

PMRT: A Training Recipe for Fast, 3D High-Resolution Aerodynamic Prediction

Sam Jacob Jacob
Volkswagen AG
Wolfsburg, Germany

sam.jacob.jacob@volkswagen.de

Carsten Othmer
Volkswagen AG
Wolfsburg, Germany

carsten.othmer@volkswagen.de

Markus Mrosek
Volkswagen AG
Wolfsburg, Germany

markus.mrosek@volkswagen.de

Harald Köstler
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany

harald.koestler@fau.de

Abstract

The aerodynamic optimization of cars requires close collaboration between aerodynamicists and stylists, while slow, expensive simulations remain a bottleneck. Surrogate models have been shown to accurately predict aerodynamics within the design space for which they were trained. However, many of these models struggle to scale to higher resolutions because of the 3D nature of the problem and data scarcity. We propose Progressive Multi-Resolution Training (PMRT), a probabilistic multi-resolution training schedule that enables training a U-Net to predict the drag coefficient (c_d) and high-resolution velocity fields ($512 \times 128 \times 128$) in 24 h on a single NVIDIA H100 GPU, 7× cheaper than the high-resolution-only baseline, with similar accuracy. PMRT samples batches from three resolutions based on probabilities that change during training, starting with an emphasis on lower resolutions and gradually shifting toward higher resolutions. Since this is a training methodology, it can be adapted to other high-resolution-focused backbones. We also show that a single model can be trained across five datasets from different solvers, including a real-world dataset, by conditioning on the simulation parameters. In the DrivAerML dataset, our models achieve a $c_d R^2$ of 0.975, matching literature baselines at a fraction of the training cost.

1. Introduction

Improving the aerodynamics of a car is essential, as it improves fuel economy throughout the vehicle’s lifetime and is even more paramount for electric vehicles, where it increases range for the same battery capacity. Typically, the aerodynamic optimization process consists of iterations between aerodynamicists and stylists as both the aerodynam-

ics and design of the car are essential. However, a major bottleneck are the time-consuming CFD simulations, which take on average 20 h and up to several days on 1000 CPU cores. This bottleneck leads to fewer iterations between aerodynamicists and stylists or reduced exploration of the design space. Aerodynamicists could use surrogate models that enable real-time aerodynamic predictions to explore the design space before committing to testing a few promising variations using expensive simulations.

1.1. Related work

Existing studies have successfully applied surrogate models for various general fluid dynamics applications [6, 23] and automotive aerodynamic applications [3, 19, 28, 31, 39]. There are several approaches to creating surrogate models, and we highlight a few subsequently. Convolutional neural network (CNN)-based methods typically operate on intermediate volumetric encodings (instead of meshes) like binary masks [30, 42], signed distance fields (SDFs) [5, 7, 12, 19, 20], or more recently alternative encodings like triplanes [8] and factorized implicit grids [9]. Earlier CNN-based approaches typically did not predict surface fields; recent approaches [8, 9] address this. Graph-based [28, 29, 35] and neural-operator-based [3, 22, 31] models work natively on meshes or point clouds, avoiding volumetric intermediates. One of the persistent challenges with surrogate models has been training at higher spatial resolutions. Several methods use low-resolution voxels or significantly downsampled meshes. Higher resolutions can improve geometry representation and enable the prediction of finer flow details. In 3D, memory and compute scale cubically, and many otherwise strong methods struggle to scale or become computationally expensive. Compounding this, high-fidelity aerodynamics simulations are scarce due to the high computational cost. Recently, several works have

