AutoHood3D: A Multi-Modal Benchmark for Automotive Hood Design and Fluid–Structure Interaction

Vansh Sharma[*]
University of Michigan
Ann Arbor, MI, USA

Harish Jai Ganesh
University of Michigan
Ann Arbor, MI, USA

Maryam Akram
Ford Research and Innovation Center
Dearborn, MI, USA

Wanjiao Liu
Ford Research and Innovation Center
Dearborn, MI, USA

Venkat Raman
University of Michigan
Ann Arbor, MI, USA

Abstract

AutoHood3D is a high-fidelity data-generation framework and multimodal dataset of 16k+ automotive-hood variants targeting a practical multiphysics challenge: deformation from fluid entrapment and inertial loading during rotary-dip painting. Each sample couples LES–FEA (1.2M cells) and provides time-resolved flow/structure fields, STL meshes, and structured language prompts. Unlike previous benchmarks (often 2D, low diversity, or missing multiphysics), AutoHood3D offers scalable 3D FSI, broad geometric diversity, and a fully reproducible pipeline. We benchmark point- and graph-based surrogates, establish in-distribution and out-of-distribution baselines, and reveal systematic prediction errors in displacement and force; point-based models excel in-distribution, while graph-based models generalize better to novel geometries. The results motivate consistent multiphysics-aware losses that enforce fluid–solid coupling. AutoHood3D provides a reproducible foundation for physics-aware surrogates, rapid generative design, and new 3D FSI benchmarks. Dataset and code URLs in A.1.

1 Introduction

Classical numerical solvers [1, 2] deliver accurate predictions for aircraft [3, 4] and turbines [5, 6] but are costly and often impractical for iterative design [7]. Recent work fuses ML with CFD for turbulence closures [8, 9], wall-modeled LES [10], and real-time control [11, 12], improving accuracy [13] and time-to-solution [14, 15]. Large SciML repositories—PDEBench [16], PINNacle [17], Airfrans [18], BubbleML [19]—and cross-domain resources such as BLASTNet [20] and ThePDEWell [21] enable systematic model development and benchmarking. Industrial datasets like DrivAerNet++ [22] and WindsorML [23] offer realistic scenarios for pressures, flows, and loads. However, gaps remain: lack of reproducible scalable workflows, physics processes represented with 2D idealizations, end-to-end pipelines are frequently low-fidelity (RANS or 2D) [18, 19]; and 3D resources limit design diversity due to cost [23]. Critically, multiphysics (e.g., multiphase, fluid-structure interaction (FSI)) is underrepresented, and no public resource pairs broad engineering design variation with a reproducible, large-scale 3D FSI benchmark and multimodal output (STL, point clouds).

In this work, we introduce *AutoHood3D* (Figure 1): +16000 3D hood geometries with time-resolved FSI, targeting a multiphysics challenge—hood deformation during rotary dip painting [24, 25]. Hoods

[*]Corresponding Author: vanshs@umich.edu

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

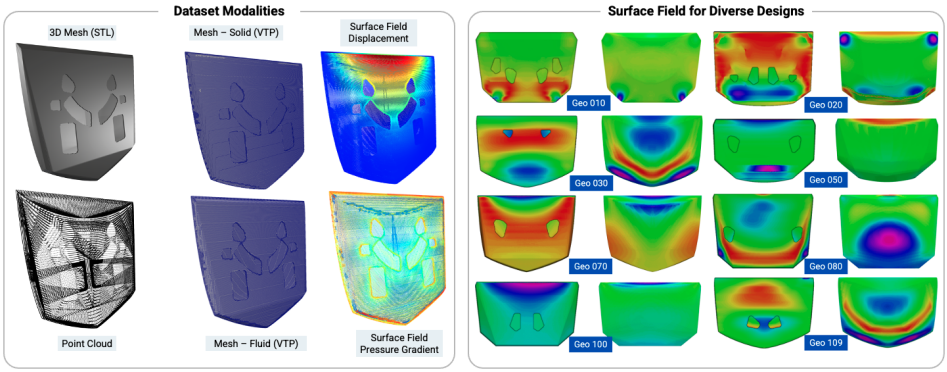

Figure 1: Dataset modalities (left): base 3D STL hood shell, surface point-cloud, raw CFD/FEA meshes, and sample surface fields extracted on the hood surface. Example variants (right): selected hood geometries with interpolated deformation fields mapped onto the STL, demonstrating how different cutout topologies yield distinct deflection patterns on both the front and rear faces.

comprise an aerodynamic outer panel and a reinforced inner lattice; beyond aerodynamics, design must satisfy pedestrian-safety and utility constraints [26, 27]. During dip painting, residual coating fluid trapped in cutouts induces transient body forces that, with inertia, cause localized deflections [24], leading to unexpected deformations in thin, complex regions (analogous to immersion-coated composites or warping wings [28]). To study this and similar problems, we release open-source geometry generation, FSI co-simulation, and analysis workflows, enabling new 3D FSI benchmarks and physics-informed PDE learning [29]. **Our contributions:**

- Introduce the first open-source dataset of more than 16000 3D automotive hood geometries with engineered cutouts, for broader design studies.
- Provide a fully reproducible end-to-end workflow, that is scalable to new component variants and extensible to diverse engineering contexts.
- Deliver a multimodal benchmark comprising high-fidelity STL meshes, raw CFD solutions with time-resolved flow and surface fields, FEA outputs, point clouds, and tokenized natural-language annotations, with performance benchmarks for five neural architectures.

