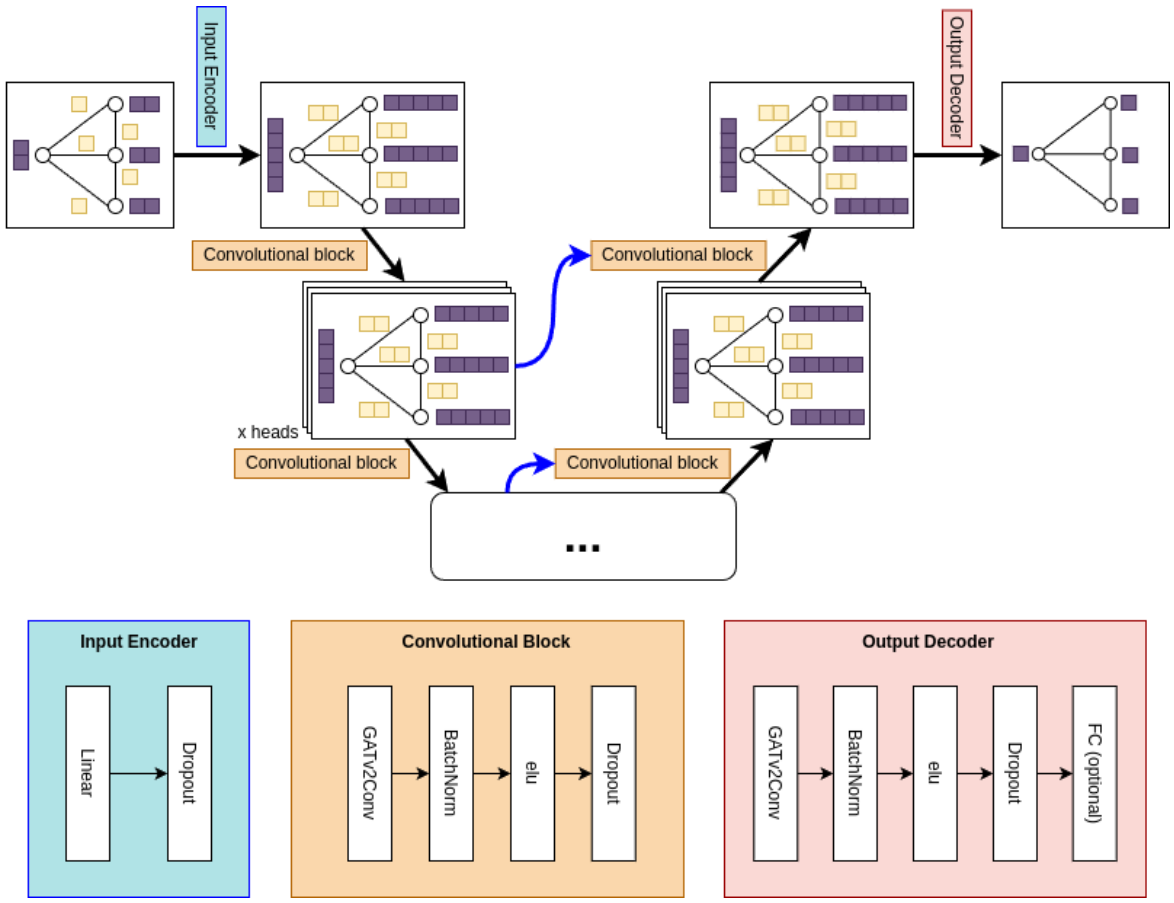

Fig. 1 Schematic of the proposed `VortexNet` graph neural network (GNN) architecture. Colors indicate different network blocks. A black arrow indicates represents direct message passing between blocks, and a blue arrow denotes a skip connection to the receiving block. The figure also presents snapshots of the graph at each computational step, showing nodes, edges, and their associated feature arrays.

where $e(\mathbf{h}_i, \mathbf{h}_j)$ is the feature importance score of neighbor j to node i, $\mathbf{a}$ is a learned weights.

The `PyTorch` `GATv2Conv` layer employs multiple attention heads that allow the model to attend to information from different representation subspaces. In the multi-head attention scenario, multiple independent attention mechanisms execute the transformation of Equation 14, and then the resulting feature array $\mathbf{h}_i^{'}$ is concatenated.

Utilizing this `GATv2Conv` layer, the Convolution Block consists of one `GATv2Conv` layer that maps the input graph with a nodal feature array in $\mathbb{R}^{\mathbb{F}}$, where $\mathbb{F}$ is defined by the hyper-parameter `hidden_feature`, to a nodal feature array in $\mathbb{R}^{\mathbb{F}\times\texttt{heads}}$ after concatenation. Additionally, `heads` is a hyper-parameter specifying the number of heads. The input-output dimension is consistent across all layers except for the first Convolution Block after the Input Encoder, where the input feature dimension is $\mathbb{R}^{\mathbb{F}\times\mathbb{F}}$. After the `GATv2Conv` layer, the feature passes through batch normalization, an exponential linear unit (elu) activation function, and a dropout regularization layer. The dropout layer includes a tunable hyper-parameter named `dropout_rate`.

In the `VortexNet` architecture, the Convolutional Block is repeated (`hops` − 1) times, where `[hops]` is a tunable hyper-parameter. It is worth noting that in many previous graph U-net-like architectures, pooling layers are used to successively reduce the computational graph size in deeper encoder layers [27, 30, 31]. However, such pooling layers are not used in the current network because the implications of using these methods remain uncertain to the authors during this work. Moreover, the removal of pooling layers reduces the size of the network architecture's hyper-parameter search space. A discussion on the implication of including pooling layers to the `VortexNet` architecture is presented in the Results section III.

Output Decoder

The Output Decoder's major role is to map latent space features to the desired output dimension. This block contains layers identical to those used in a Convolution Block, except that the `GATv2Conv` layer now maps the feature array from $\mathbb{R}^{\mathbb{F}\times\texttt{heads}}$ to $\mathbb{R}^{1}$. An optional fully connected (FC) layer is appended at the end of this block to enable graph size adjustment for the final output. In the context of this work, since the input graph and training labels have identical graph sizes, this fully connected layer is not used.

Although edge features can be extracted after the Output Decoder block, they are discarded. Consequently, the Output Decoder provides scalar value predictions for each node, which represent the field value predictions that mimic the HF training labels.

Skip Connections

The skip connection is marked by the blue curvy arrow in Figure 1. For the i^{th} Convolutional Block, its output is connected to the $(\texttt{hops} - i - 1)^{th}$ block's input. Thus, the first encoder block's output is skip-connected to the Output Decoder, the second encoder block's output is connected to the last decoder block, and so on. As the latent space dimension remains consistent across all these encoder-decoder blocks, these skip connections do not require dimension adjustments.

A skip connection weight hyper-parameter, denoted as `Alpha`, is used to control the amount of skip connection information that the subsequent decoder utilizes, such that the feature array $\mathbf{h}_i$ seen by the decoder is

$$\mathbf{h}_i = \mathrm{A}\mathbf{h}_{i,\text{skip connection}} + (1-\mathrm{A})\mathbf{h}_{i,\text{previous block}}. \tag{16}$$

2. Training

The prediction accuracy of `VortexNet` relies heavily on the training scheme. Throughout experiments, it has been observed that a training scheme with a combination of learning rate scheduling, regularization techniques, and physical loss terms helps `VortexNet` to achieve higher prediction accuracy. The following sections discuss the training scheme used in this work.

K-fold Cross Validation

The entire dataset, as described in Section II.C.1, is randomly partitioned into a $30\% - 70\%$ test-training split. For network training, only the training set is exposed to the `VortexNet`, while the test set is reserved for performance characterization. The training set is further randomly partitioned into four batches. In each fold of the k-fold training, one batch of the training set serves as the cross-validation set, such that the model prediction loss on this set drives the optimization of the model weights, while the remaining subsets serve as the training data. This technique exposes the model to diverse data distributions and prevents overfitting to specific cross-validation subsets [46].

In standard k-fold cross-validation training, the models trained in each fold are independent. After completing training across all k folds, a voting algorithm is typically employed to select the best-performing model among the folds, or a weighting algorithm is used to combine the trained models across folds.

In the current study, we applied the k-fold cross-validation training but made minor adjustments to the model voting mechanism. Instead of training mutually independent models in each fold, our training script recycles the best model weights identified so far from all previous folds' training. Specifically, a register actively stores the best-performing model, as evaluated by the cross-validation loss, and in each fold training resumes from the current best model.

In using this training scheme, the cross-validation set in the later folds may overlap with the training data of previous folds, potentially inflating the model performance in later folds. However, this strategy was chosen for two reasons. First, the size of the total training set is limited compared to the number of trainable weights in the `VortexNet`. Using standard k-fold training prohibits making a full use of the training set. Second, the way physical loss terms are used in training necessitates such model recycling, as discussed in the next section.

Loss Function

The loss function used in this study for both training and cross-validation is a compound loss defined as:

$$\mathcal{L} = \mathrm{SmoothL1Loss}\left(q^{HF}, q^{\text{pred}}\right) + \Lambda\,\mathrm{PhysicalLoss}\left(q^{\text{pred}}\right) + \Omega\,\mathrm{SignLoss}\left(q^{HF}, q^{\text{pred}}\right), \tag{17}$$

where "SmoothL1Loss," "PhysicalLoss," and "SignLoss" are the three loss terms; q^{HF} is the high-fidelity data provided as training labels; $q^{\text{pred}} = \mathcal{F}\left(q^{\text{LF}}, w_{\mathcal{L}}\right)$ is the `VortexNet` predicted field output; Λ is the weight of the physical loss which is controlled by two hyper-parameters as described in Equation 21; and Ω is a hyper-parameter controlling the weight of the sign loss (`penalty_weight`). The "SmoothL1Loss" is well-documented in the `PyTorch` manual and will not be repeated here [47].

The "SignLoss" term penalizes sign mismatches between the predicted fields and the training labels. Specifically, it is defined as:

$$\text{SignLoss}(q, \hat{q}) = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{1} \left(\text{sign}(q_i) \neq \text{sign}(\hat{q}_i) \right) \right), \tag{18}$$

where q and $\hat{q}$ are two arbitrary vectors with the same dimensions, N is the length of array q, and $\mathbf{1}$ is an indicator function that returns 1 if the statement is true and 0 otherwise. This loss term is introduced to address a limitation of the "SmoothL1Loss," which cannot differentiate between positively and negatively biased predictions. By incorporating the "SignLoss," the total loss equation can be adjusted to better handle sign errors, with its contribution controlled by the `[penalty_weight]` hyper-parameter.