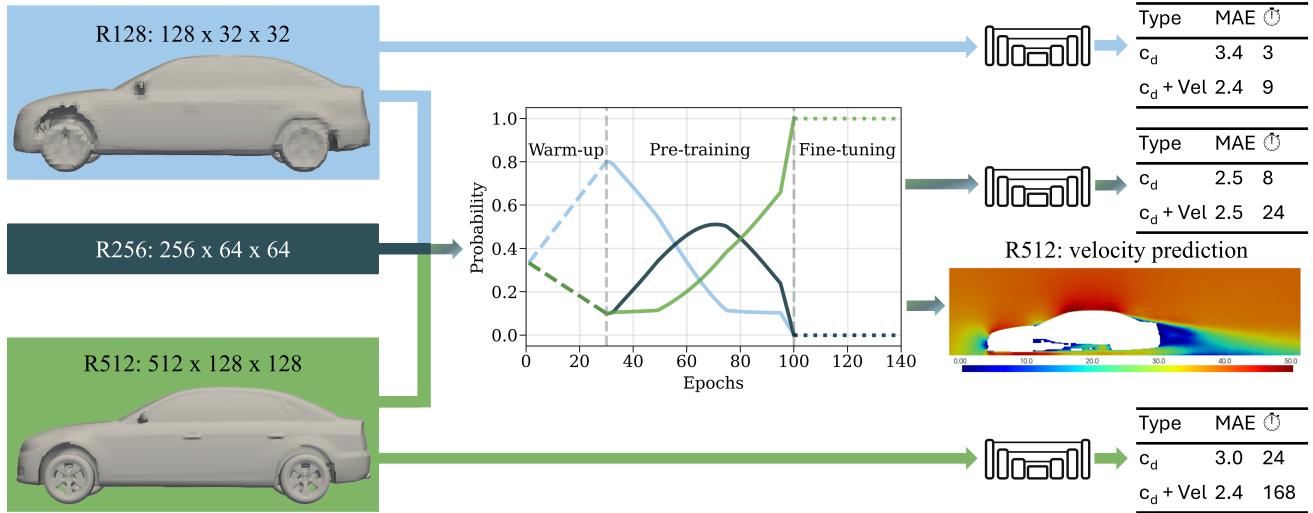


Figure 1. Overview of PMRT and single-resolution baselines. Box colors, labeled with the resolution name (with example geometries reconstructed from SDFs for R128 and R512), represent the resolution’s color used in the plot. Top/bottom: baselines trained only at low or high resolution. Middle: PMRT pipeline; graph shows resolution probabilities over epochs for a hypothetical example, during training, batches are sampled based on these probabilities. Right: c_d MAE and wall-clock time in hours for c_d -only and c_d +velocity variants with an example PMRT R512 velocity field prediction.

addressed this problem in automotive aerodynamics, including: scalable GNN [28] (partitioned multi-scale graphs with halo exchange); scalable neural operators [3, 31] (anchored branched universal physics transformers; decomposable neural operator); alternative representations like triplane [8]; factorized implicit grids [9]; and super-resolution models [46] that predict high-resolution fields from coarse fields.

In parallel, there have been various high-resolution vision strategies: these include fine-tuning/curriculum training at higher resolutions [15, 21, 24, 38, 44, 45]; mixed-resolution/patch-size and aspect-agnostic training [4, 10, 43]; zoom-in pyramids [36, 41]; backbone improvements [24, 47]; sparsity [48, 49] and token reduction [1, 51]. However, porting these ideas to 3D CFD surrogates is not always trivial; for example: in 3D memory scales cubically with voxel size; required resolution jumps are significant (e.g., from $128 \times 32 \times 32$ to $512 \times 128 \times 128$); sharp switches in resolution could destabilize training; scale-based augmentations that help vision tasks can occlude geometry, and even change the actual physical targets; vision transformers, even with these improvements, can be too expensive in cases where time-to-first prediction is critical. Nevertheless, these ideas inspire our training recipe.

1.2. Gap and Overview

Even with the above advances, we still lack a training-only recipe inspired by mixing-resolutions or curriculum-style training that scales proven voxel-based CNN approaches to high-resolution field prediction without significant archi-

tectural changes. Rather than abandoning CNNs for high-resolution-focused backbones, we adapt the training and propose Progressive Multi-Resolution Training (PMRT- see Fig. 1 for an overview): a smooth, probabilistic schedule that mixes resolutions during training, starting with an emphasis on lower resolutions and gradually transitioning to higher resolutions within a single run, avoiding hard resolution switches and enabling accurate high-resolution volumetric predictions with low compute budgets.

The availability of aerodynamic simulations is limited by high compute cost and significant human effort; for example, the estimated compute cost for the DrivAerML dataset [2] (500 simulations) is $\approx \$1M$ [3]. With the currently available data, it is not yet feasible to train a generalizable model that accurately predicts the aerodynamics of any car geometry, but surrogate models have been shown to generalize well within the design space on which they were trained; this already helps aerodynamicists. As a result, surrogate models must be retrained for new projects or for baselines that differ substantially from existing geometries.

Even with high-fidelity simulations, it is essential to validate them several times against wind-tunnel experiments during vehicle development. Similarly, the primary application of surrogate models we evaluate in this study is to explore the design space and then validate promising solutions using simulations. Therefore, when the training time is too high, it may no longer make sense to use surrogate models, as a short time-to-first prediction is crucial on new datasets. With lower training time, aerodynamicists can utilize additional data or a new dataset to train models overnight or

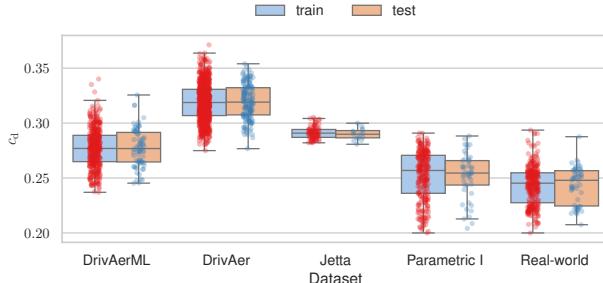


Figure 2. Distribution of c_d across splits for all datasets. The c_d values of the real-world dataset and Parametric I are subtracted by a constant such that the sample with the lowest c_d has a value of 0.2 to protect sensitive c_d data.

within a day. For these reasons, we impose a one-day training budget for our models. Finally, we train two model families: c_d -only and $c_d +$ velocity fields, since velocity fields are often unavailable (not archived due to storage costs: raw DrivAerML dataset ≈ 31 TB) or unnecessary for a given study, and they increase training time. Our main contributions include:

