Ultra-Compact Geometry-Aware Transformers for Airfoil Polar Prediction

Avneh Singh Bhatia
avnehb@gmail.com

Abstract

We present **FoilForm**, a two-stage neural surrogate for predicting aerodynamic lift and drag coefficients (C_l, C_d) directly from airfoil contour geometry across angle-of-attack sweeps. The first stage is an ultra-compact autoregressive transformer (**4,470 parameters**) with a novel *pairwise outer-product attention* mechanism that captures second-order geometric feature interactions. The second stage is a lightweight convolutional correction network (**34,290 parameters**) that refines predictions using the full-resolution contour. Trained on 1,768 airfoils from a compiled dataset of 2,946 profiles at $\mathrm{Re} = 10^5$, the full pipeline achieves validation MAE of **0.040 on** C_l and **0.0081 on** C_d across 1,178 held-out airfoils, while running at **0.3 ms per airfoil** in batched CPU inference. Ablation studies demonstrate that pairwise interactions, minimal dropout (p=0.05), and 4-layer depth are critical design choices. We benchmark against NeuralFoil (xxxlarge) and show 9× lower C_l error and 19× lower L/D error on matched evaluation, while using 8.6× fewer parameters.

1 Introduction

Aerodynamic coefficient prediction is a fundamental task in aircraft design, wind turbine optimization, and unmanned aerial vehicle development. Traditional approaches—panel methods (XFOIL [1]), Reynolds-Averaged Navier–Stokes (RANS) solvers, and wind tunnel testing—produce accurate results but are computationally expensive, making large-scale design space exploration prohibitive.

Neural surrogate models offer orders-of-magnitude speedup for aerodynamic prediction [3]. Prior approaches typically use convolutional or fully connected architectures with hundreds of thousands of parameters, fixed-resolution geometry inputs, or separate models per operating condition. These designs sacrifice compactness, generalization, or physical interpretability.

We take a fundamentally different approach. Our contributions are:

1. **Geometry tokenization**: We decompose 501-point airfoil contours into 167 non-overlapping triplet patches, each encoded into an 8-dimensional embedding by a learned autoencoder, producing a variable-length sequence suitable for transformer processing.
2. **Pairwise outer-product attention**: We introduce a novel attention block that computes explicit second-order feature interactions via learned bilinear transforms on per-token outer products—analogous to pair energies in protein structure prediction [5]—enabling the model to capture multiplicative geometric relationships that standard attention misses.
3. **Autoregressive polar decoding**: The transformer autoregressively generates (C_l, C_d) across angle-of-attack steps using KV-cached decoding, naturally conditioning each prediction on prior aerodynamic states.

4. **Residual correction**: A lightweight Conv1D + MLP correction network operating on the full-resolution contour refines the tokenized transformer's output, recovering fine-grained geometric information lost during tokenization.
5. **Extreme compactness**: The entire pipeline—4,470 transformer parameters plus 34,290 correction parameters—achieves competitive accuracy in under 39K total parameters with sub-millisecond batched inference.

2 Related Work

Neural aerodynamic surrogates. Machine learning for airfoil prediction has been explored with Deep CNNs [6], CNNs on signed distance fields [7], graph-based CFD learning workflows supported by benchmark datasets such as AirfRANS [8], and physics-informed networks [9]. NeuralFoil [4] provides a pretrained general-purpose model but requires >100K parameters.

Transformers for physical systems. Transformers have been applied to fluid dynamics [10], weather prediction [11], and molecular property prediction [12]. We adapt the autoregressive paradigm specifically for sequential polar curve generation.

Second-order interactions. Outer-product feature interactions appear in EvoFormer [5] for protein structure and in factorization machines [13] for recommendation systems. We apply the same intuition in a compact aerodynamic setting, where each block uses only 192 attention parameters.

3 Data

3.1 Dataset

The training dataset is compiled from publicly available airfoil coordinate files and coefficient tables, merged into a unified format containing **2,946 airfoil profiles** [2]. Each airfoil consists of:

- A contour represented as 501 (x, y) stations, uniformly distributed from trailing edge around the profile and back.
- Polar observations $\{(\alpha_j, C_{l,j}, C_{d,j})\}$ at observed angles of attack, with missing entries marked as NaN on a shared AoA grid spanning approximately -4° to $+4^\circ$.

All experiments use Reynolds number $\mathrm{Re} = 10^5$, typical of small UAVs and low-speed wind tunnel conditions.

3.2 Train/Validation Split

Splits are performed at the *airfoil level* (not the observation level) with a fixed random seed (`seed=42`). The default training fraction of 60% yields:

- **Training**: 1,768 airfoils (all AoA steps for each airfoil)
- **Validation**: 1,178 airfoils

This ensures the model is evaluated on entirely unseen airfoil geometries, not merely unseen operating conditions of known shapes.

4 Model Architecture

The full FoilForm pipeline consists of three stages: (1) geometry and aerodynamic tokenizers, (2) a causal transformer with pairwise attention, and (3) a residual polar correction network. Figure 1 shows the end-to-end pipeline.

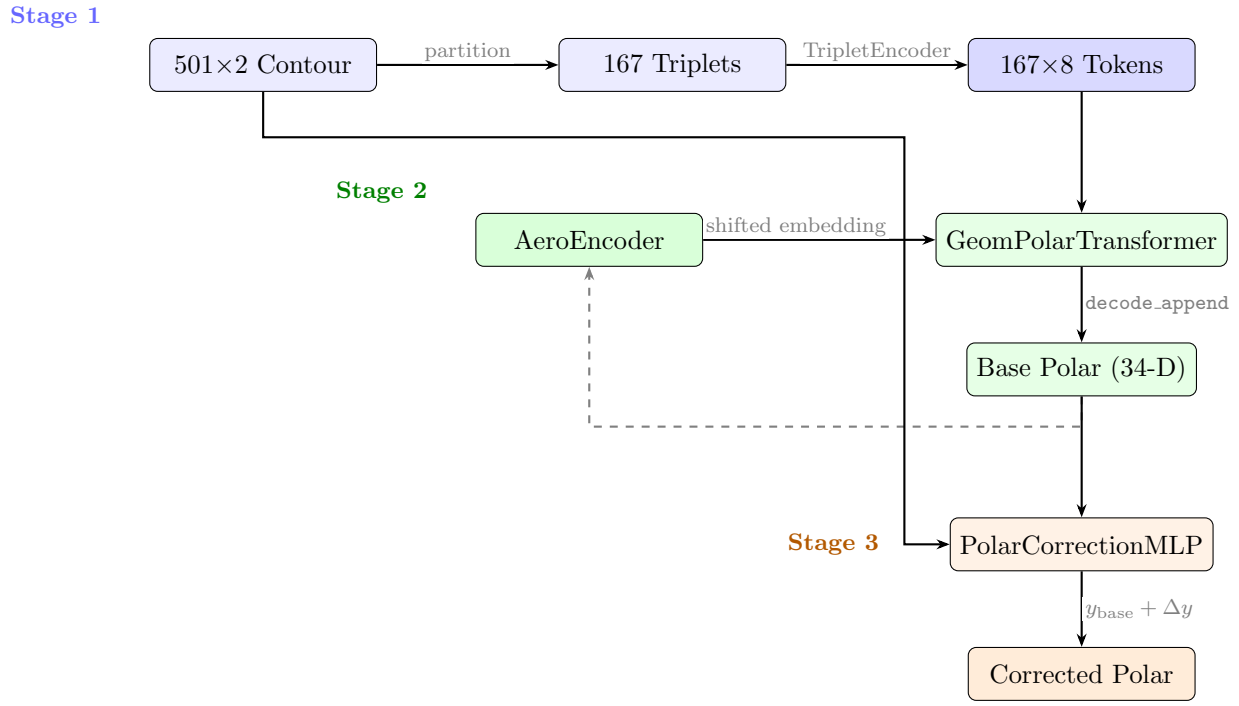

Figure 1: End-to-end FoilForm pipeline. Stage 1 tokenizes the airfoil contour into 167 geometry tokens. Stage 2 autoregressively predicts C_l, C_d at each AoA step via KV-cached decoding, with an autoregressive feedback loop (dashed). Stage 3 refines predictions using the full-resolution contour.