Finally, the "PhysicalLoss" term is introduced to further improve the model's prediction capability. In designing surrogate models, two typical approaches are commonly employed: physics-driven and data-driven. In the physics-driven approach, the governing equations can either be embedded in the aggregation function or incorporated into the loss term. For surrogate models aimed at HF flow field prediction, the residuals of the Navier-Stokes equations at selected points are often used to drive the network convergence [48]. While accurate, such networks are computationally expensive to train. Alternatively, in the data-driven approach, the physical loss is omitted, making the network comparatively cheaper to train. However, this approach often suffers from high variance in prediction errors.

Considering that the proposed surrogate model is intended for conceptual design purposes, where trend-wise accuracy and delta values of QOIs are more critical than the absolute prediction accuracy, we aim to employ a lightweight physical constraint. This constraint helps calibrate the surrogate model predictions for trend-wise quality without adding significant complexity to the network architecture. In this study, we propose a physical loss term based on LF VLM simulation to improve the network's trend-wise prediction quality. For non-linear flow features, such as flow separation, vortex breakdown, and vortex attachment phenomena in subsonic wings, the data-driven approach is used.

To compute the physical loss, recall that VLM solves the following linear equation:

$$AIC \cdot \Gamma = RHS, \tag{19}$$

where AIC is the aerodynamic influence coefficient matrix, Γ is the vortex strength, and RHS is the Neumann boundary condition for normal flow. The resolved vortex strength can then be used to calculate the total pressure coefficient difference, ΔC_p. When a surrogate model is used to predict $q^{\text{pred}} = \Delta C_p^{\text{pred}}$, this field can be passed backward through the stored lattice geometry information to compute the predicted vortex strength distribution, Γ^{pred}. The physical loss term can thus be expressed as a 2-norm of the residual:

$$\text{PhysicalLoss}(q^{\text{pred}}) = ||AIC \cdot \Gamma^{\text{pred}}(q^{\text{pred}}) - RHS||. \tag{20}$$

In the above physical loss computation, no real-time computation of the AIC and RHS matrices is required, as this information can be stored during the dataset assembly phase, thereby creating minimal overhead for network training.

When the physical loss weight Λ is large, the surrogate-predicted ΔC_p^{pred} will closely resemble the VLM results. However, this undermines the intent of using the surrogate model to map LF field data to HF field data. To address this, an exponential decay of the physical loss weight Λ is applied during the k-fold training. Initially, the model is trained to closely match the VLM solver. While the model kept the learned mapping, the physical constraint is gradually relaxed. Towards the final folds, the surrogate model's training convergence is primarily driven by data to capture the non-linear flow patterns present in HF simulations. The following equation is used to decay the physical loss weight across folds:

$$\Lambda_i = \Lambda^{\max} \exp\left(\frac{\Lambda^{\min}/\Lambda^{\max}}{k-1} (i-1) \right), \text{for } i \in \{1, ..., k\}, \tag{21}$$

where $\Lambda^{\max}$ is a hyper-parameter (`max_phys_loss`) representing the initial physical loss weight, $\Lambda^{\min}$ is a hyper-parameter (`min_phys_loss`) representing the final physical loss weight, and k is the number of folds in k-fold training.

Optimizer and Training Scheme

Optimization is carried out using the ADAM optimizer with an initial learning rate controlled by the hyper-parameter `learning_rate` and a decay rate specified by `decay_rate` [49]. A dynamic learning rate adjuster is implemented to halve the learning rate when the validation loss stops improving across training epochs. An early stopping scheduler is set

such that if the model performance does not improve over the last 500 epochs, training for the current fold is terminated, and the best model trained so far is returned to reduce the risk of overfitting. For all nodal features, 2% random noise is added to enhance the model's robustness. Throughout all training and model configurations, as determined by hyper-parameters, a batch size of 32 is used. Training is conducted on a NVIDIA A100 graphics processing unit (GPU).

III. Results

The results are organized as follows: First, the hyper-parameter optimization results are presented in Section III.A, including insights into optimal network parameters identified during the study, as well as results on hyper-parameter sensitivities. Second, `VortexNet`'s prediction accuracy is evaluated using two tests. The first test reports the local loading coefficients (ΔC_p) prediction accuracy for test set, while the second test evaluates the surrogate model's generalizability on unseen geometries. These results are presented in Section III.B. Third, we present a latent space ablation study to understand how the graph features are utilized across latent layers in Section III.C.

A. Network Training and Hyper-Parameter Optimization

The surrogate model (`VortexNet`) described in Surrogate Modeling is trained using the dataset detailed in Dataset. Since the proposed surrogate model involves several training hyper-parameters, a hyper-parameter sensitivity study is conducted to provide inductive insights into the relationship between model performance and these hyper-parameters, as well as to offer guidance for optimizing surrogate models similar to the `VortexNet` in future studies. These hyper-parameters are defined in Surrogate Modeling, but for the convenience of readers, Table 3 summarizes the hyper-parameters included in this study.

Hyper-parameter Name	Value Range	Definitions
`hops`	int$[3, 20]$	total layers of Convolutional Block
`hidden_feature`	int$[3, 64]$	hidden layers feature width
`heads`	int$[2, 8]$	number of heads for multi-heads attention
`dropout_rate`	$[0.1, 0.5]$	dropout regularization weight
`Alpha`	$[0.0, 1.0]$	skip connection bandwidth
`penalty_weight`	$[1, 10] \times 10^{-2}$	sign loss weight
`max_phys_loss`	$[0, 0.5]$	maximum physical loss weight in Equation 21
`min_phys_loss`	$[1, 1000] \times 10^{-5}$	minimum physical loss weight in Equation 21
`learning_rate`	$[1, 100] \times 10^{-2}$	initial learning rate to the ADAM optimizer
`decay_rate`	$[1, 1000] \times 10^{-5}$	decay rate to the ADAM optimizer

Table 3 List of the hyper-parameters studied, along with their value range and definitions.

The hyper-parameter optimization study is conducted using `Optuna`, an open-source hyper-parameter optimization framework that automates hyper-parameter search using advanced sampling schemes and efficient pruning algorithms [50]. The hyper-parameters, along with their bounds listed in Table 3, define the search space, and `Optuna` automatically identifies the optimal hyper-parameter settings that minimize the model test loss on the test set. The model test loss is defined as the mean-square error (MSE) between the predicted ΔC_p^{pred} and the corresponding HF ΔC_p^{HF} across the entire test set.

A total of 100 hyper-parameter configurations are generated in this study. The training time varies across configurations, but on average, each configuration takes 2.36 hours to train.

The average test loss across all hyper-parameter configurations is 9.55×10^{-2}, with a standard deviation of 4.85×10^{-2}. The average hyper-parameters across the 100 configurations are listed in Table 4 to provide a baseline for understanding the typical settings explored during the study. As the `Optuna` search spans a large hyper-parameter space, the test loss ranges from 3.99×10^{-2} to 0.246. This indicates that some combinations of hyper-parameters are sub-optimal for the prediction task.

Performing sensitivity analysis directly on the entire set of hyper-parameter samples is not ideal, as these sub-optimal parameter combinations would skew the sensitivity results. Instead, we use the K-means algorithm to cluster the

Hyper-parameter Name	Average Value
`hops`	6.95
`hidden_feature`	42.88
`heads`	5.72
`dropout_rate`	1.82×10^{-1}
`Alpha`	2.58×10^{-1}
`penalty_weight`	2.23×10^{-2}
`max_phys_loss`	2.89×10^{-1}
`min_phys_loss`	1.99×10^{-3}
`learning_rate`	6.35×10^{-2}
`decay_rate`	3.56×10^{-4}

Table 4 Average hyper-parameter values from all hyper-parameter configurations under the optimization study.

10-dimensional hyper-parameter space and perform the sensitivity study only on the subset of configurations that show consistently low test loss. Based on trial-and-error testing, the hyper-parameter configurations are clustered into five groups. To simplify the visualization of these clusters, principal component analysis (PCA) is employed to reduce the 10-dimensional parameter space to three dimensions. Figure 2(a) illustrates the clustering pattern with respect to the first two PCA axes. From these results, it is evident that cluster 3 outperforms the other clusters in terms of the test loss. This cluster is thus selected for the sensitivity study, and it is marked in green in Figure 2(b).

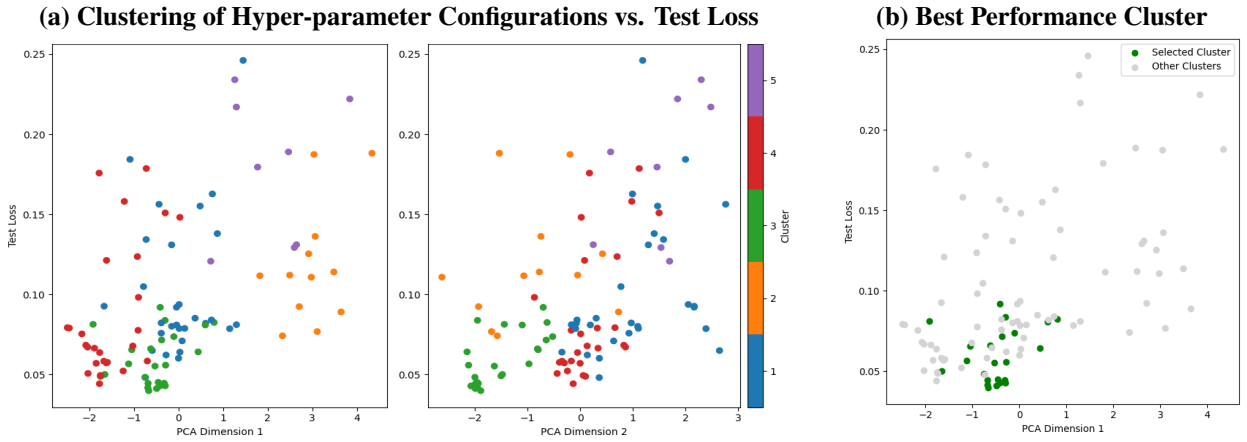

Fig. 2 Clustering results of hyper-parameter configurations based on test loss. (a) Hyper-parameter configurations K-means clusters visualized along the first two PCA axes. (b) The best-performing cluster (Cluster 3), identified based on consistently low test loss, is highlighted in green.