- Progressive Multi-Resolution Training (PMRT): a probabilistic multi-resolution schedule that enables accurate high-resolution velocity field prediction with low training costs (single NVIDIA H100 GPU: ≈ 24 h for all datasets, 7 \times faster than training only on high-resolution; ≈ 4 h for DrivAerML). It also improved c_d prediction accuracy for c_d -only models using high-resolution SDF inputs.
- Competitive DrivAerML accuracy with a U-Net, at substantially lower wall-clock time compared to methods reported in the literature.
- A single U-Net conditioned on simulation parameters trained jointly on five datasets (≈ 1900 training simulations) created using different solvers.

2. Datasets

We use five datasets, which can be classified into two types: parametric and real-world (or non-parametric). Parametric datasets are generated by morphing a baseline geometry for a set of chosen geometric parameters within predefined ranges; real-world datasets are generated from free-form deformations applied across multiple baselines without explicit parameterization. We refer to the latter as ‘real-world’ because they are similar to the datasets curated from simulations generated during vehicle development without being explicitly created for training models.

Table 1 gives an overview of the datasets. DrivAer [26] and DrivAerML [2] datasets are parametric datasets based on the publicly available DrivAer geometry introduced by Heft *et al.* [14]. DrivAerML [2] is a publicly available parametric dataset of 484 simulations with 16 morphing

parameters, simulated using OpenFOAM. DrivAer [26] is a parametric dataset of 1,000 simulations with 15 morphing parameters. Jetta [25] is a parametric dataset of 100 simulations with 6 morphing parameters. The DrivAer and Jetta datasets were simulated using Altair’s ultraFluidX, a lattice Boltzmann solver, following the simulation setup described by Mrosek *et al.* [26]. Parametric I and real-world datasets are internal datasets created by an automotive manufacturer; the simulations were performed using OpenFOAM following the setup by Islam *et al.* [18]. Parametric I is a parametric dataset comprising 308 simulations with seven morphing parameters. The real-world dataset is a non-parametric dataset comprising 374 simulations with 39 baseline geometries (Sedan or CUV). For all datasets, we use time-averaged c_d and velocity fields. Two types of time-averaging are employed in the datasets: (1) classic averaging, where simulations are run for a total of 4 s of physical time, and the final 2 s are averaged; and (2) a more recent dynamic averaging (using the tool Meancalc), where the total simulated physical time (with a maximum cap), start and end of the averaging window are dynamically adjusted to achieve a target statistical accuracy. DrivAer, Jetta, and Parametric I use classic averaging. DrivAerML [2] uses dynamic averaging to reach ± 1.5 drag counts accuracy. Approximately one-third of real-world simulations use classic averaging, while the rest use dynamic averaging to reach ± 0.5 drag counts accuracy.

Figure 2 shows the distribution of c_d across the training and test partitions. We generate the splits using stratified splitting based on c_d and morphing parameters for the parametric datasets, and on c_d and the baseline group (group of vehicles derived from the same baseline geometry) for the real-world dataset. For the validation split, we use 15 % of the samples from the training partition. Refer to the supplementary material for further evaluation of the datasets.

3. Methodology

3.1. Preprocessing

We predict the velocity field on a Cartesian grid in a region of interest (ROI) around the car. The ROI is chosen (DrivAerML ROI in supplementary) based on the region where most of the interesting features in the velocity field occur, and we interpolate the simulated velocity fields onto the Cartesian grid in the ROI. Cars are typically represented as meshes: an unstructured form of geometry representation that cannot be directly passed to CNNs. We generate SDFs for these geometries in a Cartesian grid in the same ROI where the velocity field is predicted. Each SDF value is the shortest distance from the voxel (center) to the surface of the geometry, with the sign being positive for the voxels outside the geometry and negative otherwise. We generate the SDFs using OpenVDB [27]. During training, we apply

Dataset	Group	Parametric (✓/✗)	No. morphing parameters	No. training samples	No. test samples	No. baseline geometries	Solver
DrivAerML	DrivAerML	✓	16	411	73	1	OpenFOAM
DrivAer	Parametric	✓	15	850	150	1	ultraFluidX
Jetta	Parametric	✓	6	85	15	1	ultraFluidX
Parametric I	Parametric	✓	7	262	46	1	OpenFOAM
Real-world	Real-world	✗	—	319	55	39	OpenFOAM

Table 1. Summary of the datasets used. We indicate if the dataset is parametric with ✓/✗, along with the number of morphing parameters, training / test samples, baseline geometries, and solver. We group the datasets into DrivAerML (public benchmark), Parametric (parametric changes, single baseline), and Real-world (free-form, multiple baselines), to enable easier comparison and balanced evaluation.

the same data augmentation used by Jacob *et al.* [20].

