Car Drag Coefficient Prediction From 3D Point Clouds Using A Slice-based Surrogate Model

Utkarsh Singh[1*], Adarsh Roy[2] and Absaar Ali[3]

[1*]Department of Mechanical Engineering, Delhi Technological University, Shahbad Daulatpur, Delhi, 110042, India.
[2]Department of Mathematics, Indian Institute of Technology, Hauz Khas, Delhi, 110016, India.
[3]Department of Computer Science, Delhi Technological University, Shahbad Daulatpur, Delhi, 110042, India.

*Corresponding author(s). E-mail(s): utkarshsingh_me21b16_52@dtu.ac.in;
Contributing authors: adarsh.roy@iitdalumni.com; absaarali_co20b2_24@dtu.ac.in;

Abstract

The automotive industry's pursuit of enhanced fuel economy and performance necessitates efficient aerodynamic design. However, traditional evaluation methods like Computational Fluid Dynamics (CFD) and wind tunnel testing are resource-intensive, hindering rapid iteration in early design stages. Machine learning-based surrogate models offer a promising alternative, yet many existing approaches suffer from high computational complexity, limited interpretability, or insufficient accuracy for detailed geometric inputs. This paper introduces a novel, lightweight surrogate model for aerodynamic drag coefficient ($\boldsymbol{C_d}$) prediction based on a sequential, slice-wise processing of 3D vehicle geometry. Inspired by medical imaging, 3D point clouds of vehicles are decomposed into an ordered sequence of 2D cross-sectional slices along the streamwise axis. Each slice is encoded by a lightweight PointNet2D module, and the sequence of slice embeddings is processed by a bi-directional LSTM to capture longitudinal geometric evolution. The model, trained and evaluated on the DrivAerNet++ dataset, achieves a high coefficient of determination ($\boldsymbol{R^2 > 0.952}$) and low mean absolute error ($\boldsymbol{MAE \approx 0.0061}$) in $\boldsymbol{C_d}$ prediction. With an inference time of approximately 0.025 seconds per sample on a consumer-grade GPU, our approach provides fast, accurate, and interpretable aerodynamic feedback, facilitating more agile and informed automotive design exploration.

Keywords: Aerodynamic Drag, Surrogate Modeling, Machine Learning, Point Clouds, Automotive Design, Deep Learning, Sequential Models

1 Introduction

Aerodynamic efficiency is paramount in the automotive industry, directly impacting fuel economy, emissions, vehicle stability, and the range of electric vehicles (EVs). Reducing aerodynamic drag, quantified by the drag coefficient (C_d), is a primary design goal. However, conventional evaluation methods, namely Computational Fluid Dynamics (CFD) and wind tunnel testing, present significant bottlenecks. CFD simulations, while detailed, are computationally expensive and time-consuming, with typical runs taking hours to days and requiring substantial high-performance computing (HPC) resources [1, 2]. Wind tunnel tests, though crucial for validation, involve costly facility operation and lengthy model fabrication times [3], limiting their use in early, iterative design phases. These constraints hinder rapid exploration of the design space.

To overcome these limitations, machine learning (ML)-based surrogate models have emerged as a promising alternative for rapid aerodynamic prediction [4]. These models learn the complex mapping from vehicle geometry to aerodynamic properties from data generated by high-fidelity simulations. Once trained, they can predict C_d in seconds or milliseconds. However, existing ML surrogates face several challenges. Voxel-based methods, using 3D Convolutional Neural Networks (CNNs), suffer from resolution bottlenecks and can lose fine geometric details [5]. Projection-based methods, which convert 3D shapes into 2D images for 2D CNNs [6], can suffer from information loss due to occlusions or choice of viewpoints and may lack physical interpretability. Point cloud-based methods, such as PointNet [7] and its variants [8, 9], operate directly on 3D surface points but often treat points permutation-invariantly, potentially missing crucial directional cues inherent in aerodynamic flow. More recent graph neural networks [10] and transformer-based architectures [11, 12], while powerful, can be computationally intensive and complex. For instance, TripNet [13] uses triplane representations and achieves high accuracy but still involves sophisticated geometric processing. Many of these models, particularly complex deep learning architectures, act as "black boxes," offering limited insight into how specific geometric features influence aerodynamic performance.