2 Dataset Workflow

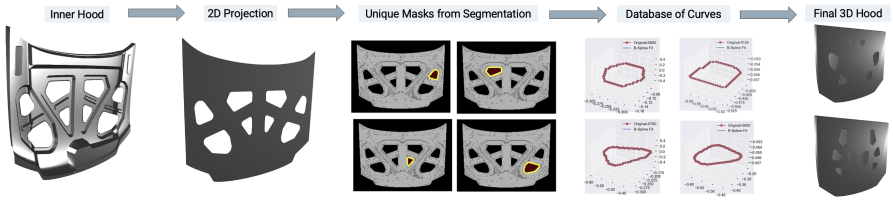

Figure 2: Workflow for generating multiple hood geometries. Starting from the base inner-hood CAD (left), the surface is projected onto a 2D plane and segmented into individual cutout masks. Each mask boundary is extracted as an ordered point-cloud curve. These curves are then re-embedded into the full 3D hood shell to produce the final geometries with engineered openings (right).

Base Geometry Generation We start from 100 inner-only hoods [30]. An outer envelope is formed through a convex hull [31], then extruded (10 mm) to yield dual shells. Cutouts are extracted by projecting the inner surface in 2D, segmenting with SAM-2 [32], and converting each mask boundary to an ordered point cloud (Fig. 2). The design database contains $\sim$ 1750 unique cutout profiles characterized by perimeter and area; K-Means on this $(\text{perimeter}, \text{area})$ space yield seven clusters [33]. New variants sample 1 to 4 curves across clusters with bilateral symmetry, enforcing (i) minimum in-plane spacing, and (ii) axial separation between mirrored planes (uniformly sampled within prescribed ranges). See A.3 for details.

Computational Setup All simulations use OpenFOAM v2312 [34]: the fluid solver (UMpimpleFoam) is an incompressible LES with Spalart–Allmaras DDES [35] and an added transient body-force term, while the structural solver (UM-solidDisplacementFoam) is small-strain linear elastic (Hooke). Because deformations are minor, we adopt one-way coupling (fluid→solid), adequate for similar FSI settings [36, 37, 38, 39]; solvers run concurrently, exchanging pressure each coupling step. Inter-solver communication uses preCICE [40] with the OpenFOAM adapter [41], mapping fluid pressure to the solid mesh via nearest-projection; both meshes originate from the same STL, keeping interpolation error negligible. Meshing employs SnappyHexMesh with targeted near-wall refinement [42, 23]; the fluid LES has ∼750k polyhedral cells ($\Delta = 3.125, \mathrm{mm}$), including ∼170–180k boundary-layer cells, and the solid surface mesh has ∼450k cells ($\Delta = 2.20, \mathrm{mm}$), sufficient for dominant deflection mode [43]. Boundary conditions: inlet applies a uniform acceleration impulse ($\alpha = 2.7, \mathrm{m/s^2}$ for $t = 0.07, \mathrm{s}$ at 30°) with sampling $\Delta t = 0.01, \mathrm{s}$ (8 snapshots); the fluid domain uses slip outer faces, a no-slip hood, and zero-gradient outflow; the structure is fully clamped with $\rho_s = 2700, \mathrm{kg/m^3}$, $\nu = 0.33$, and $E = 68.9 \times 10^9, \mathrm{N/m^2}$. Compute footprint: runs executed on a 16-node cluster (each with 8 NVIDIA H100 SXM 80GB GPUs) totaling 1536 CPU cores (Intel Xeon Platinum 8468, 2.1,GHz); each case used 12 cores and 72,GB; overall ≈5000 CPU-core hours and ≈2 TB across 37000 files.

Generative AI Prompts Using Gemma3-27B-IT [44], we built a paired text–point-cloud corpus of 2587 hoods. Prompts encoding cutout parameters (inter-curve spacing, center-plane offset, curve count) with geometry snapshots are fed to Gemma3's multimodal encoder, yielding a 3D point-cloud description of the shell and a chain-of-thought procedure. The corpus supports SFT to map language specs to 3D point clouds, enabling zero-shot shape synthesis, parametric editing, etc. (see A).

3 ML Benchmarking

Accurate prediction of fluid-induced hood deformations requires low-latency surrogates. Building on previous benchmarks [22, 18, 23], we evaluate five architectures—MLP, PointNet [45], GraphSAGE [46], Graph U-Net [47], and PointGNNConv [48]—to predict normalized deformation across geometries. Each model uses an MLP encoder to embed vertices/points (with auxiliary attributes such as surface pressure), a point/graph core to aggregate local–global context, and an MLP decoder to regress per-node outputs (hyper-parameter settings in Appendix B). Experiments use 4500 hoods split into train/val/id-test (75/15/10%) and a separate ood-test geometry; inputs are point coordinates, normals, and mesh connectivity, and the targets are U, p, and D. Training runs for 750 epochs with MSE loss (Graph U-Net at 320 epochs[2]). Performance is reported using MSE for both in-distribution and out-of-distribution tests, emphasizing sensitivity to large errors.

Model	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)
MLP	**0.25**	0.27	**0.44**	**0.37**	2.80	0.51	0.76
PointNet	0.31	**0.26**	0.48	0.55	**0.29**	**0.40**	**0.43**
GraphSAGE	4.89	1.31	2.04	4.22	14.88	3.98	6.86
Graph U-Net [2]	1.91	0.86	1.18	3.09	2.89	0.92	0.97
PointGNNConv	4.50	1.42	2.69	7.70	11.71	8.92	15.41

Table 1: ID Test: Mean squared error on the different normalized fields for all the models.

In-Distribution Tests From Table 1, simpler models—MLP and PointNet—outperform graph baselines: MLP attains the lowest MSE on U_x, U_z, and p, while PointNet provides the best overall balance. Graph models (e.g., GraphSAGE) underpredict peaks and tend to introduce oscillations near sharp geometry, a consequence of fixed-neighborhood trade-offs (over-smoothing fine cutouts vs. limited global context). The pronounced drop in D_x accuracy probably reflects $\|D\|$'s role as a pressure-integrated secondary field, which amplifies upstream U and p errors (see Appendix E).