The selected best-performing cluster consists of 24 configurations, with an average test loss of 0.059. The average hyper-parameter values in this cluster are listed in Table 5. Comparing with the average values across the entire 100 test configurations in Table 4, the best-performing cluster uses a shallower network (less `hops`) with smaller latent feature size (less `hidden_feature`). The best-performing cluster also utilizes a more stringent enforcement on physical loss during the early epochs of training (higher `max_phys_loss`), but relaxed the physical loss constraint more towards the end of training (lower `min_phys_loss`).

A box chart summarizing the hyper-parameter configurations in this cluster is presented in Figure 3. As illustrated in the figure, most runs have a hidden feature size in the range of $[28, 46]$, use a 7-head attention mechanism, employ 4-hop convolutions, and have a skip connection bandwidth in the range of $[0.00761, 0.109]$. In this cluster, the spread of hyper-parameters such as `heads`, `hops`, `decay_rate`, `penalty_weight`, and `min_phys_loss` is relatively small, indicating that these hyper-parameters have converged toward optimal values in the best-performing cluster.

Hyper-parameter Name	Average Value
hops	4.83
hidden_feature	38.6
heads	6.86
dropout_rate	1.58×10^{-1}
Alpha	1.05×10^{-1}
penalty_weight	1.29×10^{-2}
max_phys_loss	4.25×10^{-1}
min_phys_loss	9.81×10^{-4}
learning_rate	1.73×10^{-2}
decay_rate	1.50×10^{-4}

Table 5 Average hyper-parameter values from the best-performing cluster.

The model with the best performance across all 100 runs is also part of this cluster. The model has a test loss MSE of 0.0399, with hyper-parameters listed in Table 6. Comparing with the best performance cluster's average hyper-parameters, this specific model utilized a similar network but with much smaller skip connection bandwidth (smaller `Alpha`). This indicates that the best performance model's Convolutional Blocks mainly accept information from their preceding Convolutional Blocks, instead of receiving shallower features similar to that of a U-net architecture. This also indicates that although the `VortexNet` architecture is setup to be an autoencoder that has both the encoder and the decoder components, the best performance model functions like an encoder only network. More details of the `VortexNet` explainability is provided in Prediction Explanations Section III.C.

Hyper-parameter Name	Average Value
hops	4
hidden_feature	33
heads	7
dropout_rate	1.36×10^{-1}
Alpha	8.02×10^{-3}
penalty_weight	1.31×10^{-2}
max_phys_loss	4.64×10^{-1}
min_phys_loss	8.07×10^{-4}
learning_rate	1.91×10^{-2}
decay_rate	1.11×10^{-4}

Table 6 `VortexNet` hyper-parameters that lead to the lowest test loss.

For sensitivity analysis, one way to quantify the significance of input variables on a function's output value, such as the test loss in this study, is through variance-based methods like first-order Sobol' indices [51]. Such analysis is particularly suitable for non-linear functions such as the surrogate model in the present study, as the response of surrogate test loss to input hyper-parameters is inherently non-linear. However, this type of analysis is prohibitively expensive for the current study. One approach to reduce the computational cost of computing Sobol' indices is to use a surrogate model. This method, however, is also not applicable here because the size of the sample in the best-performing cluster is too small to train a surrogate model with adequate variance-capturing capability, as evaluated by the R^2 score.

As a result, the authors rely on linear sensitivity analysis to gain insights into the model's sensitivity to hyper-parameters. In linear sensitivity analysis, a linear regression model is applied to each hyper-parameter array after

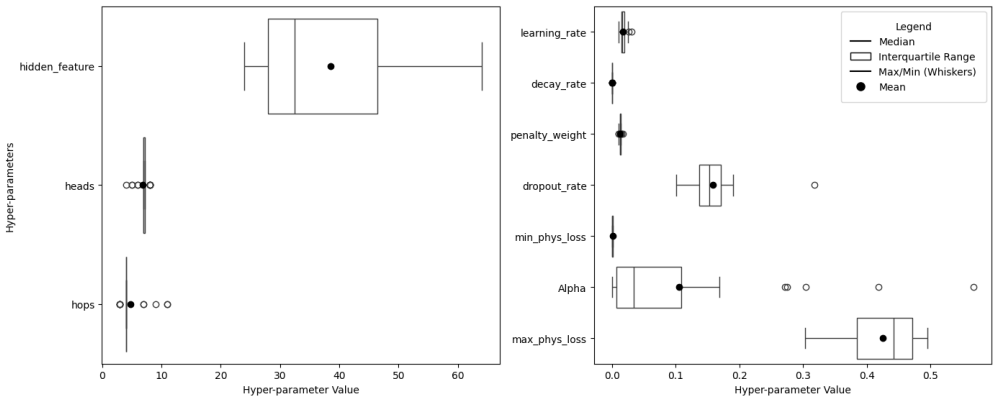

Fig. 3 Hyper-parameter statistics of the best-performing cluster.

standardizing it to a standard distribution. Consequently, the test loss can be modeled as a linear summation:

$$\mathcal{L}(X) = \sum_{i=0}^{N} \beta_i X_i, \tag{22}$$

where X_i represents a hyper-parameter array, $X_0 = 1$, N is the total number of hyper-parameters, and β_i is the corresponding linear coefficient [52]. The sensitivity of each hyper-parameter is then defined as:

$$\text{Sensitivity}(X_i) = \frac{|\beta_i|}{\sum_{j=0}^{N} |\beta_j|}. \tag{23}$$

Although this sensitivity definition does not capture the total variance, it provides an ad-hoc capability to explain the relative importance of one hyper-parameter compared to the others. The resultant sensitivities, along with the linear correlation coefficients, are presented in Figure 4.

(a) Hyper-parameters Sensitivity Index **(b) Hyper-parameters Linear Correlation**

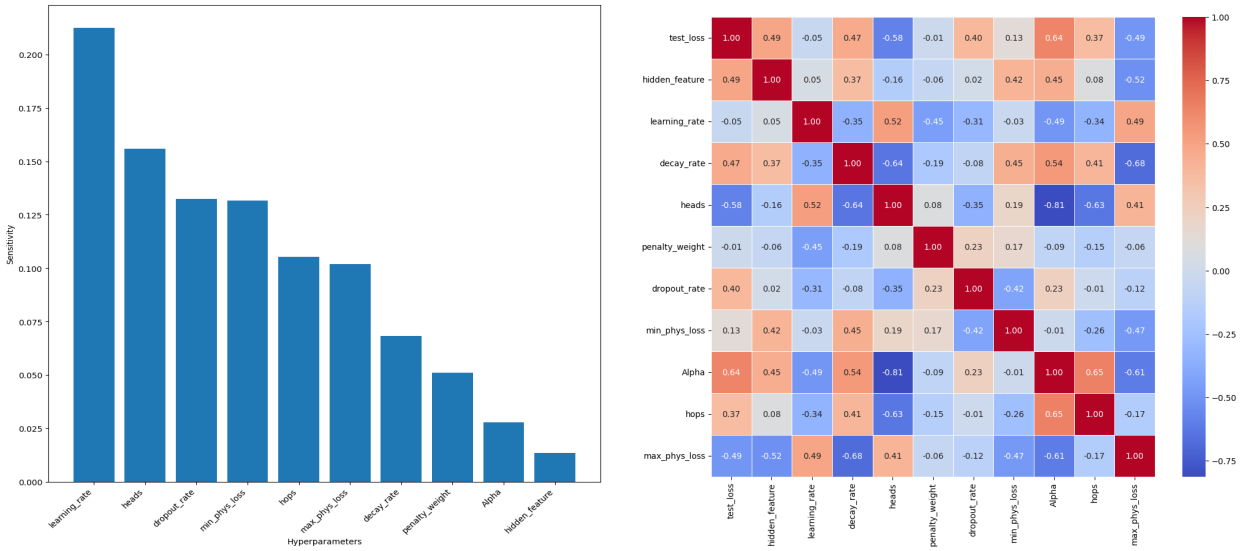

Fig. 4 Sensitivity index and correlation coefficients for hyper-parameters versus model performance in the best-performance cluster. (a) hyper-parameters sensitivity, and (b) hyper-parameters linear correlation.

For architecture hyper-parameters, it is identified that `heads`, `dropout_rate`, and `hops` have a significant impact on model prediction accuracy, while `hidden_feature` size and skip-connection weight `Alpha` have a minor impact.

Among these, increasing `heads` leads to worse model performance, likely due to limitations in dataset size and allocated training resources, causing the model to become prone to high-variance errors. Conversely, increasing `hops` and `dropout_rate` improves model's performance. Considering the high sensitivity of these hyper-parameters, future studies should consider extending their ranges to include higher values. For `hidden_feature` and `Alpha`, although linear correlation studies indicate a positive correlation, their effects remain unclear due to their relatively low sensitivity indices. A more extensive hyper-parameter search should be conducted for these parameters.

For training hyper-parameters, it is identified that `learning_rate`, `min_phys_loss`, and `max_phys_loss` have a significant impact on model performance, while `decay_rate` and `penalty_weight` do not. However, correlation studies reveal no linear relationship between `learning_rate` and test loss, likely because `learning_rate` is dependent on other training and model configuration parameters. Thus, fine-tuning of `learning_rate` should be conducted for future model training. It is also observed that a `max_phys_loss` value that is too large may negatively affect model performance. For `min_phys_loss`, in the best-performing model, the average physical loss on the validation set during the last folds of training is on the order of 40. A data loss-to-physical loss contribution ratio of $2.24:1$ is observed when the average physical loss is multiplied by its weight `min_phys_loss`. Finally, for `penalty_weight`, no significant impact is identified. This is likely because this loss component contributes minimally to the total loss, as the model becomes robust in predictions without violating sign constraints toward the end of training.

B. Aerodynamics Predictions

The main objective of this study is to develop a surrogate modeling technique that maps field data across different fidelities, thereby increasing fidelity in vehicle conceptual design without significantly raising costs. In this work, we focus on predicting local loading coefficients (ΔC_p). Aerodynamic coefficients can be obtained by integrating ΔC_p over panels, and we refer readers to Shen and Alonso for the aerodynamic coefficient results [53].