Different solvers and setup choices introduce variability; to manage this, we pass simulation parameters as an additional input: these include the simulation tool used and the setup parameters (for example, simulated physical time). We normalize (to have zero mean and unit variance) the simulation parameters and then reduce the dimensionality using principal component analysis before passing them to the model. The SDF and velocity fields are also normalized to have zero mean and unit variance. We use three voxel (grid) resolutions: $128 \times 32 \times 32$ (R128), $256 \times 64 \times 64$ (R256), and $512 \times 128 \times 128$ (R512). We train the models to predict at R128 or R512, and use R256 only during pre-training.

3.2. Progressive Multi-Resolution Training (PMRT)

Training models using high-resolution SDFs for aerodynamic applications is computationally expensive and challenging due to data scarcity. To address this, we propose PMRT, a probabilistic multi-resolution training schedule that balances compute and accuracy. During training, we dynamically sample batches from multiple resolutions based on probabilities that vary throughout the training process. The schedule emphasizes lower resolutions at the start and gradually transitions to higher resolutions. In contrast to sharp resolution switches, the PMRT schedule helps to stabilize training, and since scale-based augmentations are not applicable, a per-resolution probability floor keeps all resolutions sampled for most of training, encouraging multiscale feature learning.

We use a three-phase schedule (warm-up, pre-training, and fine-tuning) to train on the R128, R256, and R512 SDFs. Figure 1 illustrates an example PMRT schedule and shows how it affects training costs. The warm-up phase begins with equal probabilities across all resolution levels and linearly interpolates to the starting probabilities of the pre-training phase. During the pre-training phase, we gradually transition the sampling probabilities from emphasizing lower resolutions early in training to higher resolutions later in training. The pre-training schedule is generated by

discretizing a Gaussian distribution over the resolution indices, where the mean moves from low toward higher resolutions while the standard deviation shrinks. To prevent neglecting any resolution, we enforce the per-resolution probability floor by clipping the raw sampling probabilities at a minimum value. At the end of the pre-training phase for five epochs, we linearly interpolate the probabilities from the mixed distribution probabilities to the fine-tuning probabilities (only the high resolution). During warm-up and pre-training, we multiply the batch size by a factor of 4 for all the lower resolutions. Finally, we have the fine-tuning phase, where the model continues training for additional epochs using only the highest resolution. The warm-up phase slightly improves the stability of the training and convergence. Pre-training accounts for most of the convergence; fine-tuning specializes the model to R512. See the supplementary material for the implementation details and ablations of PMRT design choices.

3.3. Architecture

We use a U-Net [33] closely following Jacob *et al.* [19]. The encoder extracts features from the SDF, which are concatenated with the processed simulation parameter input to form the bottleneck vector. From this bottleneck, three identical decoders predict the three velocity components, and a prediction head predicts c_d . The architecture is modular and can be extended for additional volumetric fields and coefficients. For c_d -only models, we omit the velocity decoders.

We pass the SDF input through an input convolution block followed by six encoder blocks (concurrent spatial & channel squeeze & excitation layer [16, 34] → PReLU activation [13] → convolution layer → max-pooling (with stride 2) → group normalization [50] with residual skip connection) and finally a global pooling layer. The simulation parameter input is passed through an MLP block (linear layer → activation → dropout → normalization) to expand the simulation parameters to have the same shape as the encoder’s output, followed by a multi-head self-attention layer. The resulting vector and the encoder output are then concatenated to get the bottleneck vector.

The bottleneck vector is passed through three U-Net decoders, one for each component of velocity, which mirrors the encoder and has six decoder blocks followed by an output convolution layer. Each decoder block has a transpose convolution layer instead of the convolution and pooling layer. In addition to the standard skip connections present in the U-Net architecture, we introduce a direct skip from the input SDF to the output convolution: we compute a binary mask from the SDF, concatenate the SDF and mask, pass it through two convolution blocks (convolution → normalization → activation) to match channels, and then concatenate this feature map before the output convolution layer.

We pass the bottleneck vector through a coefficient prediction head (transformer layer → MLP block → linear layer) to predict c_d . The transformer layer allows the model to learn the relationships between the simulation parameters and the features extracted from the encoder.

To support multiple resolutions, we apply global pooling at the end of the encoder to produce a fixed-size output. In the decoder, we use nearest-neighbor interpolation to align tensor sizes when they do not match in the first decoder block. For the c_d -only models, we use a batch size of 64, and for c_d + velocity field models, we use a batch size of 16. For training, we use a cyclic learning rate [37] with the NAdam optimizer and apply stochastic depth regularization [17]. We present ablations on our architectural choices in the supplementary material.