Out-of-Distribution Test On OOD geometries (Table 2, Fig. 3), graph models lead: Graph U-Net achieves the best p MSE (4.05×10^{-2}) with competitive velocity errors, and PointGNNConv yields the lowest D_y/D_z. While MLP (U_x) and PointNet (D_x) are isolated winners, both degrade sharply on

[2]Reported at 320 epochs due to longer per-epoch time; its loss trajectory matches leading 750-epoch models.

Model	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)
MLP	**1.49**	1.89	2.97	4.96	181.02	5.87	19.81
PointNet	3.48	2.89	6.87	15.45	**73.79**	17.83	25.55
GraphSAGE	3.24	1.81	2.65	6.99	84.34	6.07	21.71
Graph U-Net [2]	2.04	1.29	1.45	**4.05**	126.62	7.69	16.13
PointGNNConv	2.84	**1.51**	**1.20**	5.79	130.08	**4.78**	**13.67**

Table 2: OOD Test: Mean squared error on the different normalized fields for all the models.

p and $|D|$. Plots confirm graph methods preserve sharp pressure features and localized deflection near cutouts, whereas simpler models over-smooth. Furthermore, see Appendix C for runtime analysis.

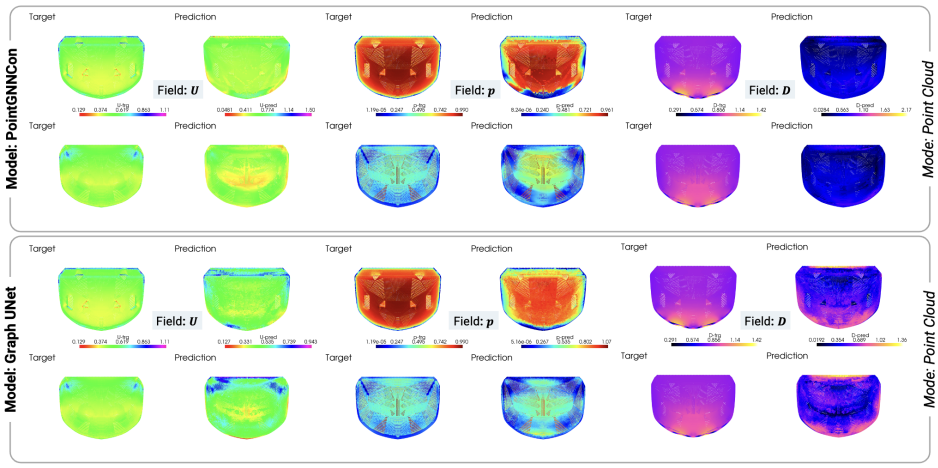

Figure 3: OOD test predictions for the top two graph-based models using point-cloud modality. For PointGNNConv (top block) and Graph U-Net [2] (bottom block), target (left) and predicted (right) fields are shown for $\|U\|$, p, and $\|D\|$ on the front (upper row of each block) and back (lower row of each block) surfaces of a representative OOD hood geometry.

4 Conclusion

We introduced *AutoHood3D*, the first open-source, large-scale 3D hood FSI benchmark and data generation framework: +16000 parametric variants and +108000 time-resolved LES–FEA snapshots on refined meshes, ensuring reproducibility and extensibility. The release spans STL, raw CFD/FEA meshes and fields, point clouds and structured prompts for text-to-geometry LLM fine-tuning. Baselines across five architectures quantify speed–accuracy trade-offs: MLP/PointNet (7–11 ms, ID MSE $\sim 10^{-3}$) compared to Graph U-Net/PointGNNConv (34–249 ms, OOD MSE down to 1.2×10^{-2}), guiding model selection. Several opportunities for improvement remain: extending cutouts from 2D projections to full 3D contours; broadening baselines to mesh-agnostic neural operators [49]; informing clustering step with curvature and modal descriptors; and replacing pure MSE with multiphysics-aware objectives that enforce fluid–solid balance, improving fidelity and design-space exploration. Future work should focus on nonlinear structural solvers, richer geometric/material parameters (e.g., mass distribution, heterogeneity), and rigorous analysis of text-to-geometry prompts on different LLMs to support generative AI workflows.

Acknowledgments and Disclosure of Funding

This work was supported at the University of Michigan by Ford Motor Company under the grant "Manufacturability-constrained closure design using physics-informed artificial intelligence".

References

[1] Hrvoje Jasak. Openfoam: Open source cfd in research and industry. *International journal of naval architecture and ocean engineering*, 1(2):89–94, 2009.

[2] Shivank Sharma, Ral Bielawski, Oliver Gibson, Shuzhi Zhang, Vansh Sharma, Andreas H Rauch, Jagmohan Singh, Sebastian Abisleiman, Michael Ullman, Shivam Barwey, et al. An amrex-based compressible reacting flow solver for high-speed reacting flows relevant to hypersonic propulsion. *arXiv preprint arXiv:2412.00900*, 2024.

[3] Jelle Houtman and Sebastian Timme. Global stability analysis of elastic aircraft in edge-of-the-envelope flow. *Journal of Fluid Mechanics*, 967, 7 2023.

[4] Sabet Seraj, Anil Yildirim, Joshua L. Anibal, and Joaquim R.R.A. Martins. Dissipation and time step scaling strategies for low and high Mach number flows. *Journal of Computational Physics*, 491:112358, 7 2023.