4.1 Stage 1: Geometry and Aerodynamic Tokenizers

4.1.1 Geometry Tokenizer

The airfoil contour $(x_1, y_1), \ldots, (x_{501}, y_{501})$ is partitioned into **167 non-overlapping triplets** of 3 consecutive stations, yielding a tensor $T \in \mathbb{R}^{167\times 6}$. Each triplet is independently encoded by a learned TripletEncoder:

$$z_i = \text{TripletEncoder}(x_{3i-2}, y_{3i-2}, x_{3i-1}, y_{3i-1}, x_{3i}, y_{3i}) \in \mathbb{R}^8 \tag{1}$$

The encoder is a 3-layer MLP with LayerNorm and GELU activations:

$$\text{Linear}(6 \to 64) \to \text{LN} \to \text{GELU} \to \text{Linear}(64 \to 64) \to \text{LN} \to \text{GELU} \to \text{Linear}(64 \to 8)$$

A symmetric TripletDecoder ($8 \to 64 \to 64 \to 6$, GELU, no LayerNorm) enables reconstruction. Both are trained jointly with MSE reconstruction loss plus an L2 embedding penalty ($\lambda = 0.01$) to regularize the latent space.

4.1.2 Aerodynamic Tokenizer

Each polar observation (α, C_l, C_d) is encoded using **Fourier features on angle of attack**:

$$\phi(\alpha) = \big[\sin(k\theta), \cos(k\theta)\big]_{k=1}^{4}, \quad \theta = \alpha \cdot \frac{\pi}{180}$$

This 8-dimensional periodic encoding is concatenated with (C_l, C_d) to form a 10-D input, then processed by the same MLP architecture as the geometry encoder:

$$e = \text{AeroEncoder}\big(\phi(\alpha) \,\|\, C_l \,\|\, C_d\big) \in \mathbb{R}^8$$

The Fourier encoding provides inductive bias for the periodic dependence of aerodynamic coefficients on angle of attack, enabling the model to learn smooth angular relationships without manual feature engineering.

4.2 Stage 2: GeomPolarTransformer

4.2.1 Input Fusion

At each sequence position i, the geometry token is concatenated with a *right-shifted* aerodynamic embedding and linearly projected:

$$x_i = \text{Dropout}\Big(\text{LayerNorm}\big(\text{Linear}_{16\to 8}([z_i \,\|\, e_{i-1}])\big)\Big) \tag{2}$$

where $e_0 = \mathbf{0}$ because the first position has no prior prediction. For later steps,

$$e_{i-1} = \text{Linear}_{3\to 8}([C_l^{(i-1)}, C_d^{(i-1)}, \alpha^{(i-1)}])$$

encodes the previous step's predicted coefficients together with the ground-truth AoA. This provides autoregressive conditioning: each position is informed by the aerodynamic state predicted at the prior step.

4.2.2 AttentionPairwiseBlock

Each transformer layer is a residual block with four branches. All branches use pre-LayerNorm and dropout (p=0.05):

1. Causal Self-Attention. Standard scaled dot-product attention with learned $W_Q, W_K, W_V \in \mathbb{R}^{8\times 8}$:

$$Q = \tanh(\text{LN}(x))W_Q, \quad K = \tanh(\text{LN}(x))W_K, \quad V = \tanh(\text{LN}(x))W_V \tag{3}$$

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}} + M_{\text{causal}}\right) V \tag{4}$$

where M_{causal} is an additive causal mask ($-\infty$ above the diagonal) and the tanh nonlinearity before projection provides bounded activations that stabilize training at this small model scale.

2. Pairwise Outer-Product Interaction (first). The distinctive component: at each sequence position, we compute a $d \times d$ outer product of the token's normalized feature vector with itself, then apply two learned bilinear transforms:

$$\hat{x} = \tanh(\mathrm{LN}(x)) \tag{5}$$

$$M_{ijk} = \hat{x}_{ij} \cdot \hat{x}_{ik} \quad \in \mathbb{R}^{B \times L \times d \times d} \tag{6}$$

$$T_1 = \mathrm{LeakyReLU}(W_{p1} M + B_{p1}) \tag{7}$$

$$T_2 = \mathrm{LeakyReLU}(W_{p2} T_1 + B_{p2}) \tag{8}$$

$$\mathrm{out} = \mathrm{mean}(T_2, \mathrm{dim} = -1) \quad \in \mathbb{R}^{B \times L \times d} \tag{9}$$

where $W_{p1}, W_{p2} \in \mathbb{R}^{d \times d}$ are learned weights and $B_{p1}, B_{p2} \in \mathbb{R}^{d \times d}$ are bias matrices initialized to zero. The mean-pooling across the last dimension of the bilinear output produces a residual update.

This mechanism captures *multiplicative feature interactions* that standard attention and MLPs cannot represent efficiently. In the context of airfoil geometry, these second-order interactions correspond to relationships between curvature, thickness, and camber features that jointly determine aerodynamic performance—analogous to pair energies in protein structure prediction [5].

3. Bottleneck MLP. A standard feedforward block with expansion factor 2:

$$\mathrm{MLP}(x) = W_2 \cdot \mathrm{LeakyReLU}(W_1 x + b_1) + b_2$$

with $W_1 \in \mathbb{R}^{16 \times 8}$ and $W_2 \in \mathbb{R}^{8 \times 16}$.

4. Pairwise Outer-Product Interaction (second). Same architecture as branch 2 with independent weights $(W_{p3}, B_{p3}, W_{p4}, B_{p4})$, providing a second round of multiplicative interaction after the MLP refinement.

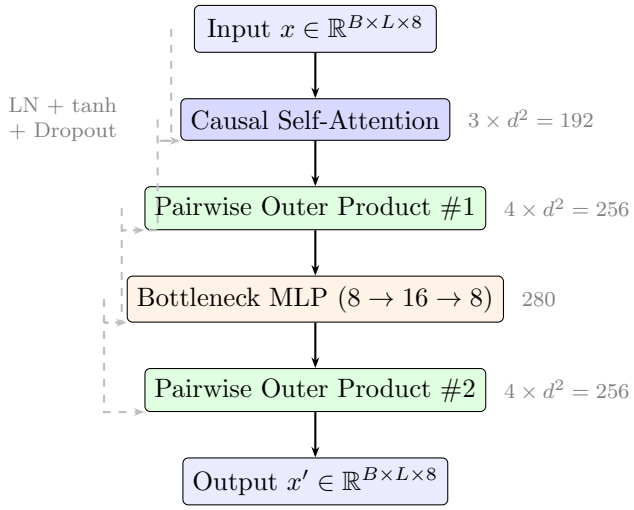

Figure 2: AttentionPairwiseBlock architecture. Each branch has a residual connection, pre-LayerNorm, and dropout. Parameter counts shown for d=8. Total per block: **1,048** parameters including 4 LayerNorms ($4 \times 16 = 64$).

4.2.3 Output Head

The last token's hidden state is projected through a three-stage linear head to produce (C_l, C_d):

$$h = x_L^{(\text{last layer})}, \quad h \leftarrow hW_8 + b_8, \quad h \leftarrow hW_{8\times 2}, \quad \hat{y} = hW_{2\times 2} + b_2 \tag{10}$$

The intermediate $8 \to 8$ projection with bias provides a nonlinear mixing layer before the rank-reducing projections to 2D output.

4.2.4 Autoregressive Decoding with KV Cache

During both training and inference, the model uses KV-cached autoregressive decoding, implemented by `decode_append`:

Algorithm 1: Autoregressive Polar Decoding

Input: Geometry tokens $Z \in \mathbb{R}^{B\times 167\times 8}$, AoA schedule $\alpha_1, \ldots, \alpha_T$

1. **Context pass:** Run all 167 tokens through full stack with causal mask; cache (K, V) per layer
2. $\hat{y}_1 \leftarrow \text{Head}(x_{167}^{(\text{last layer})})$ *// First (C_l, C_d) prediction*
3. **for** $t = 2, \ldots, T$ **do**
4. $\quad e_t \leftarrow \text{Linear}_{3\to 8}([\hat{C}_l^{(t-1)}, \hat{C}_d^{(t-1)}, \alpha_{t-1}])$ *// Condition embedding*
5. $\quad$ Run e_t through each block using cached (K, V); append new keys/values
6. $\quad \hat{y}_t \leftarrow \text{Head}(x_{\text{new}}^{(\text{last layer})})$
7. **end for**
8. **return** $[\hat{y}_1, \ldots, \hat{y}_T]$

Note that α_t is always ground-truth AoA (a user-specified operating condition), while $\hat{C}_l$ and $\hat{C}_d$ are the model's own predictions fed back autoregressively.

4.2.5 Parameter Breakdown

Table 1 provides a complete accounting of the transformer's 4,470 parameters.

Table 1: GeomPolarTransformer parameter breakdown (d=8, 4 layers).