3.4. Loss function

We use a Smooth L1 loss for both c_d and velocity. We build a volumetric loss weighting from two complementary strategies: distance-based (Trinh *et al.* [46]) and variance-based (Jacob *et al.* [19]) weighting. Distance-based weighting assigns higher weights to voxels near the vehicle surface, which decay with distance. Variance-based weighting is precomputed for a dataset based on the cell-wise velocity field variance across the training set; that is, cells with higher velocity field variance receive larger weights, typically those near the surface and in the wake.

$$w = \mathcal{N} \left[\mathcal{N} \left(c + \frac{\alpha \cdot \text{mask}}{\alpha + u\text{SDF}} \right) + \mathcal{N} \left(\|\nabla U\|_2^{1/4} \right) \right] \quad (1)$$

Equation (1) represents the combined weighting function. $(\frac{\alpha \cdot \text{mask}}{\alpha + u\text{SDF}})$ is the distance-based weighting from Trinh *et al.* [46]. The mask is 1 for regions with fluid flow and 0 otherwise, and $u\text{SDF}$ is the unsigned distance to surface. We add a constant c to the distance-based weighting so that the cells inside the vehicle geometry have a non-zero weight. This ensures the model is penalized for predicting non-zero velocities inside the geometry. α and c are constants corresponding to 5.5 (from Trinh *et al.* [46]) and 0.75. $\mathcal{N}[\cdot]$ denotes normalization so that the component has a unit mean for the sample. For $c = 0.75$, after normalizations,

interior voxels typically have a weight ≈ 0.25 in representative samples. To avoid precomputing variances, we use the gradient of the velocity magnitude ($\|\nabla U\|$ is the L2 norm of the gradient of the velocity magnitude). The gradient of velocity magnitude is higher close to the vehicle's surface and in the wake, but it can spike near the vehicle's surface, in some regions of the wake, and contains some noise. To obtain a smoother weight, we raise the gradient magnitude to a power of 0.25; the exponent was empirically chosen to preserve the goal of the weighting. To ensure the two terms contribute equally, we normalize each term, sum them, and then renormalize.

3.5. Benchmark & Evaluation

Generating simulations exclusively for training models is expensive, and to sustainably train in the future, we hope most training data comes from archived simulations from past vehicle development projects. These simulations likely use different simulation parameters and solvers; to show that it is possible to train a single model on diverse datasets and to encourage learning shared features that transfer across datasets, we train a single model jointly on all datasets. We report group-averaged metrics on five datasets to enable easier comparison between models. We partition the datasets into three groups (Tab. 1 - DrivAerML (public benchmark), Parametric (parametric changes, single baseline), and Real-world (free-form, multiple baselines)) and compute metrics within each group. We then average the group metrics so each group contributes equally. The only multi-dataset group is 'Parametric'; since the group has a disproportionately large number of DrivAer samples, we average across datasets within this group so each dataset contributes equally. For the c_d prediction, we report the Mean Absolute Error (MAE) and the Maximum Absolute Error (MaxAE; supplementary only) in drag counts¹.

Since DrivAerML is the only publicly available dataset used in our study, we retrain the best models using only the DrivAerML dataset based on the splits used in the benchmark study by Tangsali *et al.* [39]; the same split is used by all the models compared. The split has a total of 48 test samples, and 20% of the test samples are out-of-distribution samples. We compare our best models with AB-UPT [3], X-MeshGraphNet [28, 39], FIGConvNet [9, 39] and DoMINO [31, 39] on the DrivAerML [2] dataset; hereafter, we refer to these models as the literature baselines. Since some literature baselines report only the drag force R^2 score, we report the drag force R^2 score and $c_d R^2$ when comparing with the literature baselines. For velocity field prediction, we report the relative L2 error:

¹One drag count = 0.001 drag.

Variant (resolution)	Velocity (✓/✗)	PMRT (✓/✗)	Pre-training epochs	Fine-tuning epochs	Group: c_d MAE ↓	Group: relative L2 error ↓ [%]	Wall-clock time [h]
R128 ‡	✗	✗	–	–	3.4	–	3
R512	✗	✗	–	–	3.0	–	24
R512 *	✗	✗	–	–	3.0	–	66
R512	✗	✓	200	50	2.8	–	5
R512 ‡	✗	✓	400	50	2.5	–	8
R512	✗	✓	200	400	2.5	–	16
R512	✗	✓	400	400	2.5	–	18
R128 ‡	✓	✗	–	–	2.4	2.7	9
R512 *	✓	✗	–	–	2.7	2.6	72
R512 *	✓	✗	–	–	2.4	2.4	168
R512	✓	✓	50	50	2.9	2.7	18
R512 ‡	✓	✓	50	75	2.5	2.4	24

Table 2. Test metrics for models trained on all datasets. Each row is a variant with resolution R128 or R512, trained with/without velocity and with/without PMRT. We report pre-training and fine-tuning epochs, group metrics, and training time on a single NVIDIA H100 GPU. The best models selected for the downstream tables are marked ‡; rows marked * are R512 models without PMRT that exceed the 24 h budget. The PMRT R512 models compared to baseline have similar or better accuracy at significantly lower training costs.

$$\text{Rel. L2 Error} = \left(\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \frac{\|\mathbf{u}^i - \hat{\mathbf{u}}^i\|_2}{\|\mathbf{u}^i\|_2} \right) \times 100 [\%] \quad (2)$$

where n_{test} is the number of test samples; \mathbf{u}^i and $\hat{\mathbf{u}}^i$ are the true and predicted velocity field vectors, respectively.