This paper proposes a novel lightweight, sequential, and interpretable approach for C_d prediction. Our core idea is to represent the 3D vehicle geometry as an ordered sequence of 2D cross-sectional slices along the primary (streamwise) direction of airflow, analogous to how MRI or CT scans represent 3D anatomical structures. This structured representation explicitly captures the front-to-rear evolution of the vehicle's shape, which is fundamental to its aerodynamic behavior. Each 2D slice, represented as a point set, is processed by our lightweight **PointNet2D** module (our 2D adaptation of PointNet) to extract local geometric features. The sequence of these per-slice feature embeddings is then fed into a **bi-directional Long Short-Term Memory**

(LSTM) network, which models the dependencies and progression of shape features along the vehicle's length. Finally, a Multi-Layer Perceptron (MLP) regresses the C_d from the LSTM's aggregated representation. This approach offers several advantages:

- **Efficiency:** By processing 2D slices, it avoids the high computational cost of full 3D convolutions or global attention mechanisms on large point clouds.
- **Interpretability:** The sequential nature allows for potential attribution of drag contributions to specific longitudinal sections of the vehicle.
- **Geometric Intuition:** It directly models the flow-aware, front-to-rear progression of vehicle shape.

We validate our model using the large-scale DrivAerNet++ dataset [14], which provides high-fidelity CFD-computed C_d values for thousands of parametric car models. Our initial results demonstrate competitive accuracy with state-of-the-art methods, but with significantly reduced computational complexity and enhanced potential for interpretability.

The remainder of this paper is organized as follows: Section 2 details the dataset, data preprocessing techniques, and the proposed model architecture. Section 3 presents the experimental results, including performance comparisons and training dynamics. Section 4 discusses the implications of these results, limitations, and comparisons. Finally, Section 5 concludes the paper and outlines future research directions.

2 Methods

This section details the dataset, preprocessing steps, the architecture of our proposed slice-based sequential model, and the training procedure.

2.1 Dataset

We utilize the DrivAerNet++ dataset [14], a large-scale, multimodal car aerodynamics dataset. It contains over 8,000 parametric car models, spanning various body styles (fastback, notchback, estateback, SUV) and configurations (e.g., with/without detailed underbodies, rotating wheels). For each model, the dataset provides a 3D mesh, a 100k-point cloud representation, and the ground truth drag coefficient (C_d) computed using Reynolds-Averaged Navier-Stokes (RANS) $k-\omega$ SST CFD simulations. We use the point cloud representation provided via the PaddleScience platform [15]. After filtering for complete data, our working dataset comprises 7,713 unique car geometries. We adhere to the official dataset split: 5,398 samples for training, 1,157 for validation, and 1,158 for testing. The point clouds are consistently oriented with the X-axis along the streamwise direction, Y-axis laterally, and Z-axis vertically.

2.2 Data Preprocessing

The core of our preprocessing pipeline is the conversion of each 3D car point cloud into an ordered sequence of 2D cross-sectional slices.

2.2.1 Cross-Sectional Slicing

Each 3D point cloud (typically ∼100,000 points) is sliced along the primary flow direction (X-axis).

1. **Number of Slices (S):** We chose $S = 80$ slices. This value was empirically found to provide a good balance between capturing sufficient geometric detail along the car's length and maintaining a manageable sequence length for the LSTM. Too few slices would blur important local features, while too many would increase computational cost and redundancy.
2. **Binning:** For each car, the range of X-coordinates (x_{min}, x_{max}) is determined. This range is then divided into S equal bins. The width of each bin w is $(x_{max} - x_{min})/S$.
3. **Projection to YZ-Plane:** All points within the i-th bin (i.e., $x \in [x_{min} + (i-1)w, x_{min} + iw)$) are projected onto the YZ-plane by discarding their X-coordinate. This results in a 2D point set representing the cross-sectional profile of the car at that longitudinal station.

This process yields a sequence of $S = 80$ slices, each represented by a variable number of 2D points (y, z). Figure 1 illustrates this slicing concept.

2.2.2 Padding and Masking

The number of points in each 2D slice varies. To create fixed-size tensors for batch processing, we pad each slice.

- We determined the maximum number of points observed in any single slice across the entire dataset, M_{max} (found to be 6,500).
- Each 2D slice is zero-padded to have M_{max} points. Thus, each slice becomes a tensor of shape $(M_{max}, 2)$.
- A binary mask tensor of shape (S, M_{max}) is also created to distinguish real points from padded points, although our PointNet2D's max-pooling is robust to zero-padding for non-contributing points if features are non-negative or handled appropriately.

The final input representation for each car is a tensor of shape $(S, M_{max}, 2)$, i.e., $(80, 6500, 2)$.

2.3 Model Architecture

Our proposed model consists of three main components: a slice-level feature extractor (PointNet2D), a sequence model (Bi-LSTM), and a regression head (MLP). The overall architecture is depicted in Figure 3.

2.3.1 Slice-Level Feature Extraction: PointNet2D

Each 2D slice (a set of M_{max} points in $\mathbb{R}^2$) is processed independently by our PointNet2D module. This module is a simplified adaptation of the original PointNet [7] tailored for 2D point sets and designed to be lightweight. As shown in Figure 4, the PointNet2D module consists of a series of shared 1D convolutional layers (effectively acting as MLPs applied to each point) followed by a global max-pooling operation.