Component	Details	Parameters
Per AttentionPairwiseBlock (×4 layers):		
Attention Q, K, V	$3 \times d^2$	192
Pairwise ×2 (W, B each)	$2 \times 4 \times d^2$	512
Bottleneck MLP	$16 \times d + 16 + d \times 16 + d$	280
LayerNorm ×4	$4 \times 2d$	64
Block total		**1,048**
All 4 blocks	$4 \times 1{,}048$	4,192
`tuple_to_embed`	Linear($3 \to 8$)	32
`in_proj`	Linear($16 \to 8$)	136
`norm_in`	LayerNorm(8)	16
Output head	$W_{8\times 8} + b_8 + W_{8\times 2} + W_{2\times 2} + b_2$	94
Grand total		**4,470**

4.3 Stage 3: PolarCorrectionMLP

The transformer operates on the 167-patch tokenized representation, which discards sub-triplet geometric detail. A lightweight residual correction network operates on the *full-resolution* 501-point contour to recover this lost information:

$$\Delta y = g(\text{geom}, y_{\text{base}}), \qquad y_{\text{final}} = y_{\text{base}} + \Delta y \tag{11}$$

4.3.1 Architecture

The correction network has two components:

Geometry Encoder (Conv1D). The raw contour $(B, 2, 501)$ is processed by two strided 1D convolutions with tanh activation and adaptive average pooling:

$$h_1 = \tanh\big(\text{Conv1d}(2 \to 16, k{=}11, s{=}5)(x)\big) \qquad \in \mathbb{R}^{B\times 16\times 99} \tag{12}$$

$$h_2 = \tanh\big(\text{Conv1d}(16 \to 32, k{=}5, s{=}3)(h_1)\big) \qquad \in \mathbb{R}^{B\times 32\times 32} \tag{13}$$

$$h_3 = \text{flatten}\big(\text{AdaptiveAvgPool1d}(4)(h_2)\big) \qquad \in \mathbb{R}^{B\times 128} \tag{14}$$

Correction Head (MLP). The 128-D geometry feature is concatenated with the transformer's 34-D base polar output (17 C_l values + 17 C_d values across the AoA grid):

$$h_4 = \tanh\big(\text{Linear}(162 \to 128)([\, h_3 \,\|\, y_{\text{base}}])\big) \tag{15}$$

$$h_5 = \tanh\big(\text{Linear}(128 \to 64)(h_4)\big) \tag{16}$$

$$\Delta y = \text{Linear}(64 \to 34)(h_5) \tag{17}$$

The output layer is initialized with small uniform weights $\mathcal{U}(-0.01, 0.01)$ and zero bias, so the network starts as an approximate identity mapping ($\Delta y \approx 0$) and learns the residual correction.

Table 2: PolarCorrectionMLP parameter breakdown.

Component	Details	Parameters
Conv1d layer 1	$2 \to 16$, k=11, s=5	368
Conv1d layer 2	$16 \to 32$, k=5, s=3	2,592
FC1	$162 \to 128$ + bias	20,864
FC2	$128 \to 64$ + bias	8,256
FC3	$64 \to 34$ + bias	2,210
Total		**34,290**

5 Training

5.1 Tokenizer Training

Both autoencoders (geometry and aerodynamic) are trained for 300 epochs with batch size 4,096 using AdamW (lr=10^{-3}, weight decay 10^{-4}) and cosine annealing. The loss is:

$$\mathcal{L}_{\text{tokenizer}} = \text{MSE}(\hat{x}, x) + \lambda \|z\|_2^2, \quad \lambda = 0.01 \tag{18}$$

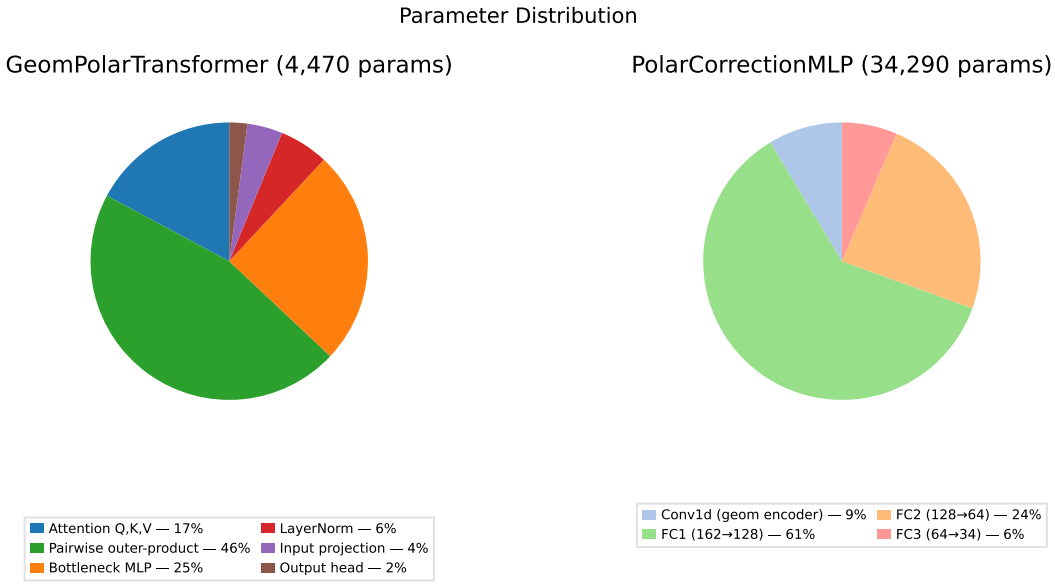

Figure 3: Parameter distribution across model components. Left: GeomPolarTransformer (4,470 parameters). Right: PolarCorrectionMLP (34,290 parameters).

5.2 Transformer Training

The GeomPolarTransformer is trained for 120 epochs with:

- Batch size: 64
- Optimizer: AdamW, lr $= 3 \times 10^{-4}$, weight decay 10^{-4}
- Schedule: Cosine annealing ($T_{\max} = 120$)
- Early stopping: patience 20, warmup 10, $\delta_{\min} = 10^{-5}$
- Loss: Masked weighted MSE on (C_l, C_d) with per-channel normalization

The loss uses per-channel scaling factors computed from the training set variance to balance the contribution of C_l (typical range -1 to $+2$) and C_d (typical range 0.005 to 0.05):

$$\mathcal{L}_{\text{tfm}} = \frac{1}{|\mathcal{V}|} \sum_{(i,t)\in\mathcal{V}} \left[w_{C_l}(C_{l,it} - \hat{C}_{l,it})^2 + w_{C_d}(C_{d,it} - \hat{C}_{d,it})^2 \right] \tag{19}$$

where $\mathcal{V}$ is the set of valid (non-NaN) observations.

5.3 Correction MLP Training

The correction network is trained for 200 epochs on the frozen transformer's output:

- Batch size: 256
- Optimizer: AdamW, lr $= 3 \times 10^{-3}$, weight decay 10^{-4}
- Schedule: Cosine annealing ($T_{\max} = 200$)
- Loss: MSE between corrected polar and ground truth

The higher learning rate (10× the transformer) is appropriate because the correction network starts near identity and needs to learn only small residuals.

6 Experiments

All ablation studies are orchestrated by a unified pipeline and logged to a structured manifest. Training defaults are held constant unless the specific variable is under study: 120 epochs, batch 64, AdamW (lr=3×10^{-4}, wd=10^{-4}), cosine annealing, dropout p=0.05, d=8, 4 layers, pairwise block, 60% train fraction.

6.1 Block Architecture Ablation

We compare five block variants, all at 4 layers with dropout 0.05:

Table 3: Block architecture ablation. All variants use 4 layers, d=8, dropout = 0.05.

Block Type	Params	Val MAE C_l	Val MAE C_d
Full Pairwise (attn + pw + MLP + pw)	4,470	**0.0747**	0.0089
Attention Only (no pw, no MLP)	1,110	0.0748	**0.0087**
Standard MLP (attn + MLP, no pw)	2,294	0.0904	0.0086
No Pairwise (attn + MLP)	2,294	0.0904	0.0086
No MLP (attn + pw, no MLP)	3,286	0.0955	0.0089