[5] Pier Carlo Nassini, Daniele Pampaloni, Roberto Meloni, and Antonio Andreini. Lean blow-out prediction in an industrial gas turbine combustor through a LES-based CFD analysis. *Combustion and Flame*, 229:111391, 3 2021.

[6] Maciej Chmielewski, Paweł Niszczota, and Marian Gieras. Combustion efficiency of fuel-water emulsion in a small gas turbine. *Energy*, 211:118961, 9 2020.

[7] Mouadh Yagoubi, David Danan, Milad Leyli-abadi, Jean-Patrick Brunet, Jocelyn Ahmed Mazari, Florent Bonnet, maroua gmati, Asma Farjallah, Paola Cinnella, Patrick Gallinari, and Marc Schoenauer. Neurips 2024 ml4cfd competition: Harnessing machine learning for computational fluid dynamics in airfoil design, 2024.

[8] Anudhyan Boral, Zhong Yi Wan, Leonardo Zepeda-Nunez, James Lottes, Qing Wang, Yi-Fan Chen, John Roberts Anderson, and Fei Sha. Neural ideal large eddy simulation: Modeling turbulence with neural stochastic differential equations. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[9] Toby van Gastelen, Wouter Edeling, and Benjamin Sanderse. Energy-conserving neural network for turbulence closure modeling. *Journal of Computational Physics*, 508:113003, 2024.

[10] Romit Maulik, Omer San, Jamey D Jacob, and Christopher Crick. Sub-grid scale model classification and blending through deep learning. *Journal of Fluid Mechanics*, 870:784–812, 2019.

[11] Long Wei, Peiyan Hu, Ruiqi Feng, Haodong Feng, Yixuan Du, Tao Zhang, Rui Wang, Yue Wang, Zhi-Ming Ma, and Tailin Wu. Diffphycon: A generative approach to control complex physical systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[12] Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. *Advances in Neural Information Processing Systems*, 31, 2018.

[13] Vansh Sharma, Michael Ullman, and Venkat Raman. A machine learning based approach for statistical analysis of detonation cells from soot foils. *Combustion and Flame*, 274:114026, 2025.

[14] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[15] Vansh Sharma, Andreas H Rauch, and Venkatramanan Raman. Accelerating cfd simulations with super-resolution feedback-informed adaptive mesh refinement. In *AIAA SCITECH 2025 Forum*, page 1467, 2025.

[16] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[17] Hao Zhongkai, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, et al. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. *Advances in Neural Information Processing Systems*, 37:76721–76774, 2024.

[18] Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022.

[19] Sheikh Md Shakeel Hassan, Arthur Feeney, Akash Dhruv, Jihoon Kim, Youngjoon Suh, Jaiyoung Ryu, Yoonjin Won, and Aparna Chandramowlishwaran. Bubbleml: A multiphase multiphysics dataset and benchmarks for machine learning. *Advances in Neural Information Processing Systems*, 36:418–449, 2023.

[20] Neiwen Ling, Xuan Huang, Zhihe Zhao, Nan Guan, Zhenyu Yan, and Guoliang Xing. Blastnet: Exploiting duo-blocks for cross-processor real-time dnn inference. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, pages 91–105, 2022.

[21] Ruben Ohana, Michael McCabe, Lucas Thibaut Meyer, Rudy Morel, Fruzsina Julia Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Stuart B. Dalziel, Drummond Buschman Fielding, Daniel Fortunato, Jared A. Goldberg, Keiya Hirashima, Yan-Fei Jiang, Rich Kerswell, Suryanarayana Maddu, Jonah M. Miller, Payel Mukhopadhyay, Stefan S. Nixon, Jeff Shen, Romain Watteaux, Bruno Régaldo-Saint Blancard, François Rozet, Liam Holden Parker, Miles Cranmer, and Shirley Ho. The well: a large-scale collection of diverse physics simulations for machine learning. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[22] Mohamed Elrefaie, Florin Morar, Angela Dai, and Faez Ahmed. Drivaernet++: A large-scale multimodal car dataset with computational fluid dynamics simulations and deep learning benchmarks. *Advances in Neural Information Processing Systems*, 37:499–536, 2024.

[23] Neil Ashton, Jordan Angel, Aditya Ghate, Gaetan Kenway, Man Long Wong, Cetin Kiris, Astrid Walle, Danielle Maddix, and Gary Page. Windsorml: High-fidelity computational fluid dynamics dataset for automotive aerodynamics. *Advances in Neural Information Processing Systems 37*, 2024.

[24] J Kim, S Park, N Kim, N Hur, and C Oh. Prediction and minimization of micro deformation on the automobile hood after dipping process. *International Journal of Automotive Technology*, 16:293–300, 2015.

[25] J Kim, Naksoo Kim, Nahmkeon Hur, and C Oh. Micro deformation of automobile hood in dipping process using stiffness. *International Journal of Automotive Technology*, 15:475–482, 2014.

[26] Mohammad Hassan Shojaeefard, Amir Najibi, and Meisam Rahmati Ahmadabadi. Pedestrian safety investigation of the new inner structure of the hood to mitigate the impact injury of the head. *Thin-Walled Structures*, 77:77–85, 12 2013.

[27] Zhijun Yang, Tao Deng, and Zhenfei Zhan. Design and analysis of vehicle active hood for pedestrian protection. *SAE technical papers on CD-ROM/SAE technical paper series*, 12 2021.

[28] C. Thill, J. Etches, I. Bond, K. Potter, and P. Weaver. Morphing skins. *The Aeronautical Journal*, 112(1129):117–139, 3 2008.

[29] Juan Diego Toscano, Vivek Oommen, Alan John Varghese, Zongren Zou, Nazanin Ahmadi Daryakenari, Chenxi Wu, and George Em Karniadakis. From pinns to pikans: Recent advances in physics-informed machine learning. *Machine Learning for Computational Science and Engineering*, 1(1):1–43, 2025.