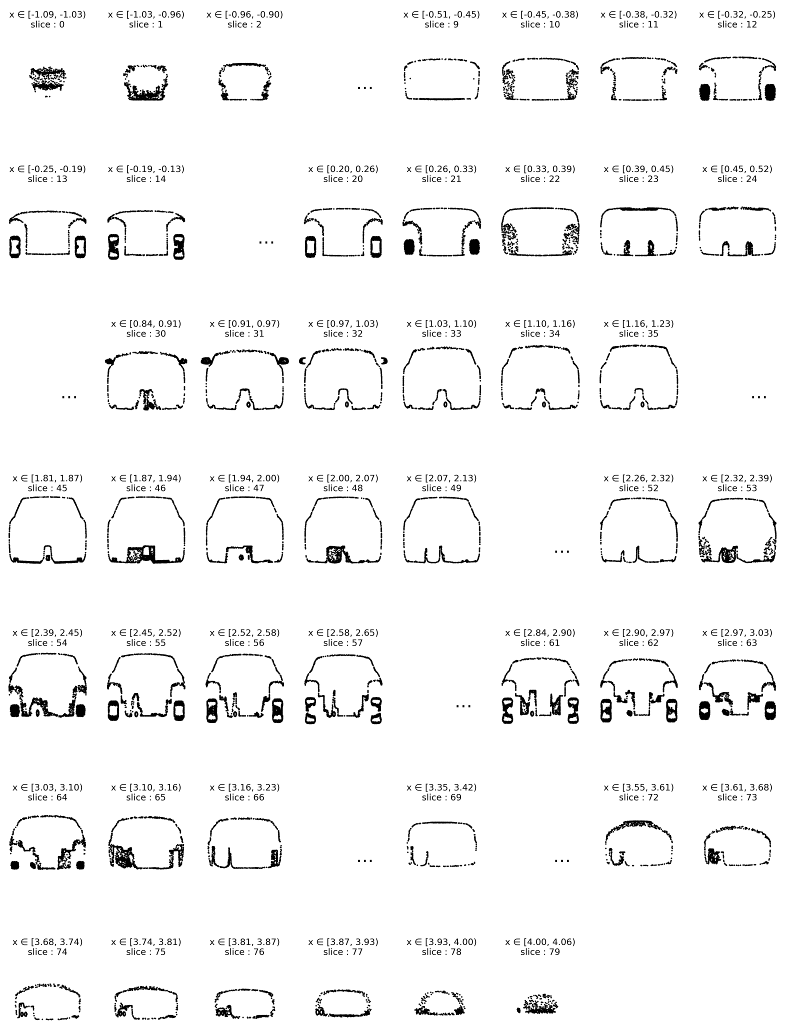

x ∈ [-1.09, -1.03) slice : 0
x ∈ [-1.03, -0.96) slice : 1
x ∈ [-0.96, -0.90) slice : 2
...
x ∈ [-0.51, -0.45) slice : 9
x ∈ [-0.45, -0.38) slice : 10
x ∈ [-0.38, -0.32) slice : 11
x ∈ [-0.32, -0.25) slice : 12
x ∈ [-0.25, -0.19) slice : 13
x ∈ [-0.19, -0.13) slice : 14
...
x ∈ [0.20, 0.26) slice : 20
x ∈ [0.26, 0.33) slice : 21
x ∈ [0.33, 0.39) slice : 22
x ∈ [0.39, 0.45) slice : 23
x ∈ [0.45, 0.52) slice : 24
...
x ∈ [0.84, 0.91) slice : 30
x ∈ [0.91, 0.97) slice : 31
x ∈ [0.97, 1.03) slice : 32
x ∈ [1.03, 1.10) slice : 33
x ∈ [1.10, 1.16) slice : 34
x ∈ [1.16, 1.23) slice : 35
...
x ∈ [1.81, 1.87) slice : 45
x ∈ [1.87, 1.94) slice : 46
x ∈ [1.94, 2.00) slice : 47
x ∈ [2.00, 2.07) slice : 48
x ∈ [2.07, 2.13) slice : 49
...
x ∈ [2.26, 2.32) slice : 52
x ∈ [2.32, 2.39) slice : 53
x ∈ [2.39, 2.45) slice : 54
x ∈ [2.45, 2.52) slice : 55
x ∈ [2.52, 2.58) slice : 56
x ∈ [2.58, 2.65) slice : 57
...
x ∈ [2.84, 2.90) slice : 61
x ∈ [2.90, 2.97) slice : 62
x ∈ [2.97, 3.03) slice : 63
x ∈ [3.03, 3.10) slice : 64
x ∈ [3.10, 3.16) slice : 65
x ∈ [3.16, 3.23) slice : 66
...
x ∈ [3.35, 3.42) slice : 69
...
x ∈ [3.55, 3.61) slice : 72
x ∈ [3.61, 3.68) slice : 73
x ∈ [3.68, 3.74) slice : 74
x ∈ [3.74, 3.81) slice : 75
x ∈ [3.81, 3.87) slice : 76
x ∈ [3.87, 3.93) slice : 77
x ∈ [3.93, 4.00) slice : 78
x ∈ [4.00, 4.06) slice : 79