4. Results

All models in this study are trained on a single NVIDIA H100 GPU. We start with c_d -only models, followed by c_d +velocity field models trained on all the datasets. Finally, we compare our models trained only on DrivAerML with literature baselines. For PMRT, we always use a probability floor of 0.1 with 10 warm-up epochs. Extended subgroup test metrics are reported in the supplementary material.

4.1. Model predicting only drag coefficient

From the results in Tab. 2, we can see that the baseline R128 model achieves a group c_d MAE of 3.4 drag counts with a wall-clock time of 3 h. Training the model at R512 without PMRT reduces the error by 0.4 drag counts, with an 8 times increased training time of 24 h. Extending the training does not improve accuracy.

With PMRT, the cheapest R512 model attains a group c_d MAE of 2.8 drag counts in 5 h, outperforming all baselines with only a 2 h increase in training cost. We ablate the pre-training epochs (200, 400) and fine-tuning epochs (50, 400). The model with the best balance of accuracy and compute cost is the PMRT R512 model with 400 pre-training and 50 fine-tuning epochs, which lowers MAE by 26 % vs R128

and 17 % vs R512 without PMRT while training three times faster (8 h). The PMRT R512 models with 400 fine-tuning epochs do not improve accuracy beyond 2.5 drag counts, but incur substantially higher training costs (16 h to 18 h).

Pre-training and fine-tuning epochs are hyperparameters that should be chosen based on experimentation. In our experiments, longer pre-training with a short fine-tuning achieved the same accuracy as longer fine-tuning, with lower wall-clock time. For a fixed budget, shifting epochs to pre-training is more compute-efficient.

4.2. Model predicting drag coefficient and velocity field

The baseline R128 c_d + velocity model (correlation plot in Fig. 4, left & middle) attains a group c_d MAE of 2.4 drag counts and a group relative L2 error of 2.7 % with a wall-clock time of 9 h. We attribute the improvement compared to R128 c_d -only model to additional supervision from jointly predicting c_d and velocity fields, which increases the information available during training. Without PMRT, R512 baselines do not reach a good solution within 24 h. When trained for 3 days, the model achieves a group c_d MAE of 2.7 drag counts; extending the training time to 7 days finally matches the R128 c_d MAE and improves the group relative L2 error by 11 % at $\approx 19\times$ the training time of R128. The best PMRT R512 model achieves a group c_d MAE of 2.5 drag counts and a group relative L2 error of 2.4 % in 24 h, delivering similar accuracy to the R512 without PMRT model while being $7\times$ cheaper to train.

For c_d prediction, the R128 c_d + velocity baseline model outperforms the R128 c_d -only baseline. While the PMRT