To evaluate the model's performance, we consider two distinct aspects. First, to gauge its prediction accuracy, we examine the root-mean-square error (RMSE) of the local loading coefficient (ΔC_p) field predictions ΔC_p^{pred} against the projected CFD references ΔC_p^{HF}, as outlined in Dataset Preparation. Second, to assess the model's generalizability and its potential for assisting in design space exploration, we evaluate its ΔC_p^{pred} RMSE against ΔC_p^{HF} using new geometries that were not exposed to the surrogate model during training. The following sections discuss these two assessments.

1. Local Loading Coefficients (ΔC_p) Prediction

The RMSEs for the training set, test set, and the entire dataset are 0.194 with an R^2 value of 0.838, 0.192 with an R^2 value of 0.841, and 0.193 with an R^2 value of 0.839, respectively. No significant difference between the test and training sets is observed and most variance in the dataset is captured by the trained model, indicating that the resultant model is well-trained.

To visualize the ΔC_p from the VLM, `VortexNet` predictions, and CFD references, a geometry configuration at high AOA was randomly selected. The selected Delta wing configuration features a leading edge sweep angle of 55° and a root airfoil of NACA2416. The free-stream conditions are AOA at 16.4°, Ma at 0.47, and a Re of 9.78×10^6. The corresponding CFD pressure coefficients (C_p) on the upper and lower wing surfaces are presented in Figure 5. The surface C_p exhibiting a "vortex lift" pattern: on the upper surface toward the wing outboard near the trailing edge, suction effects induced by the separated vortex are evident.

The aforementioned C_p is then projected onto lattice panels. Figure 6(c) shows the projected ΔC_p^{HF}, which serves as the "ground-truth" reference. The projected pressure field preserves flow features from the CFD, including faint traces of vortex lift at the wing outboards. In Figure 6(a), the ΔC_p^{LF} computed from VLM is shown. This pressure field follows potential flow theory, and no vortex lift or flow separations are observed. The VLM pressure field serves as input to `VortexNet` as part of the nodal features.

Figure 6(b) shows the ΔC_p^{pred} predicted by `VortexNet`. Comparing these pressure fields, it is evident that the predicted pressure field exhibits strong visual similarity to the projected CFD pressure field, both in the predicted ΔC_p values and the overall pressure distribution. However, artifacts of spiky, high-frequency reconstruction error are noticeable near the wing apex and wing tips. These errors can arise from the network's inability to fully resolve the fine-scale features, a common challenge in surrogate modeling. Additionally, in the mid-chord central body, the ΔC_p^{pred} prediction does not fully align with the reference field ΔC_p^{HF}. Despite these discrepancies, the authors believe that the surrogate model's predictions provide higher fidelity than those of VLM, as it captures nonlinear flow features.

Although not shown, ΔC_p^{pred} predictions for other geometries and free-stream conditions follow a similar pattern,

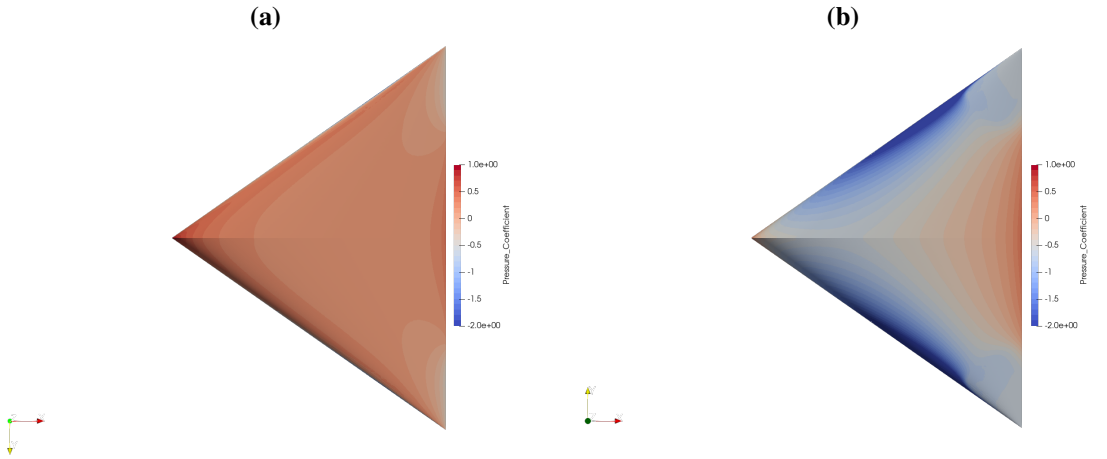

Fig. 5 The HF surface pressure coefficient obtained from CFD for a wing with 55° leading edge sweep and root airfoil profile of NACA2416 at AOA= 16.4°, $M_a = 0.47$, and $R_e = 9.78 \times 10^6$.

with `VortexNet`'s predictions appearing visually closer to the projected CFD fields (ΔC_p^{HF}) than to those of VLM (ΔC_p^{LF}).

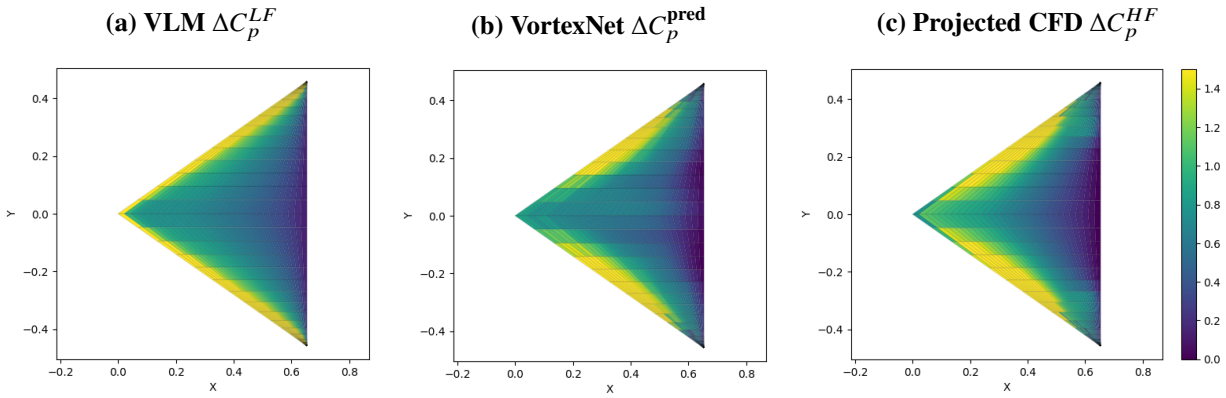

Fig. 6 Lattice panels and local loading coefficients ΔC_p are visualized using color maps ranging from $\Delta C_p = 0$ to $\Delta C_p = 1.5$, with (a) VLM ΔC_p^{LF}, also serving as the LF input, (b) `VortexNet` ΔC_p^{pred}, and (c) the reference HF ΔC_p^{HF} obtained by projecting the CFD surface pressure onto the lattice panels.

To provide a more quantitative analysis of the ΔC_p prediction error, we compared the RMSE across all geometries in the test set. The corresponding values are presented in Table 7. The RMSE ranges from 0.117 to 0.275 in the observed dataset. On average, the RMSE is 0.189 with a standard deviation of 0.0442 across all geometries. Although geometries with a leading-edge sweep angle of 75° exhibit the largest average error, an analysis of variance (ANOVA) test reveals no significant statistical bias between sweep angle and the error, or between root airfoil shape and the error. However, it can be concluded that `VortexNet`'s prediction accuracy depends on the input geometries. For some geometries, the prediction quality is better than the others.

2. Aerodynamic Prediction for Unseen Geometries

One potential use case of the proposed `VortexNet` is to assist in design space exploration during conceptual design. In such applications, a design space is first defined by specifying a set of geometric parameters and their respective bounds. A coarse sampling of geometries across this design space is then generated for both HF and LF data querying, which is used to train the surrogate model. Subsequent design space exploration can rely on the surrogate model to provide quasi-high-fidelity aerodynamic evaluations. Consequently, it is essential to evaluate the proposed surrogate

Sweep Angle [degrees]	Root Airfoil (NACA 4-digits)				
	0010	**0016**	**0024**	**2416**	**4416**
55	0.140	0.200	0.164	0.156	0.186
65	0.164	0.253	0.117	0.170	0.201
75	0.223	0.168	0.176	0.275	0.248

Table 7 Test set RMSE between ΔC_p^{pred} and ΔC_p^{HF}, grouped by wing geometries.

model's generalizability to geometries not included in the training dataset.

To study generalizability, we tested geometries and feature arrays not encountered by the network during training and evaluation. Four design points within the geometric parameter ranges of the training set's design space were selected. Specifically, we chose two leading-edge sweep angles (60° and 70°) and two root airfoils (NACA 0013 and NACA 3416). The 60° and 70° sweep angles can be viewed as intermediate values between the 55°, 65°, and 75° sweeps in the training set. The NACA 0013 airfoil is intermediate in thickness between NACA 0010 and NACA 0016, inducing a variation in wing thickness, while the NACA 3416 airfoil lies between NACA 2416 and NACA 4416, inducing a camber variation. A combination of these design variables yields four new wing geometries: 60° NACA 0013, 60° NACA 3416, 70° NACA 0013, and 70° NACA 3416. Each geometry is then simulated under 11 different far-field conditions, with AOA, Mach number, and Reynolds number sampled using LHS. The sampling bounds for AOA, Ma, and Re are $[0°, 20°]$, $[0.35, 0.5]$, and $[6.5, 10] \times 10^6$, respectively, matching those used to construct the training set as discussed in Dataset.

The resulting RMSE are listed in Table 8. The maximum RMSE observed is 0.389 for the 70° Delta wing with the NACA 3416 airfoil, and the minimum RMSE is 0.185 for the 60° Delta wing with the NACA 0013 airfoil. On average, the RMSE is 0.288, representing a 52.3% (or 2.2 standard deviation) increase from the average RMSE reported in Table 7. Under this limited new-geometry dataset, by examining the RMSE correlation to the nature of geometry interpolation, we speculate that a camber variation in underlying geometry leads to greater prediction errors for `VortexNet` than those that originate from a thickness variation.

One way to interpret the individual RMSE values presented in Table 8 is to compare the error of a specific geometry against its neighboring design points from Table 7. For instance, the wing with a 60° leading-edge sweep and a NACA 0013 airfoil has four neighbors in Table 7: 55° NACA 0010, 65° NACA 0010, 55° NACA 0016, and 65° NACA 0016. The average RMSE for these four neighbors is 0.189 with a standard deviation of 0.0491. A comparison of the RMSE of 60° NACA 0013 with its neighbors reveals no significant RMSE variation induced by interpolating among these geometries as the RMSE for new geometry (0.185) is within plus-minus two standard deviations to its neighbors' average RMSE (0.189 ± 0.0983).