Fig. 1: Conceptual illustration of 80 streamwise (X-axis) cross-sectional slices extracted from a vehicle's point cloud, showing the progressive change in contour from front to rear.

- Input: A slice of shape $(M_{max}, 2)$.
- Layers: Three 1D convolutional layers with kernel size 1. Channel sizes are $2 \to 32 \to 64 \to d_e = 256$. Each convolution is followed by a ReLU activation function.
- Max-Pooling: A global max-pooling operation is applied across the M_{max} points dimension to obtain a single feature vector of dimension $d_e = 256$ for each slice. This ensures permutation invariance for points within a slice.

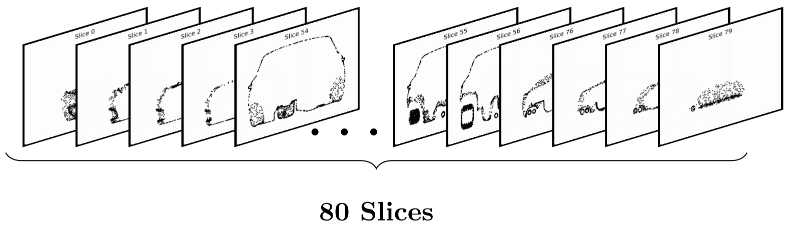


Fig. 2: Conceptual illustration of 80 streamwise (x-axis) slices extracted from a car's point cloud. Shown from an isometric side view, this depiction highlights how dense slicing captures detailed shape variation from nose to tail.

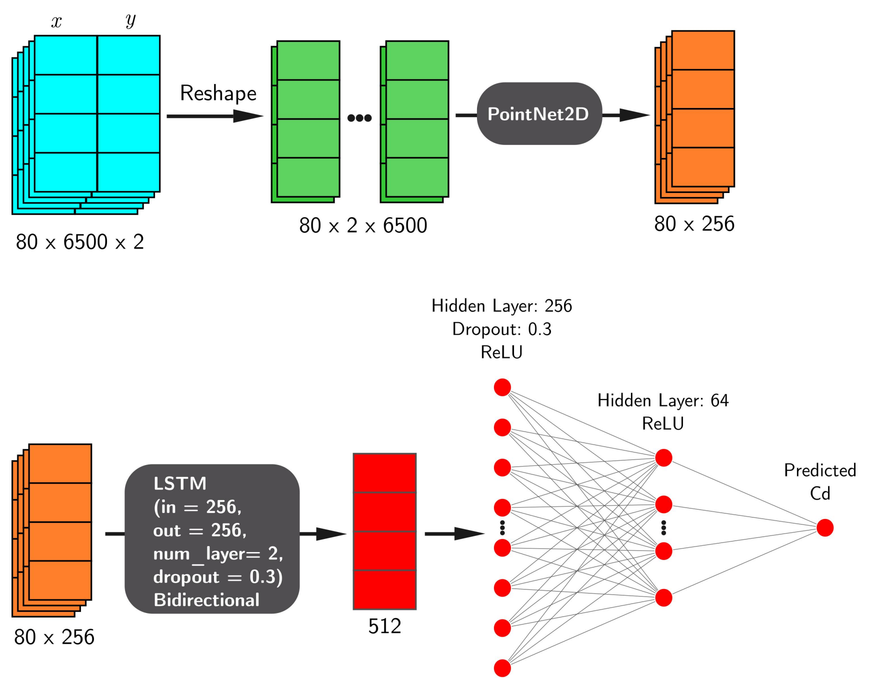


Fig. 3: Overall architecture of the proposed sequential slice-based drag prediction model. 3D point clouds are sliced; each slice is encoded by PointNet2D; the sequence of embeddings is processed by a Bi-LSTM; and an MLP regresses C_d.

The output for each car, after this stage, is a sequence of 80 embeddings, resulting in a tensor of shape $(80, 256)$.