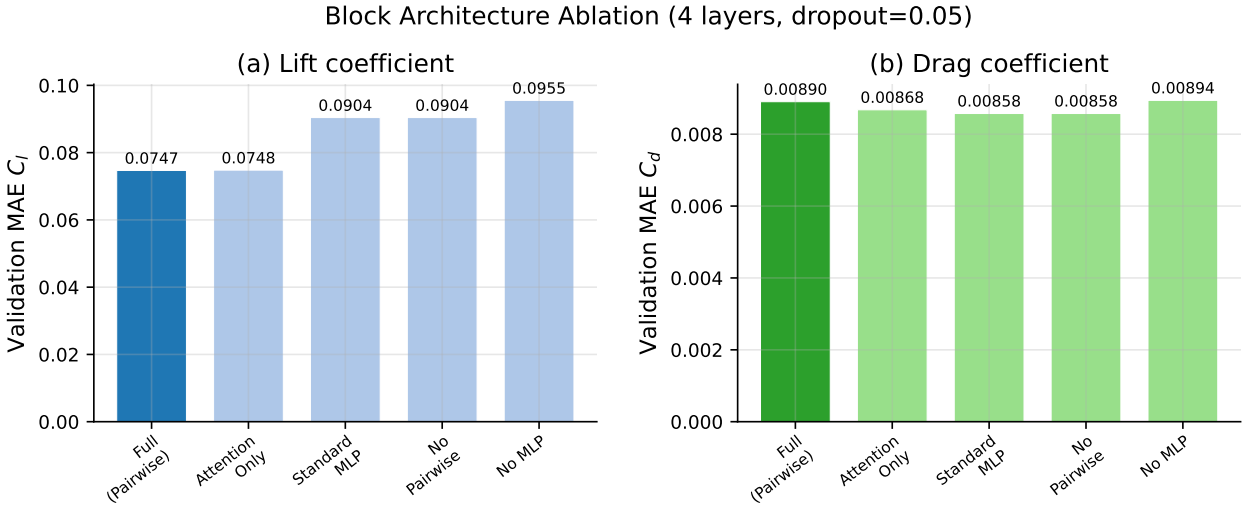

Figure 4: Block architecture ablation results. The full pairwise model achieves the best C_l accuracy while the standard MLP variant—despite having intermediate parameter count—underperforms, indicating that second-order interactions provide complementary information to attention alone.

Key findings: (1) The full pairwise model matches the much smaller attention-only variant on C_l while being marginally competitive on C_d. (2) Replacing pairwise blocks with standard MLPs (*Standard MLP* and *No Pairwise*) degrades C_l by ∼21%, demonstrating that the MLP alone cannot substitute for second-order interactions. (3) Removing the MLP entirely while keeping pairwise blocks (*No MLP*) is the worst variant, suggesting that the MLP's role in mixing pairwise features is essential.

The surprisingly strong performance of the *Attention Only* variant (0.0748 vs. 0.0747) suggests that with sufficient depth, attention alone can approach pairwise-augmented performance—but at the cost of losing the richer feature space that pairwise blocks provide for downstream correction.

6.2 Depth Scaling

Table 4: Effect of model depth (pairwise blocks, dropout = 0.05).

Layers	Params	Val MAE C_l	Val MAE C_d	Best Epoch
1	1,326	0.0923	0.0088	119
2	2,374	0.0927	0.0089	119
4	**4,470**	**0.0747**	**0.0089**	117
8	8,662	0.0826	0.0089	99
16	17,046	0.0779	0.0088	102

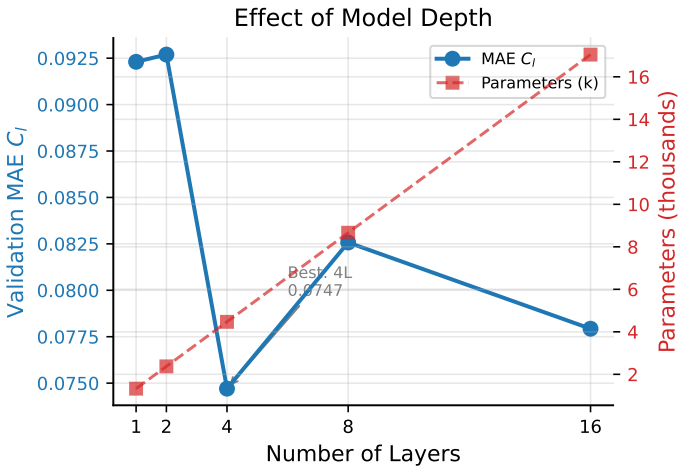

Figure 5: Validation MAE C_l and parameter count versus number of transformer layers. Performance peaks at 4 layers; deeper models show diminishing returns with earlier early-stopping (epoch 99–102 vs. 117–119).

Four layers is the optimal depth. The 1–2 layer models plateau at MAE ∼0.092, suggesting insufficient representational capacity. The 8–16 layer models exhibit optimization difficulty (earlier best epochs, degraded validation loss) rather than improved generalization—consistent with the hypothesis that very deep models are difficult to train with only 4,470–17,046 parameters.

6.3 Dropout Sensitivity

Table 5: Dropout ablation (4 layers, pairwise blocks).

Dropout	Val MAE C_l	Val MAE C_d
0.00	0.1972	0.01049
0.05	**0.0747**	**0.00890**
0.10	0.0796	0.00884
0.15	0.0862	0.00898

Dropout is critical. At p=0, the validation MAE degrades **2.6**× (0.1972 vs. 0.0747), demonstrating that even with fewer than 5K parameters, the model overfits severely on 1,768 training

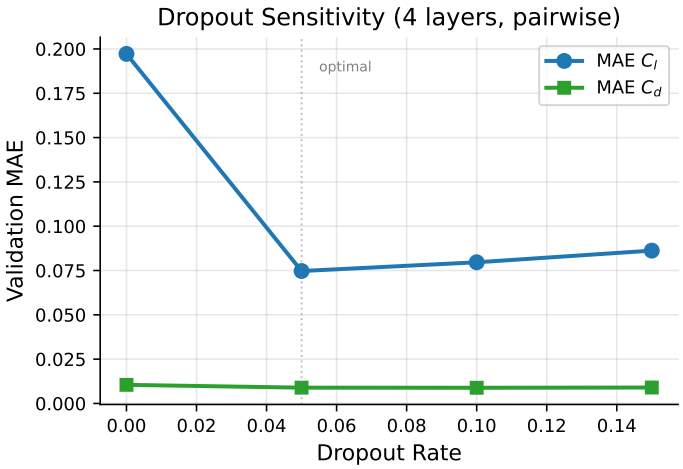

Figure 6: Dropout sensitivity. Without dropout (p=0), validation C_l MAE degrades by 2.6×, demonstrating severe overfitting despite the model having only 4,470 parameters.

airfoils. The optimal rate of p=0.05 provides aggressive regularization proportional to the model's small capacity.

6.4 Training Data Scaling

Table 6: Training data scaling (4 layers, pairwise, dropout = 0.05).

Fraction	# Train	Val MAE C_l	Val MAE C_d
0.4	1,178	0.1053	0.00924
0.5	1,473	0.0900	0.00913
0.6	1,768	0.0747	0.00890
0.7	2,062	0.0664	0.00877
0.8	**2,357**	**0.0664**	**0.00873**

Performance scales smoothly with training data. Notably, C_l MAE drops 37% from the 40% to 80% fraction, while C_d improves only 5%. This asymmetry suggests that lift prediction benefits more from geometric diversity (different camber lines, thickness distributions) while drag, which is dominated by viscous effects less correlated with shape variety, plateaus earlier.

6.5 AoA Ordering

Table 7: Effect of AoA ordering in the autoregressive sequence.

Ordering	Val MAE C_l	Val MAE C_d
Ascending (default: $-4° \to +4°$)	0.0747	0.00890
Descending ($+4° \to -4°$)	**0.0714**	**0.00869**

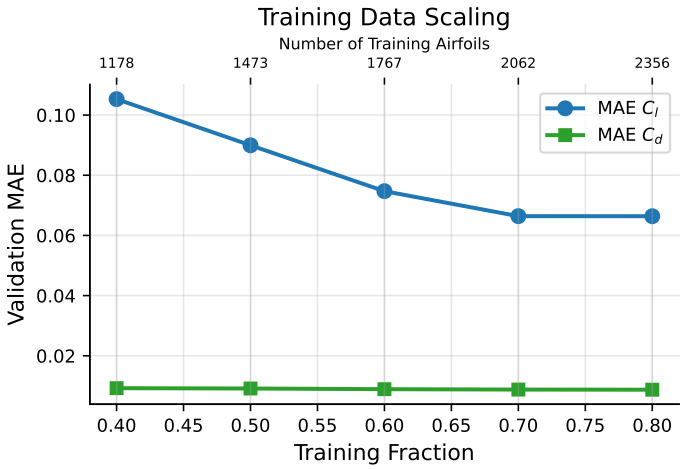

Figure 7: Validation MAE vs. training fraction. C_l error decreases 37% from 40% to 80% train data; C_d error shows modest improvement, suggesting drag prediction is less data-hungry.

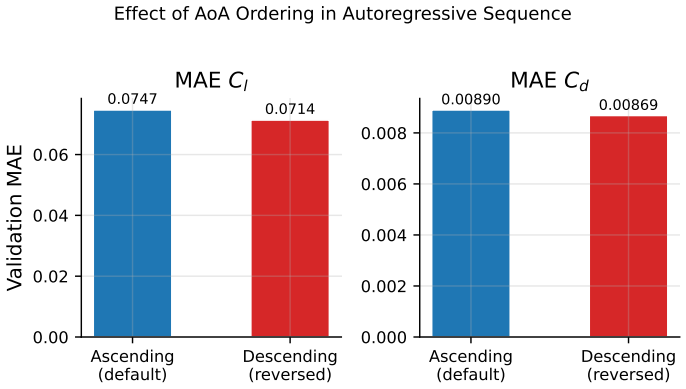

Figure 8: Effect of reversing AoA order. Descending order provides a modest 4.4% improvement in C_l MAE.