Applying a similar argument to the 70° NACA 0013, 60° NACA 3416, and 70° NACA 3416 wings, we find that the RMSE for the 70° NACA 0013 configuration does not significantly differ from reference values in the test set reported in Table 7. However, for the two wings with the NACA 3416 airfoil, both deviate significantly in RMSE relative to their neighboring design points. Within this limited geometry variation test, we also do not observe a strong dependence of RMSE on sweep angle variations. These results support our speculation that a camber variation may introduce larger prediction errors, while overall `VortexNet` maintains moderate generalizability when presented with unseen geometries, as evidenced by the consistent prediction accuracy for the 60° and 70° NACA 0013 wings.

The exact nature of the prediction accuracy variation across the design space is a topic for future investigation. First, a larger unseen geometry test set is needed to examine the model's prediction accuracy more thoroughly and confirm whether these deviations are persistent. Second, the discrepancy could be due to insufficient training samples provided with camber variations. Expanding the design space coverage to include more camber shapes in the `VortexNet` training set may help reduce prediction errors. Finally, panel distribution used in VLM may need improvements. Currently, even chord-wise spacing is used, and this spacing scheme may be inadequate for capturing surface curvature variations, particularly near the leading and trailing edges.

A qualitative study was also conducted by visually examining the resulting ΔC_p^{pred} fields with reference to VLM and CFD ΔC_p results. For comparison across geometries, we present the ΔC_p for all geometries at AOA of 18°, Ma of 0.4 and Re of 7×10^6. The results are presented in Figure 7.

When comparing `VortexNet`'s prediction accuracy to the results shown in Figure 6, we observe that for previously unseen geometries, the surrogate model introduces more artifacts in its predictions. For the 70° NACA 0013 configuration,

Sweep Angle [degrees]	Root Airfoil (NACA 4-digits)	
	0013	**3416**
60	0.185	0.323
70	0.257	0.389

Table 8 `VortexNet` prediction RMSE for each new geometries.

the surrogate model fails to accurately capture the highest ΔC_p values. Small wavelength artifacts of high- and low-pressure variation can be seen for 70° geometries along the chord at half span y-location ($y/b = 0.5$). Nevertheless, when considered alongside other fidelity methods, `VortexNet` still demonstrates intermediate capability between VLM and CFD in capturing flow features. The authors believe that by refining the surrogate model architecture, discussed in Section Prediction Explanations, and improving the training data sampling scheme, the model's generalizability can be further enhanced.

C. Prediction Explanations

While the proposed `VortexNet` demonstrates strong prediction accuracy and generalizability, the underlying mechanism by which the GNN leverages and aggregates input graph features remains unclear. The complexity of GNNs makes extracting human-intelligible explanations from trained networks challenging [54]. Nevertheless, the authors have conducted a preliminary analysis of the `VortexNet`'s latent space. This section aims to understand how input graph features are utilized across latent layers and provide guidance for refining future `VortexNet`-like GNN architectures.

Among various methods for latent space analysis, we performed an ablation study on latent space features to investigate the resultant ΔC_p predictions. The best-performing model, as discussed in Section III.A, was used. For the data sample, we selected a run with a 55-degree sweep, NACA2416 root airfoil, $M_a = 0.47$, and $Re = 9.78 \times 10^6$ (the same case shown in Figure 6). The study begins with the latent space after the first Convolutional Block in Figure 1. At this stage, the nodal feature array was recorded, and clustering was performed on the feature array. Since the node features exist in a high-dimensional space ($\mathbb{R}^{\mathbb{F}\times\texttt{heads}}$), the t-SNE dimensionality reduction technique was applied to project the feature space to two dimensions [55]. K-means clustering was then used to classify nodes into four clusters [56], and their corresponding nodal indices were recorded.

The ablation study involves recursively isolating the contribution of each cluster by zeroing out nodal feature arrays belonging to other clusters at the current layer. This operation is repeated across all Convolutional Block layers. For instance, when analyzing the contribution of cluster one after the first Convolutional Block (layer 0), the nodal feature arrays of all other clusters at the current layer were manually set to zero. The modified feature array was then propagated through subsequent Convolutional Blocks to the final ΔC_p prediction. This approach isolates the local loading coefficient contributions from each cluster.

For deeper layers beyond the first Convolutional Block, the clustering definition obtained after the first Convolutional Block was reused by keeping track of the associativity between the nodal index and the cluster bins. At a specific layer, nodal feature arrays corresponding to a specific cluster were retained, while others were set to zero, and the resultant feature arrays were propagated to the Output Block. By analyzing ΔC_p predictions of a single cluster across multiple layers, the effects of multi-hop convolution can be studied. This controlled modification allows for an analysis of how individual layers influence the final output.

The results of the ablation study are presented in Figure 8. Each column corresponds to ΔC_p predictions using the nodal features of a specific cluster, while each row represents the effects of conducting the ablation study at a particular layer. The final row (layer 3) corresponds to the model's final output ΔC_p for each clusters. When the pressure fields of all clusters are superposed, the result recovers the `VortexNet`'s prediction on ΔC_p^{pred} shown in Figure 6.

Two trend-wise insights can be identified from this figure. Firstly, different clusters correspond to distinct regions of the wing. For example, "Cluster 0" corresponds to the leading edge of the wing, while "Cluster 3" focuses more on the afterbody. "Cluster 1" and "Cluster 2" primarily correspond to the left and right outboard sections of the wing, as evident from the ΔC_p prediction differences at "Layer 0." However, this distinction becomes less apparent across deeper layers.

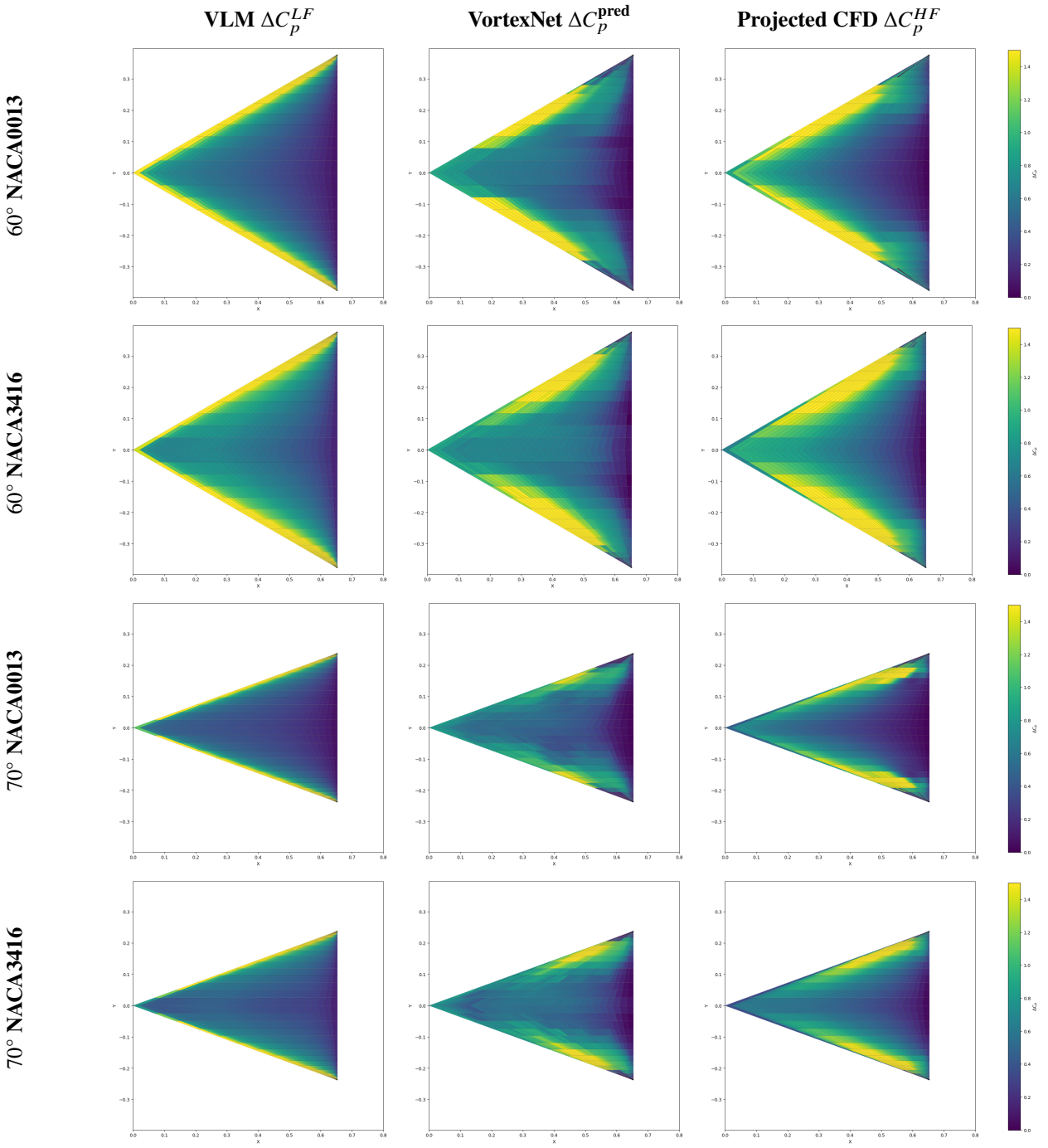

Fig. 7 Lattice panels and local loading coefficients ΔC_p are visualized using color maps ranging from $\Delta C_p = 0$ to $\Delta C_p = 1.5$ for four new geometries. From top to bottom: 60° NACA 0013, 60° NACA 3416, 70° NACA 0013, and 70° NACA 3416. From left to right: ΔC_p^{LF} obtained from VLM, `VortexNet` ΔC_p^{pred} predictions, and projected CFD ΔC_p^{HF}.