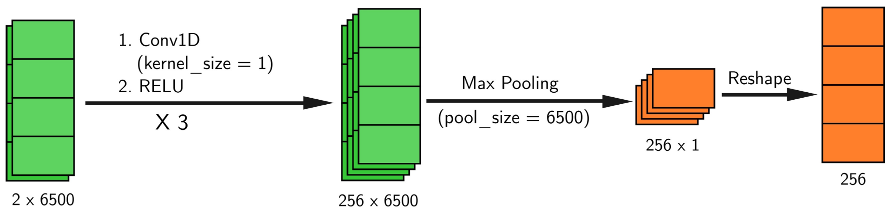


Fig. 4: Architecture of the PointNet2D module for encoding individual 2D slices. It uses shared 1D convolutions and max-pooling to generate a fixed-size embedding for each slice.

2.3.2 Sequence Modeling: Bi-Directional LSTM

The sequence of $S = 80$ slice embeddings (each of dimension $d_e = 256$) is processed by a bi-directional Long Short-Term Memory (Bi-LSTM) network. The Bi-LSTM captures dependencies and contextual information from both forward (front-to-rear) and backward (rear-to-front) directions of the slice sequence.

- Layers: 2 Bi-LSTM layers.
- Hidden Dimension: Each LSTM direction has a hidden state dimension of 256.
- Output: The outputs from the final hidden states of both directions are concatenated. For a 2-layer Bi-LSTM with hidden dimension $h = 256$, the concatenated output from the last time step's hidden states (specifically h_n from the forward LSTM of the last layer and h_0 from the backward LSTM of the last layer, or simply concatenating the final hidden states of the forward and backward passes of the last layer) results in a car-level embedding of dimension $2 \times 256 = 512$.

2.3.3 Regression Head: MLP

The 512-dimensional car embedding from the Bi-LSTM is fed into a Multi-Layer Perceptron (MLP) to regress the scalar C_d value.

- Layers: The MLP consists of three fully connected layers: $512 \to 256 \to 64 \to 1$.
- Activations: ReLU activations are used after the first two layers. A dropout layer with a rate of 0.3 is applied after the first ReLU for regularization.
- Output: A single scalar value representing the predicted C_d.

The total number of trainable parameters in the model is approximately 2.80 million.

2.4 Training Details

- **Loss Function:** We use the Smooth L1 Loss (Huber Loss), which is less sensitive to outliers than Mean Squared Error and provides smooth gradients.

$$\mathcal{L}(y, \hat{y}) = \begin{cases} 0.5(y - \hat{y})^2, & \text{if } |y - \hat{y}| < \beta \\ \beta(|y - \hat{y}| - 0.5\beta), & \text{otherwise} \end{cases}$$

  We use $\beta = 1.0$.

- **Optimizer:** Adam optimizer with an initial learning rate of 1×10^{-4}.
- **Batch Size:** A batch size of 4 was used due to GPU memory constraints with the large M_{max}.
- **Epochs:** The model was trained for 100 epochs, and the best model was selected based on the highest R^2 score on the validation set.
- **Hardware:** Training was performed on a single NVIDIA RTX 4060 GPU.

3 Results

This section presents the performance of our proposed slice-based sequential model. We first define the evaluation metrics, then provide quantitative comparisons with state-of-the-art methods, discuss computational efficiency, and finally analyze training dynamics and error distributions.

3.1 Evaluation Metrics

We use standard regression metrics to evaluate model performance:

- **Mean Squared Error (MSE):** $\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$. Measures the average squared difference between predicted and true values.
- **Mean Absolute Error (MAE):** $\frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$. Measures the average absolute difference, less sensitive to outliers than MSE.
- **Coefficient of Determination (R^2):** $1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$. Represents the proportion of variance in the dependent variable that is predictable from the independent variables. An R^2 of 1 indicates perfect prediction.
- **Maximum Absolute Error (MaxAE):** $\max_i |\hat{y}_i - y_i|$. Indicates the worst-case prediction error.

For these metrics, y_i is the true C_d, $\hat{y}_i$ is the predicted C_d, $\bar{y}$ is the mean of true C_d values, and N is the number of samples.

3.2 Quantitative Performance

We compare our model's performance on the DrivAerNet++ test set (1,158 samples) against several published surrogate models. Table 1 summarizes these results. Our PointNet2D+BiLSTM model achieves an R^2 of 0.9525 and an MAE of 6.111×10^{-3}.

Our model demonstrates performance comparable to the state-of-the-art TripNet on DrivAerNet++ in terms of R^2, while significantly outperforming earlier methods like RegDGCNN and PointNet on this larger, more diverse dataset. The low MAE indicates that, on average, our predictions are very close to the true C_d values.

3.3 Computational Efficiency

Table 2 compares the computational aspects of our model with other relevant methods. Our model is lightweight, with 2.80 million parameters, and achieves fast inference.

The inference latency of 0.025 seconds per sample on an NVIDIA RTX 4060 highlights the model's suitability for real-time feedback in interactive design scenarios.