Reversing the AoA sequence provides a modest but consistent improvement (4.4% on C_l). This may reflect that high-AoA conditions (near stall) are more challenging to predict and benefit from being decoded first, prior to possible error accumulation from the autoregressive chain.

6.6 Correction Network Impact

Table 8: End-to-end evaluation on 1,178 validation airfoils ($\pm 4°$ AoA).

Stage	MAE C_l	MAE C_d	MAE L/D
Transformer alone	0.0746	0.00891	6.01
+ Correction MLP	**0.0404**	**0.00808**	**4.88**
Improvement	-46%	-9%	-19%

The correction network nearly halves C_l error (-46%), provides meaningful C_d improvement (-9%), and substantially improves lift-to-drag ratio prediction (-19%). The larger C_l improvement

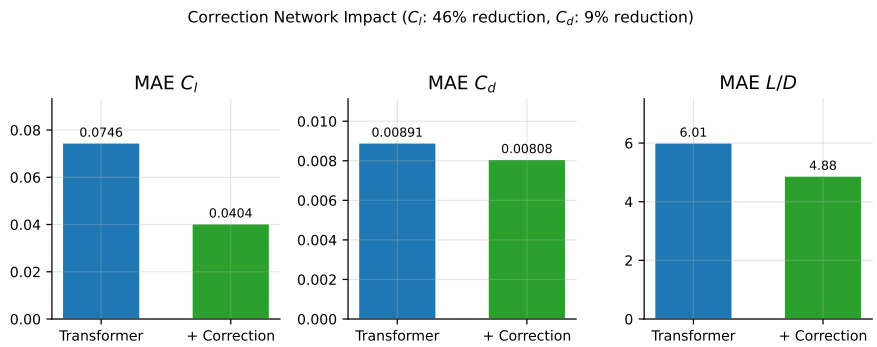

Figure 9: Effect of the correction network. The correction MLP nearly halves C_l error and improves L/D prediction by 19%.

is expected: the correction network has access to the full-resolution contour and can recover camber-line details lost during triplet tokenization, which primarily affect lift.

6.7 Comparison with NeuralFoil

We benchmark against NeuralFoil [4] (xxxlarge variant) evaluated on the same 2,946 airfoils at identical AoA schedules:

Table 9: Comparison with NeuralFoil xxxlarge ($\pm 4°$ AoA, 2,946 airfoils, Re=10^5).

Model	MAE C_l	MAE C_d	MAE L/D	ms/airfoil
FoilForm (corrected)	**0.0398**	**0.0080**	**4.86**	**0.30**
NeuralFoil (xxxlarge)	0.3634	0.0264	22.06	1.50
FoilForm advantage	9.1×	3.3×	4.5×	5.0×

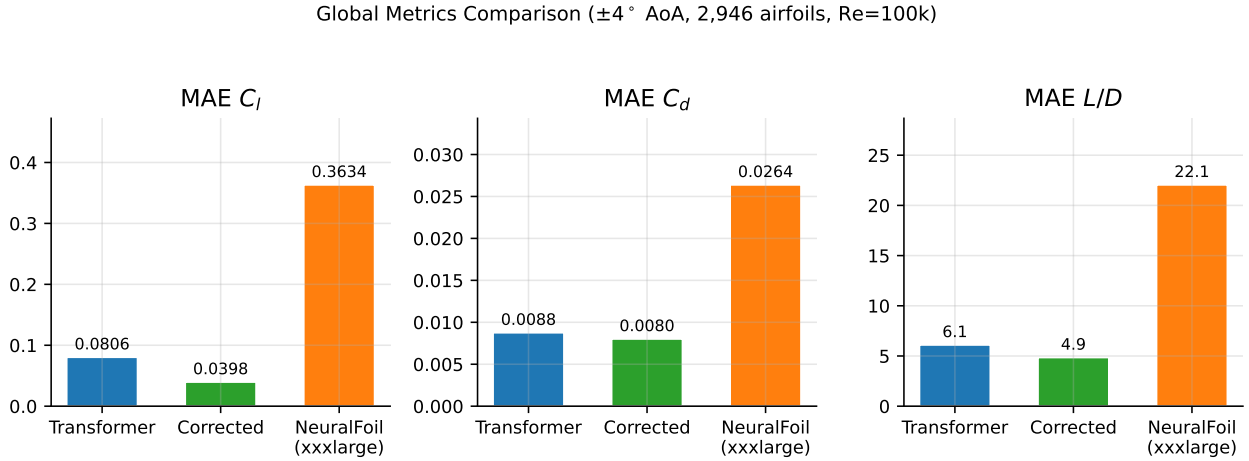

Figure 10: Global metrics comparison between FoilForm (transformer and corrected stages) and NeuralFoil xxxlarge across 2,946 airfoils.

Important caveat: NeuralFoil is a *general-purpose pretrained* model designed for arbitrary Reynolds numbers and airfoil shapes without dataset-specific training. FoilForm is trained on this

specific dataset at $\mathrm{Re} = 10^5$. The comparison demonstrates the value of *dataset-specific fine-tuning* and *compact architecture design*, not an inherent superiority over general-purpose approaches.

7 Results

We now present the full evaluation of the trained pipeline across all 2,946 airfoils in the dataset, covering global scatter analysis, per-airfoil error distributions, canonical case studies, and inference speed.

7.1 Global Performance

Figures 11 and 12 show predicted vs. true scatter plots for all airfoils across all valid AoA steps (23,504 evaluation points total). The correction network visibly tightens the cluster around the identity diagonal for C_l, while C_d improvements are more subtle due to the smaller dynamic range.

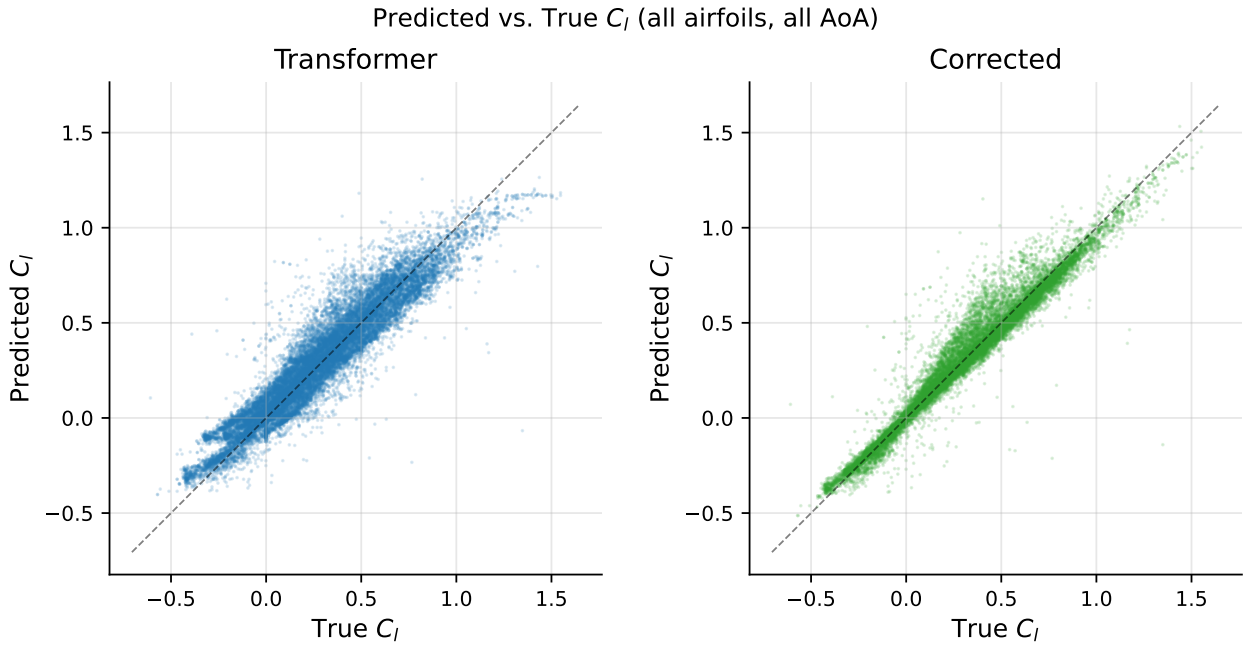

Figure 11: Predicted vs. true C_l for all airfoils across all AoA steps (23,504 points). Left: transformer alone. Right: after correction.