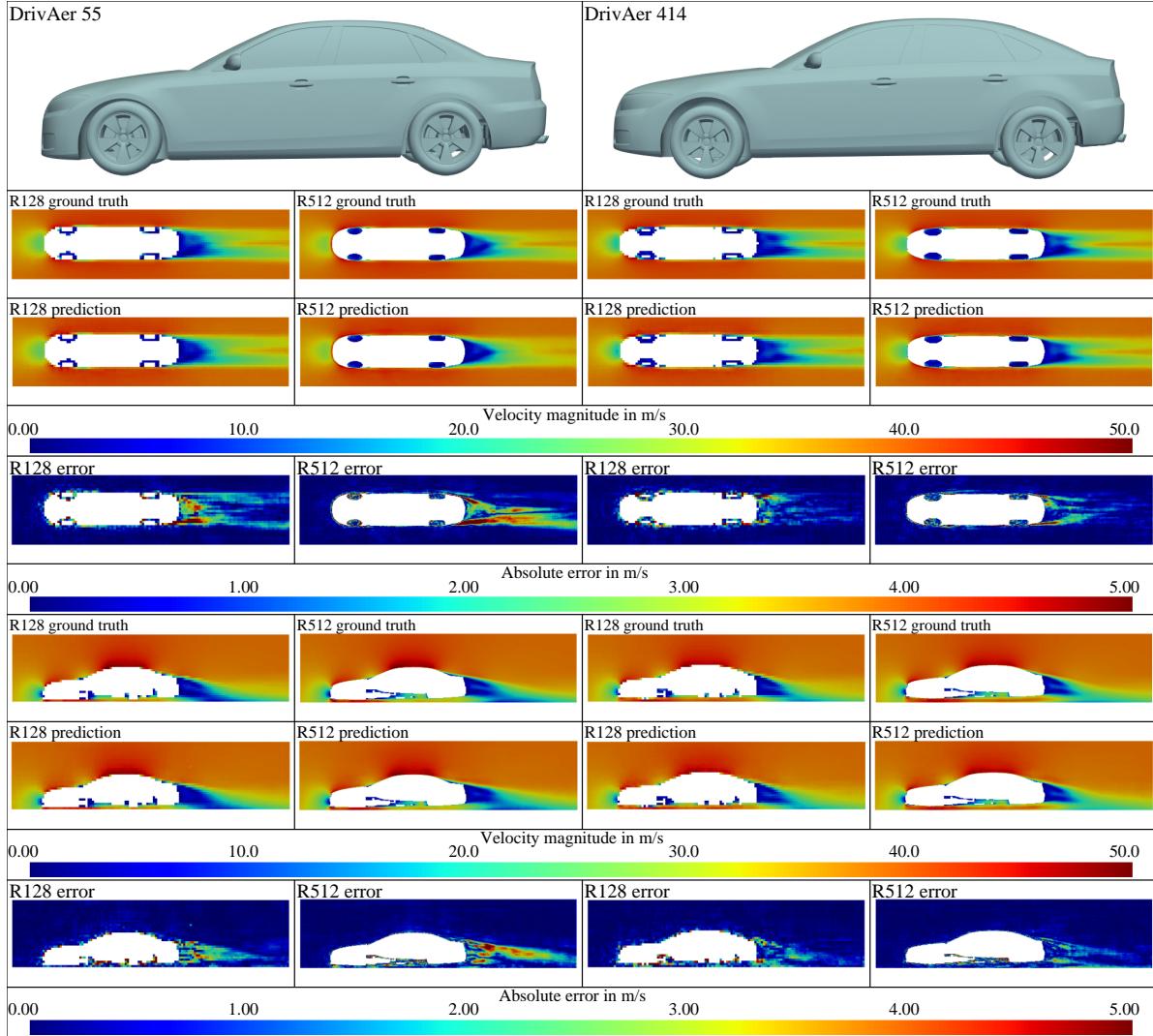


Figure 3. Comparison of true and predicted velocity field slices between R128 and PMRT R512 models for two DrivAerML samples.

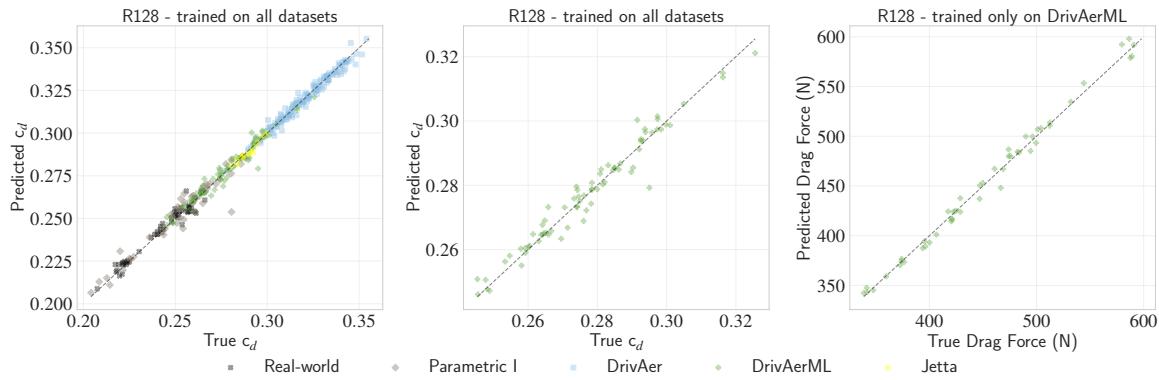


Figure 4. Correlation plots for R128 models predicting c_d and velocity: (1) c_d on all datasets; (2) c_d on DrivAerML (model trained on all datasets); (3) drag force on DrivAerML with a model trained only on DrivAerML using benchmark split.

Method	Velocity (✓/✗)	$c_d: R^2$ score ↑	Drag force: R^2 score ↑	Velocity: relative L2 error [%] ↓ *	NVIDIA H100 GPU-hours
AB-UPT †	✓	0.976	0.992	5.99	7
DoMINO	✓	—	0.984	—	184 ‡
FIGConvNet	✗	—	0.975	—	—
X-MeshGraphNet	✓	—	0.921	—	—
Ours: R128	✗	0.952 ± 0.014	0.986 ± 0.004	—	1
Ours: R128	✓	0.970 ± 0.003	0.991 ± 0.001	3.7 ± 0.1	1
Ours: PMRT R512	✗	0.975 ± 0.002	0.993 ± 0.001	—	1
Ours: PMRT R512	✓	0.974 ± 0.002	0.992 ± 0.001	2.4 ± 0.1	4

Table 3. DrivAerML benchmark split evaluation [39]: comparison with literature baselines. To keep the evaluation fair, our models are trained only on DrivAerML. We report $c_d R^2$, drag force R^2 , velocity relative L2 error [%], and compute cost in GPU-hours ($1 \times$ NVIDIA H100). Notes: “Ours” shows mean \pm std over five runs; † AB-UPT shows the median over five runs for which the author provided the metrics; ‡ GPU-hours for DoMINO were provided by the authors; * the metric is not comparable across methods due to different prediction regions; all other values are reproduced from the benchmark study [39] where some of them are not reported.