[30] Patricia Wollstadt, Mariusz Bujny, Satchit Ramnath, Jami J. Shah, Duane Detwiler, and Stefan Menzel. Carhoods10k: An industry-grade data set for representation learning and design optimization in engineering applications. *IEEE Transactions on Evolutionary Computation*, 26(6):1221–1235, 2022.

[31] Yves Dejonghe. Py-madcad. `https://github.com/jimy-byerley/pymadcad`, 2024.

[32] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[33] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[34] Andrew Heather, Mattijs Janssens, Mark Olesen, Prashant Sonakar, Pawan Ghildiyal, Kutalmış Berçin, Matej Forman, Chiara Pesci, Martin Lichtmes, Jiri Polansky, Ann Ronchetti, Fred Mendonça, Swapnil Salokhe, and Venkata Ramana Eaga. OpenFOAM v2406 Software. Available at: https://www.openfoam.com/news/main-news/openfoam-v2312, 2023.

[35] M Kh Strelets. A new version of detached-eddy simulation, resistant to ambiguous grid densities. *Theoretical and computational fluid dynamics*, 20:181–195, 2006.

[36] Vilas J. Shinde, Jack J. McNamara, and Datta V. Gaitonde. One-Way response of a flexible panel to shock wave boundary layer interaction. *AIAA Aviation 2019 Forum*, 6 2023.

[37] Elliot Kimmel, Daning Huang, Vansh Sharma, Jagmohan Singh, Venkatramanan Raman, and Peretz P. Friedmann. Evaluation of Shock Wave-Boundary Layer interaction modeling capabilities for use in a hypersonic aerothermoelastic framework. *AIAA SCITECH 2022 Forum*, 1 2024.

[38] Friedrich-Karl Benra, Hans Josef Dohmen, Ji Pei, Sebastian Schuster, and Bo Wan. A comparison of One-Way and Two-Way coupling methods for numerical analysis of Fluid-Structure interactions. *Journal of Applied Mathematics*, 2011(1), 1 2011.

[39] Nora Hagmeyer, Matthias Mayr, Ivo Steinbrecher, and Alexander Popp. One-way coupled fluid–beam interaction: capturing the effect of embedded slender bodies on global fluid flow and vice versa. *Advanced Modeling and Simulation in Engineering Sciences*, 9(1), 6 2022.

[40] G Chourdakis, K Davis, B Rodenberg, M Schulte, F Simonis, B Uekermann, G Abrams, HJ Bungartz, L Cheung Yau, I Desai, K Eder, R Hertrich, F Lindner, A Rusch, D Sashko, D Schneider, A Totounferoush, D Volland, P Vollmer, and OZ Koseomur. preCICE v2: A sustainable and user-friendly coupling library [version 2; peer review: 2 approved]. *Open Research Europe*, 2(51), 2022.

[41] Gerasimos Chourdakis, David Schneider, and Benjamin Uekermann. OpenFOAM-preCICE: Coupling OpenFOAM with external solvers for multi-physics simulations. *OpenFOAM® Journal*, 3:1–25, Feb 2023.

[42] Angelina I. Heft, Thomas Indinger, and Nikolaus A. Adams. Introduction of a new realistic generic car model for aerodynamic investigations. *SAE technical papers on CD-ROM/SAE technical paper series*, 4 2012.

[43] Wadhah Garhuom, Simeon Hubrich, Lars Radtke, and Alexander Düster. A remeshing strategy for large deformations in the finite cell method. *Computers & Mathematics with Applications*, 80(11):2379–2398, 4 2020.

[44] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[45] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[46] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[47] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

[48] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020.

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

[50] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[51] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. In *International Conference on Learning Representations*, 2021.

[52] Mark EJ Newman. Mixing patterns in networks. *Physical review E*, 67(2):026126, 2003.

[53] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.

Technical Appendices and Supplementary Material

A Dataset

A.1 Dataset Structure and URLs

- **Code** available at : `https://github.com/vanshs1/AutoHood3D/`
- Dataset of SFT LLM prompts (in 06_LLM_Generation/Final_consolidated_prompts.jsonl): `https://github.com/vanshs1/AutoHood3D/`
- Dataset of hood base shells/skins: `https://doi.org/10.7910/DVN/9268BB`
- Dataset of 4k hoods
  - STLs: `https://doi.org/10.7910/DVN/HEILMB`
  - Simulation Data (raw): `https://doi.org/10.7910/DVN/VCKOK5`
  - Processed for ML task: `https://doi.org/10.7910/DVN/6OAFF8`
  - Processed for ML task (Graphs): `https://doi.org/10.7910/DVN/WODNWY`
  - Test ML workflow**: `https://doi.org/10.7910/DVN/FSYRJA`
- Dataset of 12k hoods
  - STLs: `https://doi.org/10.7910/DVN/ZOVXLI`
- Dataset of 12k hoods (randomized)
  - STLs: `https://doi.org/10.7910/DVN/OJXIS1`

The updated URLs for the datasets, particularly the simulation data for the 12k case due to its large size, and the corresponding Croissant metadata for each dataset are shared on the GitHub repository. A compact ML test set** of 100 preprocessed cases is included for the rapid validation of training workflows. From these, users can easily sample smaller subsets (e.g., 12 or 32 batch sizes) to perform local testing of the ML pipelines.

A.2 Contents

The dataset contents are as follows:

- 10000+ randomized hoods cut from 108 unique base hood shells.
- 12000+ clustered hoods cut from 108 unique base hood shells.
- An additional 4500+ clustered hoods.
- JSON files for 2500+ prompts.
- 108 base shells for making new geometries.
- 1750 different curve cut-out files.