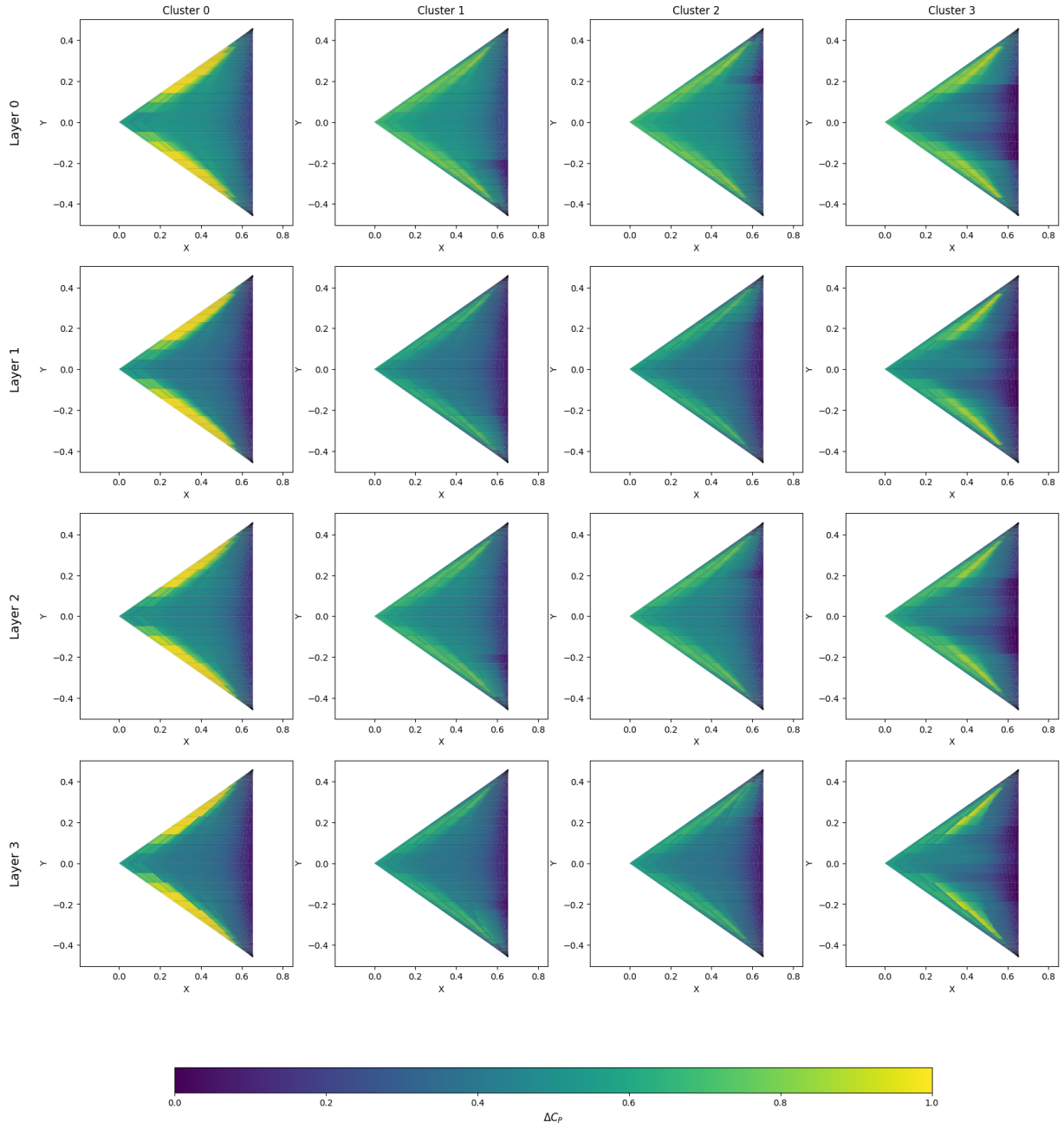

Fig. 8 Ablation study results illustrating ΔC_p predictions for node features within each cluster (columns). The rows represent the effects of the ablation study conducted at specific layers.

Secondly, across layers, field features become increasingly "localized." In the earlier layers, most field features exhibit large wavelengths, corresponding to global aerodynamic characteristics. In the later layers, the features shift to smaller wavelengths, and sharp discontinuities sometimes appear. This observation aligns with the expected behavior of such U-net like deep networks: deeper layers distill more localized features while shallower layers attention on more global features.

To assist in relating these clusters to specific regions of the wing, Figure 9 is provided. This figure shows by color coding of the clusters in Figure 8 and their corresponding position on the wing. Certain selected nodal indices are also presented along with their nodes to help assist future discussion.

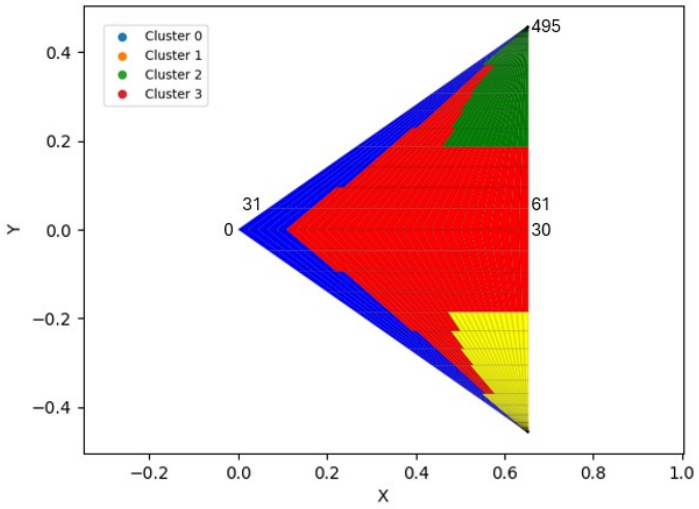

Fig. 9 Identified cluster locations on the wing, with several nodal indices marked along the nodes for reference.

To quantitatively visualize the ΔC_p prediction from this study, we compared the relative error (ϵ) between the prediction from one isolated cluster ($\Delta C_p^{\text{isolated}}$) at a specific layer and the unmodified ΔC_{pred} prediction. The relative error is defined as:

$$\epsilon(\Delta C_p^{\text{isolated}}) = \frac{\Delta C_p^{\text{isolated}} - \Delta C_p^{\text{pred}}}{\Delta C_p^{\text{pred}}}. \tag{24}$$

This relative error is a vector with a length equal to the number of nodes. In Figure 10, this vector is plotted along the y-axis and ordered by nodal index. The nodal index of 0 corresponds to the wing apex, while indices 495 and 900 correspond to the right tip and left tip of the wing, respectively, as illustrated in Figure 9. Along the x-axis, each column represents a specific "layer-cluster" combination. For instance, the x-tick labeled “01” corresponds to the relative error $\Delta C_p^{\text{isolated}}$ obtained by isolating cluster 1 at layer 0.

From Figure 10, comparing the locations of high-error nodes (marked in yellow) across layers confirms that different clusters attend to distinct regions of the wing, as high error appears in different areas for each cluster. For instance, in “Layer 0”, high-error nodes are primarily located along the leading and trailing edges of the wing. For example, when analyzing the relative error of Layer 0 Cluster 3 (marked by x-label “03”), where the cluster mainly focuses on the afterbody, the leading-edge prediction exhibits high error. Thus a periodic high-error pattern can be observed in nodal indices that are multiples of 30.

When comparing error patterns across layers, a more intriguing trend emerges. Globally, there are oblique “corridors” of high-error nodes that shift to higher nodal indices as layers become deeper. After correlating the high error nodal index to physical geometry, it is identified that these corridors represent high-error regions moving from the outer edges of the wing toward the inner regions. This observation aligns with the message-passing mechanism of the GNN, where nodal features are aggregated from neighboring nodes at each hop. The slope of these corridors indicates the speed at which information propagates through the graph. From the results in Figure 10, it is evident that information propagation occurs at a relatively low speed. By the final Convolution Block, much of the information has not reached the chord-wise center of the wing (e.g., nodal indices that are multiples of 15). This performance is suboptimal because, for information to traverse the wing geometry, a significantly deeper network would be required. However, deeper networks are more challenging to train and prone to high-variance error.

To address this limitation, improved edge definitions that connect nodes beyond direct neighbors could be implemented. Alternatively, using a fully connected graph may help mitigate this issue, though the implications of adopting such a graph remain unclear as the addition of trainable parameters may decrease the prediction accuracy under limited HF samples. Additionally, since pooling layers are not utilized in this network, we believe that incorporating a properly designed pooling mechanism could assist information propagation by reducing the computational graph size toward deeper layers. This highlights the need for future research on optimizing graph definitions and model architectures in `VortexNet`-like surrogate modeling techniques.

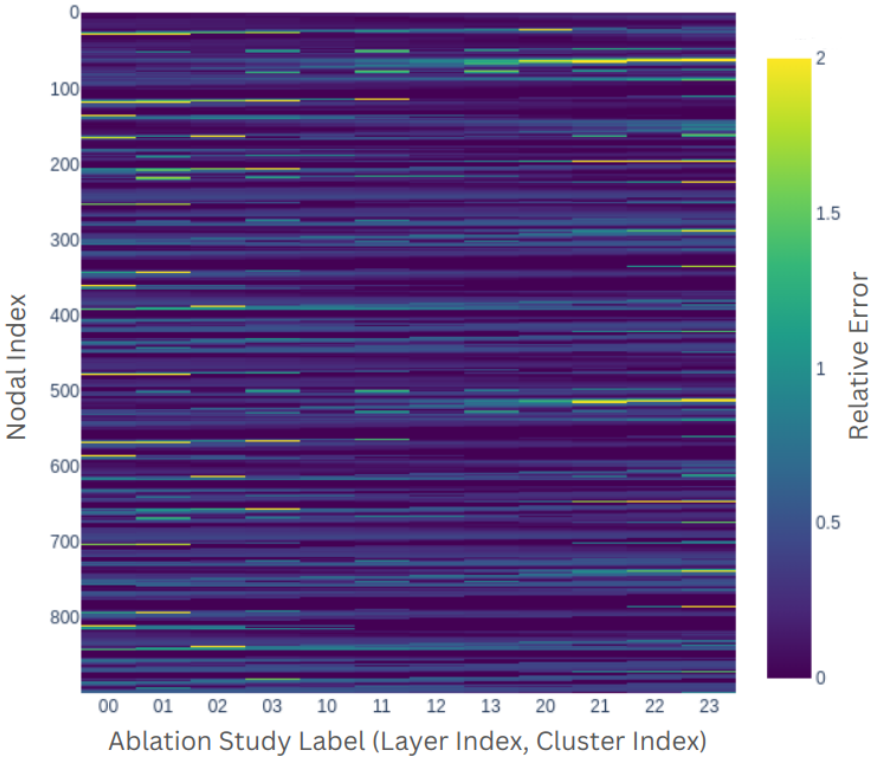

Fig. 10 Relative error (ϵ) for ΔC_p predictions under the ablation study, plotted by graph nodal index (VLM control points) along the y-axis and "layer-cluster" combinations along the x-axis.