Table 1: Quantitative comparison of drag–prediction models on the DrivAerNet++ test set. Bold entries indicate our model. † rows are DrivAerNet numbers included for context when DrivAerNet++ results were unavailable.

Model	Dataset subset	MSE (10^{-5})	MAE (10^{-3})	MaxAE (10^{-2})	R^2
PointNet2D+BiLSTM (Ours)	**DrivAerNet++ (1158)**	**6.60**	**6.111**	**4.50**	**0.9525**
TripNet [13]	DrivAerNet++ (1200)	9.10	7.17	7.70	0.957
RegDGCNN [14]	DrivAerNet++ (1200)	14.20	9.31	12.79	0.641
PointNet [7][a]	DrivAerNet++ (1200)	14.90	9.60	12.45	0.643
Original DrivAerNet results (context)					
TripNet [13]	DrivAerNet (600)†	2.60	4.03	1.27	0.972
FIGConvNet [16]	DrivAerNet (600)†	3.23	4.42	2.13	0.957
RegDGCNN [9]	DrivAerNet (600)†	8.01	6.91	8.80	0.901
PointNet++ [8][b]	DrivAerNet (600)†	7.81	6.76	3.46	0.896

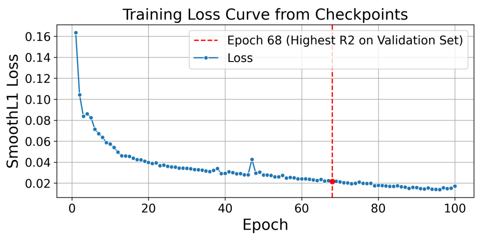
[a] Reported in [14]. [b] Reported in [13].

Table 2: Computational-cost comparison across representative surrogate models. GPU-memory and training-time figures are approximate and depend on implementation details and hardware. Our model was trained with a batch size of 4.

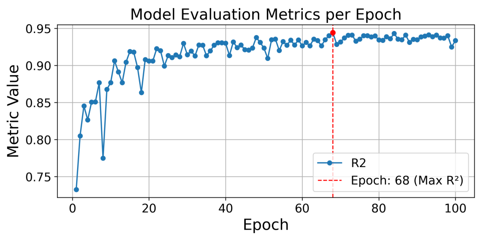
Model (reference)	Batch size	Inference latency (s/ sample)	GPU memory (GB)	Params (M)	Training time / HW
PointNet2D+BiLSTM (Ours)	4	**0.025**	5.9	**2.80**	**21 h / RTX 4060**
TripNet [13]	2	∼0.10	–	∼5.1	∼24 h / V100
RegDGCNN [9]	2	∼0.15	–	∼3.5	∼16 h / V100
DrivAer Transformer [10]	2	∼0.20	–	∼4.2	30–40 h / A100
Nissan 3D-CNN [4]	1	∼1.00	7.5	∼2.75	∼10 h / RTX 2080 Ti

3.4 Training Dynamics and Error Analysis

The model was trained for 100 epochs. Figure 5a shows the Smooth L1 training loss curve, indicating steady convergence. Figure 5b displays the validation R^2 score per epoch, with the best performance ($R^2 = 0.9525$) achieved at epoch 68, which was selected as the final model.



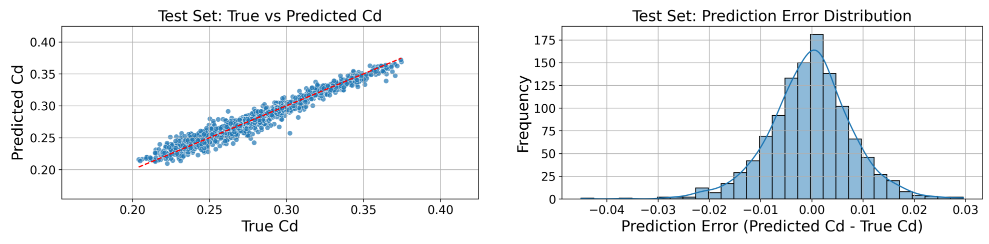
(a)



(b)

Fig. 5: Training dynamics: (a) Training loss curve. (b) Validation R^2 score over epochs. Best validation R^2 (0.9525) was at epoch 68.

To analyze the prediction quality on the test set, Figure 6 presents a scatter plot of predicted versus true C_d values and a histogram of prediction errors.



(a) (b)

Fig. 6: Test set performance analysis: (a) Scatter plot showing strong correlation between predicted and true C_d values. (b) Histogram of prediction errors, centered near zero with small spread.

The scatter plot (Figure 6a) shows a tight clustering of points around the $y = x$ line, indicating high agreement between predictions and ground truth. The error histogram (Figure 6b) is unimodal and centered close to zero, with the majority of errors falling within a narrow range (e.g., $\pm 0.015 C_d$). This demonstrates that the model generalizes well to unseen data and does not exhibit significant systemic bias. The MaxAE of 0.045 indicates that even the largest errors are within a reasonable range for early-stage design guidance.