In Figure 4, the naming convention of a standard clustered geometry is shown. The base shell geo_010 refers to the standard outer shell from which the cutouts are carved. The “clusterID" 5 is assigned based on the strata of the perimeter and the area of the cuts. Further details on clustering are provided in A.3. All geometries are symmetrical, with “crvCount" referring to the number of cutouts on either side of the hood. “Curve ID" 0290_0694_0710_0627 functions to parameterize the geometry. The center distance “cd _0.030" refers to the minimum distance between the cutouts across the mirror plane, and the “md_0.015" refers to the minimum distance between the center of two cutouts on either side of the hood.

A.3 Clustering Cutouts

Figure 5 delineates nine clusters of engineered cutouts stratified by perimeter and area. Cluster 5 comprises the most elaborate shapes, with a mean perimeter of approximately 0.84 and mean area around 0.045. In contrast, clusters 2 and 7 represent the most compact designs, exhibiting mean perimeters between 0.43 and 0.49 and mean areas near 0.015. In contrast, several intermediate clusters (e.g., Clusters 1, 3, and 4) exhibit partial overlap, indicating gradual transitions in perimeter

File Format	Description
Hood geometries	
geo_***_clusterID_x_...stl	Surface mesh of the hood geometry from hood of base shell ***, cluster x (see Fig.4)
geo_***_...stl	Surface mesh of the hood geometry from hood of base shell *** with random cutouts (no cluster sampling) (see Fig.4)
Results and other data	
solid (or fluid)_...vtp	Raw CFD and FEA data, contains field values for the physical quantities. VTP is a poly-data format derived from VTK data type.
batch_**.pt	Torch dataset format containing fields for ML training.
x.json	JSON file containing LLM prompts or structured geometry list for creating ML training dataset.

Table 3: Summary of the dataset contents

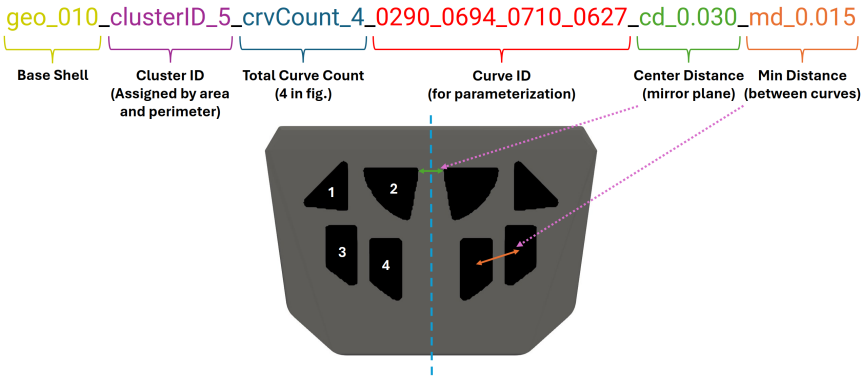

Figure 4: Example hood showing the physical definitions of the geometry naming convention

and area metrics and demonstrating that certain mid-range shapes share similar feature compositions. This overlap underscores the continuous nature of our design manifold and suggests opportunities for interpolation between cluster centroids when generating new variants. This calibrated partitioning of cutout topologies provides the dataset with a comprehensive representation of feature scales and intricacies, supporting systematic investigation of geometry-driven variations in fluid–structure response.

A.4 LLM Prompt

We designed a structured meta-prompt to guide Gemma3's vision-LLM into producing paired text–point-cloud training examples. For each hood image and the corresponding variation specification, the prompt instructs the model to generate: (1) a User Input section that formalizes the design request from the base description and variation details; (2) a Chain-of-thought section that explicitly reasons through curve counts, spacing, symmetry, and surface orientation; and (3) a Solution placeholder for the resulting point-cloud output. By enforcing this three-part schema, we automatically extract self-documented examples, complete with transparent intermediate reasoning, suitable for supervised fine-tuning of downstream LLMs on CAD text-to-geometry tasks. The code along with necessary details is shared online and the prompt is shown below.

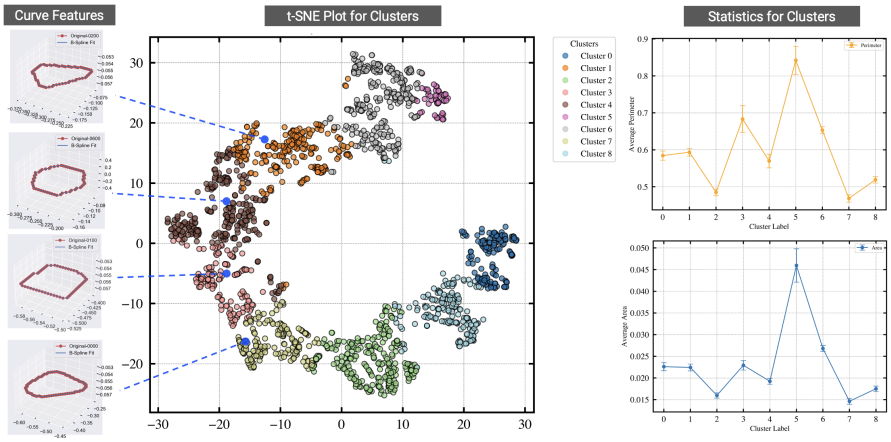

Figure 5: Clustering of extracted cutout curves. Left: representative boundary-curve features (original and B-spline fits) for four sample clusters. Center: t-SNE projection of all curves, colored by cluster label (0–8), illustrating distinct groupings based on perimeter and area metrics. Right: mean perimeter (top) and area (bottom) with error bars for each cluster, highlighting the geometric diversity captured across the nine classes.