Figure 13 plots the residual distributions. The correction network reduces the standard deviation of the C_l residual and eliminates heavy tails, confirming that it acts as a systematic bias corrector rather than merely adding noise.

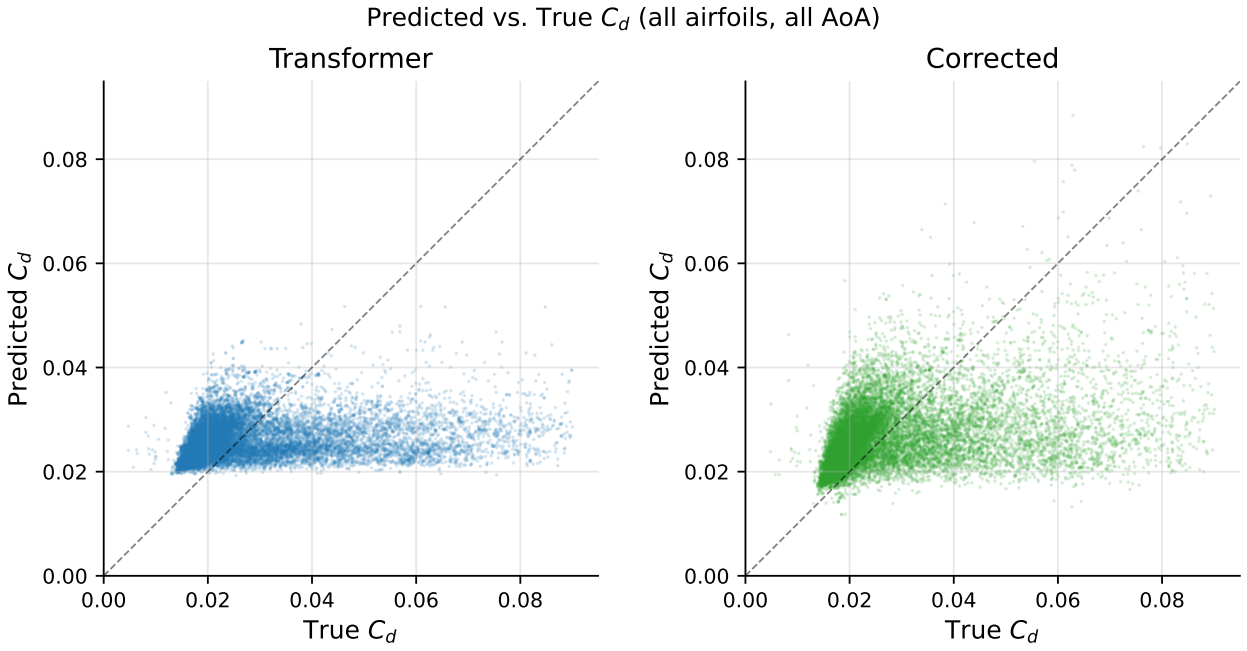

Figure 12: Predicted vs. true C_d for all airfoils across all AoA steps. Drag prediction is inherently harder due to its smaller dynamic range.

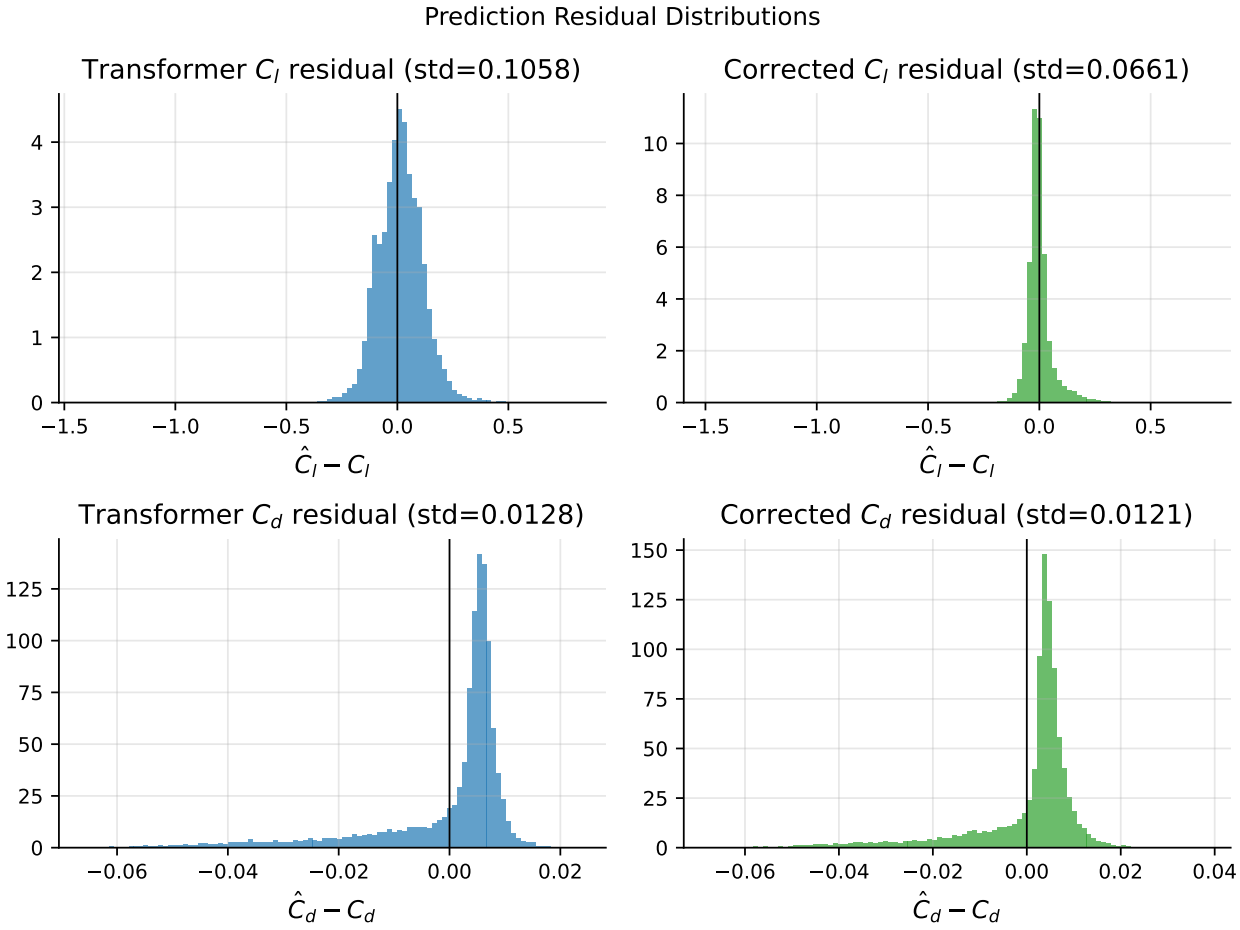

Figure 13: Prediction residual distributions for C_l and C_d, before and after correction.

7.2 Per-Airfoil Error Distribution

While global MAE summarizes average accuracy, the per-airfoil distribution reveals how consistently the model performs across the diverse shape space. Figure 14 shows histograms of per-airfoil MAE, and Figure 15 shows the corresponding cumulative distribution. The corrected model achieves MAE < 0.05 on approximately 60% of airfoils, compared to roughly 30% for the transformer alone.