Finally, the latent spaces after t-SNE projection at layer 0 to 2 are visualized in Figure 11. At shallower layers, as shown in Figure 11(a), the distribution of nodal feature clusters loosely follows the geographical distribution of the corresponding nodes in physical space. For instance, one can still identify the relative positions of the leading edges, afterbody, left outboard wing, and right outboard sections from the t-SNE axes. However, in deeper layers, this grouping by physical space separation diminishes. In Figure 11(b), the clusters corresponding to the left and right outboard sections merge. This phenomenon suggests that, at this layer, information related to flow conditions becomes more dominant. Since the flow conditions for the left and right outboard sections are symmetric due to the free-stream conditions in the training dataset, the geographical distinctions in physical space become irrelevant.

At even deeper layers, as shown in Figure 11(c), the physical space separation becomes nearly indistinguishable, and the clustering instead reflects how different geographical components share similar flow conditions. Examining the distribution of features across the space, one may argue that the latent layers exhibit some level of coherent manifold structure. However, as pooling layers are not utilized in the current architecture, the resultant visualization appears too noisy, and the presence and significance of such manifolds remain unclear. Further research is needed to incorporate pooling mechanisms into the graph definition and investigate whether the latent space presents a coherent manifold structure.

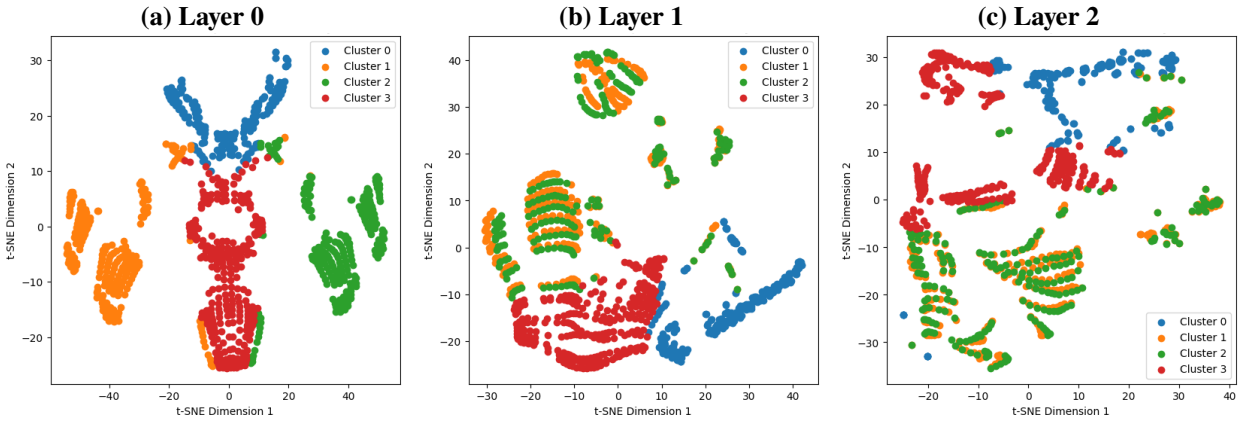

Fig. 11 Projected latent space nodal features plotted against the t-SNE axes, with scatter dot colors indicating the corresponding clusters of the nodes.

IV. Conclusions and Future Work

In this work, we propose a novel graph neural network (GNN)-based surrogate model named `VortexNet` for aircraft conceptual design. The surrogate model integrates high-fidelity (HF) computational fluid dynamics (CFD) data into field data predictions generated by low-fidelity (LF) prediction tools, such as vortex lattice method (VLM). Compared to traditional surrogate modeling techniques, which primarily focus on specific scalar variables, the field data prediction capability allows for arbitrary Quantities of interests (QOIs) to be evaluated within conceptual design environments. By leveraging GNN, the surrogate model learns a mapping of field data differences between HF and LF predictions. Once trained, the surrogate model can generate quasi-HF field data predictions that capture the pressure field impacted by nonlinear flow physics at a cost comparable to that of running a VLM, thereby addressing the fidelity-cost trade-off challenge in the conceptual design phase.

The `VortexNet` architecture utilizes U-net-style skip connections for global feature capturing and employs graph attention network (GAT) convolutional blocks for local flow feature capturing, such as vortex lift and flow separation. Data assembly methods, implicit physical constraint enforcement, and training schemes tailored to `VortexNet` are also explored and presented. To optimize model prediction accuracy, a hyper-parameter optimization study was conducted, identifying specific hyper-parameters that need closer attention in future `VortexNet`-like surrogate model training.

The model's prediction accuracy is evaluated both qualitatively and quantitatively. When provided with an input graph seen during training, the surrogate model demonstrates strong performance in reconstructing qualitative flow features and achieves low prediction RMSE, establishing significant fidelity improvements over LF-based aerodynamic prediction methods. The model's generalizability is also assessed, showing moderate capability in predicting pressure

fields for unseen geometries. However, performance declines are observed when wing camber is varied, highlighting areas for further refinement. Despite these limitations, the model demonstrates potential as a mesh-agnostic surrogate. Such capability is critical for conceptual aircraft design under design space exploration. Finally, a latent space ablation study is conducted to preliminarily explain the prediction mechanism of the proposed network. The study reveals that `VortexNet` achieves its prediction capability by effectively blending geometrical information and flow features in the latent space. It also indicates a need for future improvements in graph topology and model architecture, as the current information-passing mechanism among graph nodes operates inefficiently.

Future work will focus on enhancing the model's generalizability and improving feature-capturing efficiency for more complex geometries, graphs, and broader aerodynamic conditions. We believe that the model's prediction capability can be further enhanced through the introduction of pooling mechanisms, refinement of edge connectivity graphs, and the use of more representative training datasets. The application of the proposed surrogate model for interpolating between fidelity levels beyond VLM and RANS CFD also presents an intriguing avenue for further studies. Future work may also include comparison with other surrogate modeling methodologies for field data prediction in order to characterize training data requirements, prediction accuracy, and computational cost for design space exploration tasks, relative to comparable methods.

V. Acknowledgment

This work was supported by the Air Force Office of Scientific Research under Grant FA9550-22-1-0004 and NASA, Award 80NSSC19K1661, under the Commercial Supersonics Technology (CST) program, Supersonic Configurations at Low Speeds (SCALOS), with Sarah Langston as the NASA technical grant monitor. The authors extend their gratitude to Eli Livne, Kuang-Ying Ting, Chester Nelson, and Kenneth Wiersema for their valuable discussions on supersonic aircraft configurations. The first author also wishes to thank Emilio Botero for his discussion of the `SUAVE` software environment and Ran Jiang for her assistance with the schematic drawing. Additionally, the authors would like to thank the Stanford Research Computing Center (SRCC) for providing computational resources on the Sherlock cluster and the Google Cloud Research Credit Program for granting access to GPU resources on the Google Cloud Platform.

References

[1] Brunton, S. L., Noack, B. R., and Koumoutsakos, P., "Machine Learning for Fluid Mechanics," *Annual Review of Fluid Mechanics*, Vol. 52, 2020, pp. 477–508. https://doi.org/10.1146/annurev-fluid-010719-060214.

[2] Shen, Y., Patel, H. C., Xu, Z., and Alonso, J. J., "Application of Multi-Fidelity Transfer Learning with Autoencoders for Efficient Construction of Surrogate Models," *AIAA SCITECH 2024 Forum*, 2024, pp. 1–19. https://doi.org/10.2514/6.2024-0013.

[3] Neufeld, D., Chung, J., and Behdinian, K., "Aircraft Conceptual Design Optimization Considering Fidelity Uncertainties," *Journal of Aircraft*, Vol. 48, No. 5, 2011, pp. 1602–1612. https://doi.org/10.2514/1.C031312.

[4] Raymer, D. P., *Aircraft design: A conceptual approach*, 4th ed., AIAA education series, American Institute of Aeronautics and Astronautics, Reston, Va., 2006. URL http://www.loc.gov/catdir/toc/ecip068/2006004706.html.

[5] Martins, J. R., "A Coupled-Adjoint Method for High-fidelity Aero-structural Optimization," Ph.D. thesis, Stanford University, 2002.

[6] Wang, L., Diskin, B., Biedron, R. T., Nielsen, E. J., and Bauchau, O. A., "High-Fidelity Multidisciplinary Sensitivity Analysis and Design Optimization for Rotorcraft Applications," *AIAA Journal*, Vol. 57, No. 8, 2019, pp. 3117–3131. https://doi.org/10.2514/1.J056587.

[7] Wendorff, A., Variyar, A., Ilario, C., Botero, E., Capristan, F., Smart, J., Alonso, J., Kulik, L., Clarke, M., Colonno, M., Kruger, M., Vegh, J. M., Goncalves, P., Erhard, R., Fenrich, R., Orra, T., St. Francis, T., MacDonald, T., Momose, T., Economon, T., Lukaczyk, T., and Maier, W., "SUAVE: An Aerospace Vehicle Environment for Designing Future Aircraft," , 2020. URL https://github.com/suavecode/SUAVE.

[8] MacDonald, T., Clarke, M., Botero, E. M., Vegh, J. M., and Alonso, J. J., "SUAVE: An Open-Source Environment Enabling Multi-Fidelity Vehicle Optimization," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017. https://doi.org/10.2514/6.2017-4437.

[9] Owens, D., "Weissinger's model of the nonlinear lifting-line method for aircraft design," *36th AIAA Aerospace Sciences Meeting and Exhibit*, 1998. https://doi.org/10.2514/6.1998-597.

[10] Budziak, K., *Aerodynamic Analysis with Athena Vortex Lattice (AVL)*, Hamburg: Aircraft Design and Systems Group (AERO), Department of Automotive and Aeronautical Engineering, 2015. URL https://www.fzt.haw-hamburg.de/pers/Scholz/arbeiten/TextBudziak.pdf.

[11] Miranda, L. R., Elliott, R. D., and Baker, W. M., "A generalized vortex lattice method for subsonic and supersonic flow applications," *NASA. Langley Res. Center Vortex-Lattice Utilization*, 1977. URL https://ntrs.nasa.gov/citations/19780008059.

[12] *Vortex-Lattice Utilization*, National Aeronautics and Space Administration, 1976. URL https://ntrs.nasa.gov/api/citations/19760021075/downloads/19760021075.pdf.