Meta Prompt

You are a CAD-focused Vision-LLM. Analyze the image provided along with these guidelines:
Description of the base geometry without cuts: description_template
Variation Details: variation_details
Produce exactly three labeled sections—no additional commentary:
1. **User Input** Formulate the user request using the base description and variation details.
2. **Chain of Thought** Show internal reasoning step by step: - Start with: "Let me think through the requirements..." - Parse counts, distances, symmetry from the Variation Details. - Identify inner vs. outer face features from the base description. - Plan point-cloud density and cut placement.
3. **Solution** Provide a placeholder for the point-cloud output: "Solution: point_cloud"

B Hyperparameter Tuning

We adopt a unified set of hyperparameters that yielded stable performance across all architectures, with the specific parameters of each model (e.g., encoder-decoder configurations, hidden-channel width) tuned within this framework (see Table 5). All models are trained using the AdamW [50] optimizer, paired with a ReduceLROnPlateau scheduler configured as detailed in Table 4. For PointGNNConv, we reduced the base learning rate to 5×10^{-5} to prevent training instabilities observed in early trials, and the step patience for GraphSAGE is set to 9^* . Although the training pipeline supports distributed data-parallel (DDP) execution over the full 16000+ design ensemble, the results presented here are based on a reduced subset of 4500 hood geometries trained using a single H100 GPU.

Weight Decay	Step Factor	Min LR	Step Patience	Epochs	LR Warmup
1.00×10^{-6}	0.1	1.00×10^{-7}	7^*	750	Linear, 1.5%

Table 4: Optimizer and scheduler hyperparameters.

C Runtime Analysis

Table 6 summarizes the inference times per sample and the parameter counts for each architecture, highlighting their relative efficiency profiles. MLP and PointNet achieve sub-10 ms inference with strong in-distribution accuracy (see Table 1), ideal for high-throughput prototyping. Graph U-Net and

Parameter	PointGNNCon	GraphUNet	GraphSAGE	PointNet	MLP
encoder	[10,128,128,64]	[10,128,128,64]	[10,128,128,64]	[10,256,128,64]	[10,256,128,64]
decoder	[64,128,128,7]	[64,128,128,7]	[64,128,128,7]	[64,128,256,7]	[64,128,256,7]
nb_hidden_layers	4	–	4	–	4
size_hidden_layers	64	64	64	–	256
base_nb	–	–	–	8	–
layer	–	SAGE	–	–	–
pool	–	random	–	–	–
nb_scale	–	5	–	–	–
pool_ratio	–	[0.5,0.5,0.5,0.5]	–	–	–
list_r	–	[0.05,0.2,0.5,1,10]	–	–	–
batch_size	128	128	128	128	128
nb_epochs	750	352	750	750	750
lr	1e-4	1e-4	1e-4	1e-4	1e-4
max_neighbors	4	4	4	–	–
r	0.05	0.05	0.05	–	–

Table 5: Hyperparameter settings for each model.

Model	Parameter Count	Time per Epoch (s)	Time per Inference (ms)
MLP	319623	96 ± 4	9.66
PointNet	154463	120 ± 3	11.14
GraphSAGE	93703	147 ± 3	9.74
Graph U-Net	3550535	1520 ± 35	248.86
PointGNNConv	168598	366 ± 4	34.34

Table 6: Comparison of model sizes and runtimes. Inference time averaged for 200 samples.

PointGNNConv, while costing 200–250 ms per inference, provide higher OOD fidelity (see Table 2) using advanced message passing. GraphSAGE offers a compromise with 11 ms latency and moderate generalization.

D Graph Construction with Max-Neighbor Tuning

In the graph model setup, each node in the mesh is connected to all neighbors within a fixed radius, defining the local geometric context, where “max_neighbors" (mNN) caps the number of edges per node. By building the full mesh graph once (so no contour detail is lost to sub-sampling mesh points or repeated graph construction) and keeping the radius constant, we promote “max_neighbors" to the only complexity control. To measure its effect on training loss and per-epoch runtime, we train GraphSAGE (using same settings as mentioned above except the eopchs set to 150) on the complete mesh, preserving every cutout, and systematically vary only max_neighbors (testing 4, 8, 16, and 32), thus identifying the optimal neighborhood size that balances accuracy and speed. Here, the entire dataset is reloaded at each epoch, maintaining a constant memory footprint. This optimal setting captures critical geometric detail without incurring unnecessary computational cost.

mNN	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)	Time per Epoch	Dataset Size (GB)
4	4.65	1.31	2.00	4.13	14.56	3.78	6.56	147 ± 2	201.1
8	3.10	1.19	1.97	5.15	9.95	1.76	3.09	345 ± 7	290.4
16	2.93	1.07	1.71	4.30	10.49	4.28	8.53	443 ± 15	469.2
32*	1.91	0.98	1.40	3.35	8.10	1.51	2.83	2361 ± 245	826.6

Table 7: mNN Test: Mean squared error using the ID testing dataset for the different normalized fields using different mNN settings. *For mNN=32 setting, the model results are reported at 130 epochs due to longer per-epoch runtime.

Table 7 demonstrates that increasing the GraphSAGE neighborhood size (mNN) steadily reduces MSE across all fields but at the expense of both runtime and data-storage footprint. With mNN=4, the model trains in only 147 s per epoch on a 201 GB dataset, producing acceptable ID errors. Doubling to 8 neighbors halves certain displacement errors (e.g. D_y) but more than doubles per-epoch time and dataset size. By mNN=32, further accuracy gains are marginal while per-epoch time reaches ~ 2361 s and the storage exceeds 800 GB, this configuration was stopped at 130 epochs due to intractability.