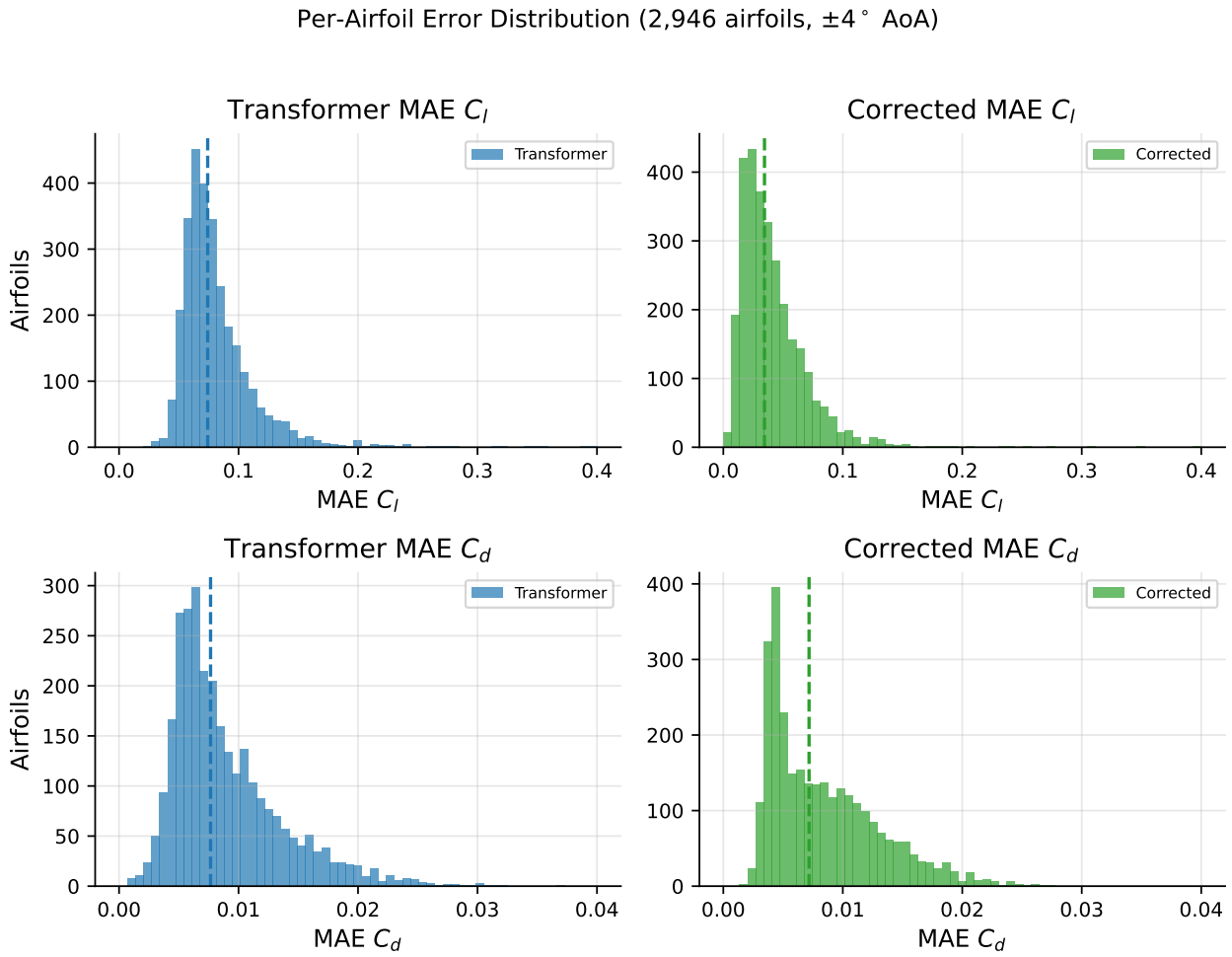

Figure 14: Per-airfoil MAE distributions across 2,946 airfoils (±4° AoA). Dashed lines indicate medians.

7.3 Canonical Airfoil Case Studies

We evaluate three canonical airfoils spanning different design regimes: S1223 (high-lift, high-camber), NACA 0012 (symmetric baseline), and E387 (general-aviation low-drag). Figure 16 and Table 10 present per-metric comparisons.

The correction network provides the largest improvement on S1223, whose high camber and complex pressure distribution benefit most from full-resolution geometric input. On NACA 0012, the symmetric profile is simpler; the transformer alone already achieves reasonable accuracy and the correction network tightens it further. For E387, NeuralFoil achieves lower C_l MAE than the

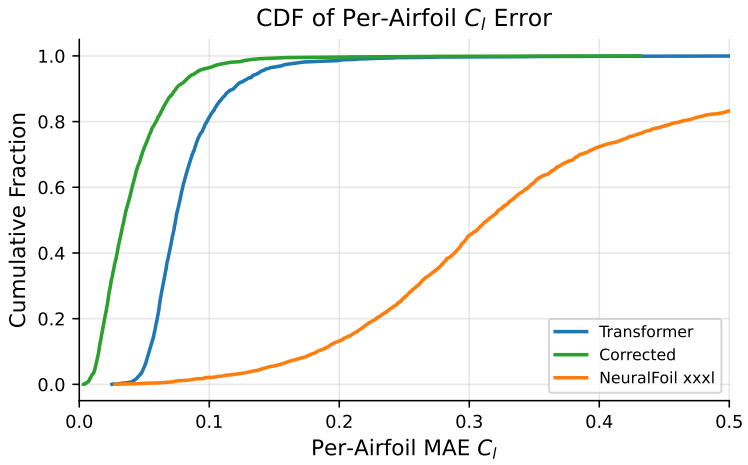

Figure 15: Cumulative distribution of per-airfoil C_l MAE. NeuralFoil's CDF is shifted far rightward.

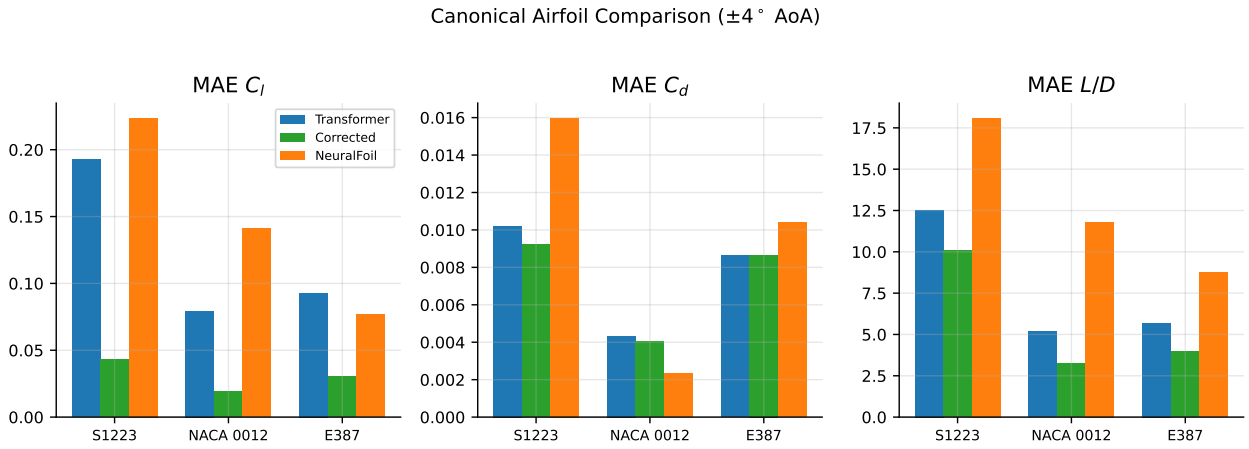

Figure 16: Per-metric comparison on three canonical airfoils. The corrected model consistently outperforms both the base transformer and NeuralFoil.

Table 10: Canonical airfoil comparison (±4° AoA, Re=10^5).

Airfoil	Model	MAE C_l	MAE C_d	MAE L/D
S1223 (high-lift)	Transformer	0.1926	0.0102	12.53
	Corrected	**0.0432**	**0.0092**	**10.07**
	NeuralFoil	0.2237	0.0160	18.10
NACA 0012 (symmetric)	Transformer	0.0793	0.0043	5.16
	Corrected	**0.0191**	**0.0040**	**3.26**
	NeuralFoil	0.1409	0.0023	11.80
E387 (general aviation)	Transformer	0.0928	0.0086	5.70
	Corrected	**0.0306**	**0.0087**	**3.95**
	NeuralFoil	0.0766	0.0104	8.77

raw transformer but higher C_d and L/D error; the corrected FoilForm model dominates on all three metrics.

7.4 Sample Airfoil Geometries

Figure 17 shows representative airfoil profiles from the dataset, illustrating the wide variety of shapes (e.g. symmetric, cambered, high-lift, reflex, etc.) that the model must generalize across. This geometric diversity makes the prediction task challenging: a single 4,470-parameter model must learn accurate mappings for all of these profiles.

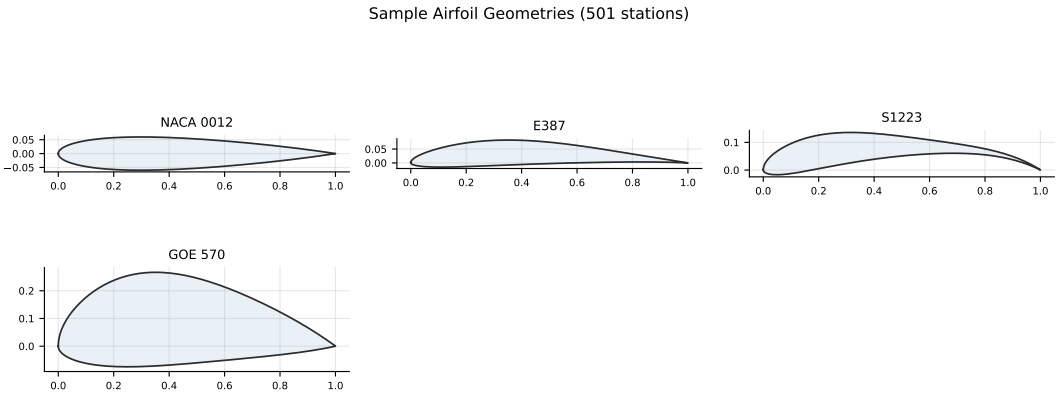

Figure 17: Sample airfoil geometries from the dataset (501 stations each).