4 Discussion

The results presented in Section 3 demonstrate that our proposed sequential slice-based model achieves high accuracy and efficiency for automotive aerodynamic drag prediction. The R^2 value of 0.9525 on the DrivAerNet++ test set signifies that the model captures over 95% of the variance in drag coefficients, performing comparably to more complex state-of-the-art methods like TripNet [13], while using a significantly simpler architecture and fewer parameters (2.80M for our model vs. ~5.1M for TripNet). The low MAE of approximately 0.0061 indicates that the average prediction error is very small, making it a reliable tool for distinguishing between design candidates.

The effectiveness of the slice-based approach can be attributed to its ability to explicitly model the geometric evolution of the vehicle's shape along the streamwise axis. Aerodynamic drag is highly sensitive to how the vehicle body first meets the airflow at the front, how the cross-sectional area and shape change along its length, and how the flow detaches at the rear. By processing an ordered sequence of 2D slices, the PointNet2D module learns salient features from each local cross-section, and the Bi-LSTM integrates this information sequentially, effectively learning the impact of these progressive shape changes on overall drag. This is a more direct way of capturing flow-relevant geometric information compared to methods that treat the point cloud as an unordered set or rely on 2D projections that might obscure crucial 3D features.

A key advantage of our model is its computational efficiency. With an inference time of ~0.025 seconds per sample on a consumer-grade GPU, it is substantially faster than traditional CFD and many complex deep learning surrogates (e.g., those based on 3D convolutions or full transformers on point clouds). This speed is critical for early-stage design, enabling engineers to rapidly evaluate numerous design variations, perform sensitivity analyses, and conduct shape optimization interactively. Furthermore, the relatively low parameter count contributes to faster training times and reduced risk of overfitting, especially when datasets might be limited for very specific vehicle types not yet covered by large public benchmarks.

Compared to other approaches, our method strikes a balance between performance and interpretability. While global 3D methods like PointNet or DGCNNs can be powerful, their permutation-invariant nature or complex graph structures can make it difficult to understand which parts of the geometry contribute most to the prediction. Our slice-based sequence allows, at least conceptually, for an investigation into how individual slices or segments of slices influence the final C_d prediction through analysis of LSTM activations or attention mechanisms if a transformer were used for sequence modeling. This potential for enhanced interpretability, by linking drag to longitudinal sections, can provide designers with more actionable feedback.

Despite its strengths, the proposed method has limitations.

1. **Inter-slice Information Loss:** While 80 slices provide good resolution, some fine 3D geometric details that do not significantly alter the 2D profile of any single slice but exist between slice planes or are inherently 3D in nature (e.g., complex underbody channels, small winglets with specific orientations not aligned with slices) might not be fully captured. The projection onto the YZ plane also means that 3D curvature within a slice's thickness is lost.
2. **Surface-only Scalar Prediction:** The current model predicts only the scalar C_d value and does not provide information about pressure or velocity fields on the vehicle surface or in the surrounding flow. Such field predictions are valuable for detailed aerodynamic analysis and are offered by some more complex surrogate models [13, 16].
3. **Absence of Explicit Physics Priors:** The model is purely data-driven. It does not inherently enforce physical laws like conservation of mass or momentum. This could lead to less robust predictions for out-of-distribution shapes not well-represented in the training data.
4. **Fixed Number of Slices:** The choice of $S = 80$ slices was empirical. An adaptive slicing strategy, perhaps denser in regions of high geometric change, might improve performance or efficiency, but would add complexity.

The broader impact of such a fast and accurate surrogate model is the potential to democratize aerodynamic analysis in the early stages of automotive design. It allows for more extensive design space exploration, leading to potentially more aerodynamically efficient vehicles developed in shorter timeframes and at lower costs.

5 Conclusion

This paper introduced a lightweight and efficient neural network architecture for predicting automotive aerodynamic drag coefficients (C_d) from 3D vehicle point clouds. Our novel approach transforms the 3D geometry into an ordered sequence of 2D cross-sectional slices along the streamwise axis. These slices are individually encoded using a PointNet2D module, and their sequential geometric evolution is captured by a bi-directional LSTM, with a final MLP regressing the C_d.

Trained and evaluated on the large-scale DrivAerNet++ dataset, our model achieved a coefficient of determination (R^2) of 0.9525 and a mean absolute error (MAE) of 6.111×10^{-3}. These results are competitive with more complex state-of-the-art surrogate models, demonstrating the efficacy of the slice-based sequential representation. A key advantage of our method is its computational efficiency, with an inference time of approximately 0.025 seconds per vehicle on a consumer-grade GPU and a modest parameter count of 2.80 million. This enables rapid aerodynamic feedback, facilitating extensive design iteration and optimization in the early conceptual phases of vehicle development. The inherent structure of the model, focusing on longitudinal geometric progression, also offers potential for enhanced interpretability compared to global "black-box" 3D models.