[13] Joshi, H., and Thomas, P., "Review of vortex lattice method for supersonic aircraft design," *The Aeronautical Journal*, 2023, pp. 1–35. https://doi.org/10.1017/aer.2023.25.

[14] Polhamus, E. C., "A concept of the vortex lift of sharp-edge delta wings based on a leading-edge-suction analogy," *NASA. Langley Res. Center*, 1966. URL https://ntrs.nasa.gov/citations/19670003842.

[15] Huynh, D., Pasquale, D. D., Prince, S., and Ahuja, V., "Application of a Semi-Empirical Method to Model Subsonic Vortex Lift over Sharp Leading-Edge Delta Wings," *AIAA SCITECH 2023 Forum*, 2023. https://doi.org/10.2514/6.2023-2457.

[16] Miller, D. S., and Wood, R. M., "Leeside flows over delta wings at supersonic speeds," *Journal of Aircraft*, Vol. 21, No. 9, 1984, pp. 680–686. https://doi.org/10.2514/3.45014.

[17] Kennedy, M. C., and O'Hagan, A., "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 63, No. 3, 2001, pp. 425–464. https://doi.org/https://doi.org/10.1111/1467-9868.00294.

[18] Mukhopadhaya, J., Whitehead, B. T., Quindlen, J. F., Alonso, J. J., and Cary, A. W., "Multi-fidelity modeling of probabilistic aerodynamic databases for use in aerospace engineering," *International Journal for Uncertainty Quantification*, Vol. 10, No. 5, 2020, pp. 425–447. https://doi.org/10.1615/Int.J.UncertaintyQuantification.2020032841.

[19] Berkooz, G., Holmes, P., and Lumley, J. L., "The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows," *Annual Review of Fluid Mechanics*, Vol. 25, 1993, pp. 539–575. https://doi.org/https://doi.org/10.1146/ANNUREV.FL.25.010193.002543.

[20] Mrosek, M., Othmer, C., and Radespiel, R., "Reduced-Order Modeling of Vehicle Aerodynamics via Proper Orthogonal Decomposition," *SAE Int. J. Passeng. Cars - Mech. Syst.*, Vol. 12, 2019, pp. 225–236. https://doi.org/10.4271/06-12-03-0016.

[21] Zhang, X., Xie, F., Ji, T., Zhu, Z., and Zheng, Y., "Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization," *Computer Methods in Applied Mechanics and Engineering*, Vol. 373, 2021, p. 113485. https://doi.org/10.1016/j.cma.2020.113485.

[22] Black, N., and Najafi, A. R., "Learning finite element convergence with the Multi-fidelity Graph Neural Network," *Computer Methods in Applied Mechanics and Engineering*, Vol. 397, 2022, p. 115120. https://doi.org/10.1016/j.cma.2022.115120.

[23] Li, J., Li, Y., Liu, T., Zhang, D., and Xie, Y., "Multi-fidelity graph neural network for flow field data fusion of turbomachinery," *Energy*, Vol. 285, No. July, 2023, p. 129405. https://doi.org/10.1016/j.energy.2023.129405.

[24] Chen, J., Hachem, E., and Viquerat, J., "Graph neural networks for laminar flow prediction around random two-dimensional shapes," *Physics of Fluids*, Vol. 33, No. 12, 2021. https://doi.org/10.1063/5.0064108.

[25] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W., "Learning Mesh-Based Simulation with Graph Networks," , 2021. URL https://arxiv.org/abs/2010.03409.

[26] Shao, X., Liu, Z., Zhang, S., Zhao, Z., and Hu, C., "PIGNN-CFD: A physics-informed graph neural network for rapid predicting urban wind field defined on unstructured mesh," *Building and Environment*, Vol. 232, 2023, p. 110056. https://doi.org/https://doi.org/10.1016/j.buildenv.2023.110056.

[27] Yang, S., Vinuesa, R., and Kang, N., "Enhancing Graph U-Nets for Mesh-Agnostic Spatio-Temporal Flow Prediction," , 2024. URL https://arxiv.org/abs/2406.03789.

[28] Liu, Q., Zhu, W., Jia, X., Ma, F., and Gao, Y., "Fluid Simulation System Based on Graph Neural Network," , 2022. URL https://arxiv.org/abs/2202.12619.

[29] Gao, H., and Ji, S., "Graph U-Nets," *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov, PMLR, 2019, pp. 2083–2092. URL http://proceedings.mlr.press/v97/gao19a.html.

[30] Wei, L., and Freris, N. M., "Multi-scale graph neural network for physics-informed fluid simulation," *The Visual Computer*, 2024. https://doi.org/10.1007/s00371-024-03402-6.

[31] Deshpande, S., Bordas, S. P., and Lengiewicz, J., "MAgNET: A graph U-Net architecture for mesh-based simulations," *Engineering Applications of Artificial Intelligence*, Vol. 133, 2024, p. 108055. https://doi.org/10.1016/j.engappai.2024.108055.

[32] Palacios, F., Alonso, J., Duraisamy, K., Colonno, M., Hicken, J., Aranake, A., Campos, A., Copeland, S., Economon, T., Lonkar, A., et al., "Stanford University Unstructured (SU2): an Open-source Integrated Computational Environment for Multi-physics Simulation and Design," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013, p. 287. https://doi.org/https://doi.org/10.2514/6.2013-287.

[33] Palacios, F., Economon, T. D., Aranake, A., Copeland, S. R., Lonkar, A. K., Lukaczyk, T. W., Manosalvas, D. E., Naik, K. R., Padron, S., Tracey, B., et al., "Stanford University Unstructured (SU2): Analysis and Design Technology for Turbulent Flows," *52nd Aerospace Sciences Meeting*, 2014, p. 0243. https://doi.org/https://doi.org/10.2514/6.2014-0243.

[34] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J., "SU2: An Open-Source Suite for Multiphysics Simulation and Design," *AIAA Journal*, Vol. 54, No. 3, 2016. https://doi.org/10.2514/1.J053813.

[35] Spalart, P., and Allmaras, S., "A one-equation turbulence model for aerodynamic flows," *30th Aerospace Sciences Meeting and Exhibit*, 1992. https://doi.org/10.2514/6.1992-439.

[36] Shur, M. L., Strelets, M. K., Travin, A. K., and Spalart, P. R., "Turbulence Modeling in Rotating and Curved Channels: Assessing the Spalart-Shur Correction," *AIAA Journal*, Vol. 38, No. 5, 2000, pp. 784–792. https://doi.org/10.2514/2.1058.

[37] Seraj, S., and Martins, J. R. R. A., "Predicting the High-Angle-of-Attack Characteristics of a Delta Wing at Low Speed," *Journal of Aircraft*, Vol. 59, No. 4, 2022, pp. 1071–1081. https://doi.org/10.2514/1.C036618.

[38] Jameson, A., Schmidt, W., and Turkel, E., "Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes," *14th Fluid and Plasma Dynamics Conference*, 1981. https://doi.org/10.2514/6.1981-1259.

[39] Pointwise, "Pointwise, Cadence Design Systems," , retrived 26 April 2023. URL http://www.pointwise.com/.

[40] PyTorch, "Tanh Standardization," , retrived 30 November 2024. URL https://pytorch.org/docs/stable/generated/torch.nn.Tanh.html.

[41] Hamilton, W. L., Ying, R., and Leskovec, J., "Representation Learning on Graphs: Methods and Applications," *CoRR*, Vol. abs/1709.05584, 2017. URL http://arxiv.org/abs/1709.05584.

[42] Zhao, Y., Li, H., Zhou, H., Attar, H. R., Pfaff, T., and Li, N., "A review of graph neural network applications in mechanics-related domains," *Artificial Intelligence Review*, Vol. 57, No. 11, 2024, p. 315. https://doi.org/10.1007/s10462-024-10931-y.

[43] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S., "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 32, No. 1, 2021, p. 4–24. https://doi.org/10.1109/tnnls.2020.2978386.

[44] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y., "Graph Attention Networks," *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[45] Brody, S., Alon, U., and Yahav, E., "How Attentive are Graph Attention Networks?" , 2022. URL https://arxiv.org/abs/2105.14491.

[46] Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., Ridella, S., et al., "The'K'in K-fold Cross Validation." *ESANN*, Vol. 102, 2012, pp. 441–446. URL https://www.esann.org/sites/default/files/proceedings/legacy/es2012-62.pdf.

[47] PyTorch, C., "Smoothl1loss," , retrived 30 November 2024. URL https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html.

[48] Eivazi, H., Tahani, M., Schlatter, P., and Vinuesa, R., "Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations," *Physics of Fluids*, Vol. 34, No. 7, 2022, p. 075117. https://doi.org/10.1063/5.0095270.

[49] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun, 2015. URL http://arxiv.org/abs/1412.6980.

[50] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M., "Optuna: A Next-generation Hyperparameter Optimization Framework," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Association for Computing Machinery, New York, NY, USA, 2019, p. 2623–2631. https://doi.org/10.1145/3292500.3330701, URL https://doi.org/10.1145/3292500.3330701.

[51] Smith, R. C., "Uncertainty quantification: Theory, implementation, and applications," SIAM, 2013, pp. 321–344.

[52] Frey, H. C., and Patil, S. R., "Identification and review of sensitivity analysis methods," *Risk Anal*, Vol. 22, No. 3, 2002, pp. 553–578. https://doi.org/10.1111/0272-4332.00039.

[53] Shen, Y., and Alonso, J. J., "Performance Evaluation of a Graph Neural Network-Augmented Multi-Fidelity Workflow for Predicting Aerodynamic Coefficients on Delta Wings at Low Speed," *AIAA SCITECH 2025 Forum*, 2025.

[54] Ying, R., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J., "GNNExplainer: Generating Explanations for Graph Neural Networks," , 2019. URL https://arxiv.org/abs/1903.03894.

[55] Van der Maaten, L., and Hinton, G., "Visualizing data using t-SNE." *Journal of machine learning research*, Vol. 9, No. 86, 2008, pp. 2579–2605. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

[56] MacQueen, J., "Some methods for classification and analysis of multivariate observations," *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*, 1967. URL https://projecteuclid.org/ebooks/berkeley-symposium-on-mathematical-statistics-and-probability/Some-methods-for-classification-and-analysis-of-multivariate-observations/chapter/Some-methods-for-classification-and-analysis-of-multivariate-observations/bsmsp/1200512992.