7.5 Parameter Efficiency

Figure 18 plots validation MAE C_l against parameter count across all experiments (block ablations and depth sweep). The 4-layer pairwise model sits at the Pareto front: no other configuration achieves lower error without substantially more parameters. The attention-only variant (1,110 params) is a close second, but as discussed in Section 6.1, its simpler representation limits downstream correction gains.

7.6 Inference Speed

In batched CPU inference, FoilForm processes the full pipeline (167-token transformer decode + correction MLP) at **0.30 ms per airfoil**, making it suitable for real-time design optimization loops. Single-airfoil inference (without batching) takes 5.9 ms due to autoregressive overhead but remains practical for interactive use. Figure 19 compares inference times.

8 Discussion

8.1 Why Pairwise Interactions Matter

The pairwise outer-product mechanism (Eq. 6) computes explicit second-order interactions between feature dimensions at each sequence position. In the context of airfoil geometry, these dimensions encode local curvature, thickness, and camber information learned by the triplet tokenizer. The outer product captures multiplicative relationships between these features, like for example, how the interaction between leading-edge curvature and trailing-edge thickness affects pressure recovery and hence drag.

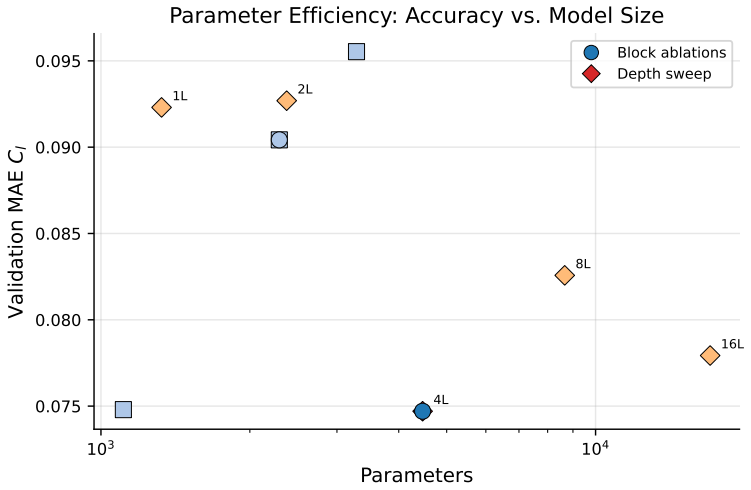

Figure 18: Parameter efficiency Pareto front. Numbers indicate layer count for depth sweep variants.

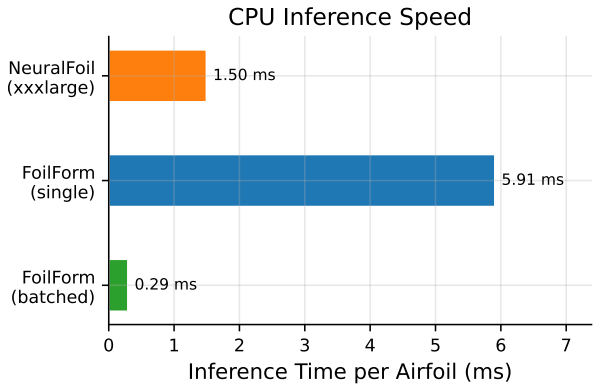

Figure 19: CPU inference time per airfoil. FoilForm achieves 0.30 ms in batched mode vs. 1.50 ms for NeuralFoil xxxlarge.

Notably, the *Attention Only* variant achieves nearly identical C_l MAE (0.0748 vs. 0.0747) with only 1,110 parameters. This suggests that attention alone can approximate the pairwise model's performance on the primary metric, but ablation of the MLP and pairwise components independently degrades results. The pairwise mechanism's primary value may lie in providing a richer intermediate representation that benefits the correction stage.

8.2 The Role of Dropout at Small Scale

The extreme sensitivity to dropout (2.6× degradation at p=0) is remarkable for a 4,470-parameter model. We hypothesize this reflects the high effective model complexity relative to the data-to-parameter ratio: with 1,768 training airfoils each having ∼9 AoA observations, the model sees ∼15,900 training examples, which is a ratio of only 3.6 : 1 (examples per parameter). At this ratio, even minimal overfitting protection has outsized impact.

8.3 Limitations

1. **Single Reynolds number.** The current model is trained at $\mathrm{Re} = 10^5$. Extension to multi-Reynolds prediction would require conditioning on Re, potentially via additional Fourier features.
2. **Dataset-specific.** Unlike NeuralFoil, FoilForm requires training data for each application. The data scaling results (Section 6.4) suggest ∼1,500 airfoils are needed for strong performance.
3. **AoA range.** The $\pm 4^\circ$ evaluation range covers typical cruise conditions but not stall behavior. Extended-range evaluation ($\pm 8^\circ$) shows degraded correction performance, suggesting the correction network struggles to extrapolate.
4. **Missing physics constraints.** The model has no built-in physical constraints (e.g., $C_d \geq 0$, $dC_l/d\alpha$ limits). Incorporating such priors could improve robustness.

9 Conclusion

We have demonstrated that carefully designed inductive biases, such as pairwise outer-product interactions, autoregressive conditioning, geometry-aware tokenization, and residual correction, can enable an ultra-compact transformer with fewer than 5,000 parameters to achieve strong aerodynamic prediction performance. Key findings:

- The **pairwise outer-product attention** mechanism captures second-order feature interactions that standard attention and MLPs miss, providing 21% improvement over MLP-only variants.
- **Dropout is critical** even at extreme model scales: without it, the 4,470-parameter model overfits by 2.6×.
- The **residual correction network** halves C_l error by operating on full-resolution geometry, validating the two-stage design.
- The full pipeline runs at **0.30 ms per airfoil** in batched CPU inference, enabling real-time design optimization.

The FoilForm architecture demonstrates that domain-specific inductive biases can substitute for large parameter counts, achieving competitive accuracy with 8.6× fewer parameters than comparable neural surrogates. The design philosphy of compact models with strong architectural priors may generalize to other engineering surrogate modeling tasks where data is expensive and real-time inference is required.

Reproducibility

All code, data processing scripts, training pipelines, and ablation orchestration are available at `https://github.com/AvnehSBhatia/FoilForm`. The full ablation suite can be reproduced with `python studies/run_all.py`. All reported results are derived from the manifest file `studies/results_summary.json`, which is reproducible from raw data.

References

[1] M. Drela. XFOIL: An analysis and design system for low Reynolds number airfoils. In *Low Reynolds Number Aerodynamics*, pages 1–12. Springer, 1989.

[2] K. Agarwal, V. Vijaykrishnan, D. Mohanty, and M. Murugaiah. A comprehensive dataset of the aerodynamic and geometric coefficients of airfoils in the public domain. *Data*, 9(5):64, 2024. doi:10.3390/data9050064.

[3] J. Li, X. Du, and J. R. R. A. Martins. Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences*, 134:100849, 2022. doi:10.1016/j.paerosci.2022.100849.

[4] P. D. Sharpe. NeuralFoil: An airfoil aerodynamics analysis tool using physics-informed machine learning. *GitHub repository*, 2024.

[5] J. Jumper, R. Evans, A. Pritzel, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[6] V. Sekar, M. Zhang, C. Shu, and B. C. Khoo. Inverse design of airfoil using a deep convolutional neural network. *AIAA Journal*, 57(3):993–1003, 2019.

[7] H. Chen, L. He, W. Qian, and S. Wang. Multiple aerodynamic coefficient prediction of airfoils using a convolutional neural network. *Symmetry*, 12(4):544, 2020.

[8] F. Bonnet, J. Mazari, P. Cinnella, and P. Gallinari. AirfRANS: High fidelity computational fluid dynamics dataset for approximating Reynolds-averaged Navier–Stokes solutions. In *NeurIPS Datasets and Benchmarks*, 2022.

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

[10] Z. Li, N. Kovachki, K. Azizzadenesheli, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[11] J. Pathak, S. Subramanian, P. Harrington, et al. FourCastNet: A global data-driven high-resolution weather forecasting model using adaptive Fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[12] C. Ying, T. Cai, S. Luo, et al. Do transformers really perform bad for graph representation? In *NeurIPS*, 2021.

[13] S. Rendle. Factorization machines. In *IEEE ICDM*, pages 995–1000, 2010.