Thus, mNN=4 delivers the most favorable speed–accuracy–size balance in this case, especially for larger graph models where high neighbor counts become prohibitively slow and memory intensive.

E Analysis of MLP Performance

Based on results for in-distribution tests in Sect. 3, MLP model tends to show better performance on multiple output variables (such as U_x, p) compared to more complex graph models. In this section, our aim is to explore the underlying reasons for these results, which could potentially be attributed to overfitting. Another possible explanation is that "MLPs can incorporate geometric information, given their encoder's width being four times that of any other model." While it is true that the MLP's hidden layer width is four times the dimensionality of its encoder output—in our case, a 64 to 256 configuration—this does not necessarily imply that the MLP is aggregating information from neighboring points. Instead, each vertex's feature vector (comprising its coordinates, normals, and distance field) is processed independently through fully connected layers, without any explicit pooling or message passing over spatial neighbors. Thus, MLP's unexpectedly strong in-distribution performance cannot be directly attributed to local geometric aggregation in the same way as with graph models. To further investigate this, we performed the following analysis:

Hypothesis	**Description**	**Result**
A. **Non-homophily**	If adjacent nodes differ greatly, graph aggregation may harm performance.	**Rejected**: high homophily found (see Table 9), so GNNs should benefit from local pooling.
B. **Sensitivity to neighbor count (k)**	GraphSAGE performance depends on neighborhood size.	**Confirmed**: larger k reduces prediction error but at steep compute and I/O cost.
C. **MLP Capacity & Overfitting**	The wide MLP memorizes training-set patterns, inflating in-distribution scores.	**Supported**: ablation shows ID error rises with hidden-dim reduction while OOD error improves (Table 11).

Table 8: Summary of hypotheses, their motivations, and empirical findings.

E.1 Homophily Test

The hood surface often exhibits cutout edges and curvature changes that could cause non-homophily. In such cases, neighborhood pooling of GNNs can oversmooth fine features—whereas an MLP's point-wise mapping avoids this [51], potentially explaining its ID test errors. Therefore, to measure the non-homophily, we computed numeric assortativity ($\bar{r}$) [52] and a standard edge-wise multi-dimensional cosine similarity [53] over the entire in-distribution test data/graphs:

Field	U_x	U_y	U_z	p	D_x	D_y	D_z	**Multi-dim Cosine**
$\bar{r}$	+0.9756	+0.9739	+0.9711	+0.9650	+0.9910	+0.9933	+0.9877	+0.9449
σ_r	0.0037	0.0106	0.0059	0.0058	0.0301	0.0154	0.0322	0.0109

Table 9: Mean assortativity ($\bar{r}$) and its standard deviation for each physical field with cosine similarity.

Interpretation: All fields exhibit strong homophily ($\bar{r} > 0.96$), so neighbor aggregation should be beneficial for GNNs—not detrimental—if effectively leveraged. This can be done by carefully selecting the "max neighbors" (mNN), as the quality of feature aggregation in GNNs is significantly influenced by this choice. This is the focus of Sec. D, where we explore the impact of mNN selection on GNN performance in more detail.

E.2 GraphSAGE mNN Sweep

In a separate experiment (see Table 7), we varied GraphSAGE's neighbor count (mNN = 4,8,16,32). Increasing the neighborhood size from 4 to 32 reduces ID MSE by up to 40-50%, but:
Compute Time: per-epoch time grows by an order of magnitude.

Data I/O: dataset size inflates proportionally to neighborhood features stored.
Higher mNN reduced in-distribution errors at the expense of increased computation time, illustrating the classic trade-off of neighborhood aggregation.

E.3 MLP Hidden-Size Ablation Study

Our ablation study compared MLPs with hidden-layer widths of 256, 128 and 64—keeping all other hyperparameters consistent with Table 5. The following is the complete set of results:

Model	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)
MLP 256	0.25	0.27	0.44	0.37	2.80	0.51	0.76
MLP 128	0.32	0.30	0.49	0.47	4.59	0.75	0.91
MLP 064	0.39	0.37	0.53	0.58	7.63	0.74	1.02

Table 10: ID Test: Mean squared error on the different normalized fields for all the MLP models.

Model	U_x ($\times 10^{-2}$)	U_y ($\times 10^{-2}$)	U_z ($\times 10^{-2}$)	p ($\times 10^{-2}$)	D_x ($\times 10^{-2}$)	D_y ($\times 10^{-2}$)	D_z ($\times 10^{-2}$)
MLP 256	1.49	1.89	2.97	4.96	181.02	5.87	19.81
MLP 128	1.58	1.60	2.68	4.65	175.28	7.58	22.91
MLP 064	2.14	1.09	1.59	3.38	124.11	8.50	20.55

Table 11: OOD Test: Mean squared error on the different normalized fields for all the MLP models.

ID Trend: error increases steeply as hidden dimension shrinks, confirming that the 256-dim model leverages high capacity to fit training data.
OOD Trend: the smaller MLPs generalize marginally better, indicating the largest network is more susceptible to overfitting the data.
Together, these studies suggest (Table 8) that the advantage of MLPs on ID tests is not directly due to hidden neighbor aggregation; rather, their wider layers facilitate point-wise memorization of global patterns. On the other hand, graph-based models take advantage of the existing homophily within the data, but they must strike a careful balance between neighborhood size and the risk of oversmoothing, as well as the associated computational overhead. As such, the choice of model for practical surrogate design will ultimately depend on the specific application: MLPs are better suited for high-throughput, in-distribution tasks, while GNNs—when tuned for maximum neighborhood size (mNN)—are more effective for robust generalization, assuming sufficient computational resources are available.