Future work will focus on several avenues. Enhancing slice encoding with more sophisticated 2D shape descriptors or exploring advanced sequence models like transformers could further improve accuracy. Extending the model to predict surface pressure distributions or even simplified flow fields would provide richer aerodynamic insights. Investigating adaptive slicing techniques and incorporating physics-informed neural network principles could enhance robustness and accuracy, particularly for out-of-distribution geometries. Ultimately, integrating such fast and interpretable surrogate models into interactive CAD tools holds the promise of significantly accelerating and improving aerodynamic design in the automotive industry.

Declarations

- **Funding**
  This research received no external funding.

- **Conflict of Interest**
  The authors declare that they have no conflict of interest.
- **Ethical Approval**
  This article does not contain any studies involving human participants or animals performed by any of the authors.
- **Informed Consent**
  Not applicable.
- **Data Availability**
  The dataset used in this study, DrivAerNet++, is publicly available via PaddleScience. Any additional data or code will be made available upon reasonable request.

References

[1] Report, I.: OEM CFD Workflows in Automotive (2025). Retrieved from internal engineering whitepapers (2025)

[2] Research, Z.E.: ANSS Sets CFD Simulation Record Using AMD GPUs. https://www.nasdaq.com/articles/anss-sets-cfd-simulation-record-using-amd-gpus-frontier-supercomputer (2024)

[3] Company, F.M.: Ford Invests $200M in new Wind Tunnel Facility. https://www.motorauthority.com/news/1108886_ford-invests-200m-in-new-wind-tunnel-facility (2017)

[4] Akasaka, K., Chen, F., Teraguchi, T.: Surrogate model development for prediction of car aerodynamics using machine learning. Nissan Technical Review **89**, 79–84 (2022). https://www.nissan-global.com/EN/TECHNICALREVIEW/PDF/NISSAN_TECHINICAL_REVIEW_89_En_ALL.pdf

[5] Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 922–928 (2015). https://doi.org/10.1109/IROS.2015.7353481 . IEEE

[6] Song, J., *et al.*: Data-driven car drag prediction with depth and normal renderings. Journal of Mechanical Design **146**(5), 051714 (2023) https://doi.org/10.1115/1.4068104

[7] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 652–660 (2017) https://doi.org/10.1109/CVPR.2017.16

[8] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems **30**, 5099–5108 (2017)

[9] Elrefaie, M., Dai, A., Ahmed, F.: Drivaernet: A parametric car dataset for data-driven aerodynamic design and prediction. arXiv preprint arXiv:2403.08055 (2024)

[10] He, J., Luo, X., Wang, Y.: Drivaer transformer: A high-precision and fast prediction method for vehicle aerodynamic drag coefficient based on the drivaernet++ dataset. arXiv preprint arXiv:2504.08217 (2025)

[11] Jiang, J., Li, G., Jiang, Y., Zhang, L., *et al.*: Transcfd: A transformer-based decoder for flow field prediction. Engineering Applications of Artificial Intelligence **123**, 106340 (2023) https://doi.org/10.1016/j.engappai.2023.106340

[12] Xiang, H., Ma, Y., Dai, Z., Wang, C., Zhang, B.: Aerodit: Diffusion transformers for reynolds-averaged navier-stokes simulations of airfoil flows. arXiv preprint arXiv:2412.17394 (2024). https://arxiv.org/abs/2412.17394

[13] Chen, Q., Elrefaie, M., Dai, A., Ahmed, F.: Tripnet: Learning large-scale high-fidelity 3d car aerodynamics with triplane networks. arXiv preprint arXiv:2503.17400 (2025). https://arxiv.org/abs/2503.17400

[14] Elrefaie, M., Morar, F., Dai, A., Ahmed, F.: Drivaernet++: A large-scale multimodal car dataset with computational fluid dynamics simulations and deep learning benchmarks. arXiv preprint arXiv:2406.09624 (2024). https://arxiv.org/abs/2406.09624

[15] PaddleScience Contributors: DrivAerNet++ Example – PaddleScience Documentation. https://paddlescience-docs.readthedocs.io/zh-cn/latest/zh/examples/drivaernetplusplus/ Accessed: 2025-05-12 (2025)

[16] Choy, C., Kamenev, A., Kossaifi, J., et al.: Factorized implicit global convolution for automotive computational fluid dynamics prediction. arXiv preprint arXiv:2502.04317 (2025). https://arxiv.org/abs/2502.04317