R512 c_d -only model and the R128 c_d + velocity baseline model have similar c_d accuracy with similar wall-clock time (8 and 9 h); the choice depends on whether fields are needed. If fields are unnecessary or unavailable, training the PMRT R512 c_d -only is a viable option.

By visually comparing R128 and PMRT R512 velocity field predictions in Fig. 3, we observe finer spatial detail captured by the R512 model. For example, even in the R128 ground truth, some details are lost in the wake, which are present in the R512 ground truth, and many of these details are also captured by the R512 model’s prediction. This presents a clear trade-off: we recommend PMRT R512 model, as it provides higher-detail velocity fields with lower relative L2 error. However, with limited compute, the R128 model remains a practical choice for accurate c_d with coarse velocity field prediction. For both models, errors are concentrated in similar regions: near the vehicle surface and in the wake.

We observe that the c_d MAE plateaus at 2.4 drag counts, achieved by the R128 c_d + velocity model; we hypothesize a few reasons for the lack of c_d prediction improvement in the R512 models: (1) a fundamental limit due to simulation noise that can distort physical trends; for example, the c_d and velocity fields are time-averaged; the DrivAerML simulations were set up for a statistical accuracy of ± 1.5 drag counts; this is already close to what our models and other SOTA models achieve; (2) architectural limitations: we use a U-Net that has been shown to work for low resolution aerodynamic applications, further accuracy improvements might need architectural improvements; (3) despite its coarseness, R128 may still encode sufficient geometric information for accurate c_d prediction.

4.3. Benchmarking on DrivAerML dataset

For a fair comparison with existing models, we train our models only on the DrivAerML dataset using the split used in the benchmark study by Tangsali *et al.* [39]. The results can be seen in Tab. 3, some metrics are not available because the literature baselines did not report them. We train our model five times with different seeds and report the mean and standard deviation of the metrics. Among the literature baselines, AB-UPT has the best accuracy, followed by DoMINO. Except for the R128 c_d -only model, our c_d and drag force R^2 scores are similar to AB-UPT. Figure 4 (right-most) shows our model captures the trend well. For the velocity field, our model has a lower relative L2 error compared to AB-UPT. However, the relative L2 error metrics are not comparable because the methods predict the fields in different regions, and our method predicts on a Cartesian grid; the literature baselines predict on unstructured grids.

Training on DrivAerML takes ≈ 4 h for the PMRT R512 c_d + velocity field model and ≈ 1 h for others, compared to 7 h for AB-UPT. While conventional CNN-based models, like ours, have limitations (e.g., the inability to directly predict surface fields and dependence on resolution), methods such as AB-UPT and DoMINO address these. Despite these limitations, our models strike a balance due to their lower compute cost, making them suitable for scenarios requiring short time-to-first prediction on new datasets or with limited compute or when fields are unavailable. Moreover, we have shown that our models can scale to multi-dataset scenarios using different solvers while keeping the compute cost reasonable.

5. Conclusion

In this study, we address the problem of high computational cost in training CNN-based surrogates at high resolutions

for automotive aerodynamics. We present PMRT, a probabilistic multi-resolution training schedule that enables high-resolution training at substantially lower cost without significant architectural changes. With PMRT, we can train a U-Net on five datasets (≈ 1900 samples) for high-resolution field prediction in 24 h on a single NVIDIA H100 GPU, up to $7\times$ cheaper than our high resolution without PMRT model, while matching or improving accuracy. We demonstrate that conditioning with simulation parameters enables us to train a single model on datasets from different solvers, including a real-world dataset. Our models also achieve similar accuracy compared to literature baselines at a fraction of the training cost on the DrivAerML dataset. Since PMRT is a training methodology, we leave its evaluation on other high-resolution-focused backbones to future work.

Acknowledgments

The authors acknowledge the assistance of colleagues who collected the dataset and helped us throughout the study.

Disclaimer

The results, opinions and conclusions expressed in this publication are not necessarily those of Volkswagen Aktiengesellschaft.

References

- [1] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. VATT: transformers for multimodal self-supervised learning from raw video, audio and text. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2021. Curran Associates Inc. [2](#)
- [2] Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schonwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, Burkhard Hupertz, and Danielle Maddix. DrivAerML: High-Fidelity Computational Fluid Dynamics Dataset for Road-Car External Aerodynamics, 2025. [2, 3, 5](#)
- [3] Benedikt Alkin, Maurits Bleeker, Richard Kurle, Tobias Kronlachner, Reinhard Sonnleitner, Matthias Dorfer, and Johannes Brandstetter. AB-UPt: Scaling Neural CFD Surrogates for High-Fidelity Automotive Aerodynamics Simulations via Anchored-Branched Universal Physics Transformers, 2025. [1, 2, 5](#)
- [4] Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. FlexiViT: One Model for All Patch Sizes . In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14496–14506, Los Alamitos, CA, USA, 2023. IEEE Computer Society. [2](#)
- [5] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019. [1](#)
- [6] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52(Volume 52, 2020):477–508, 2020. [1](#)
- [7] Fangge Chen and Kei Akasaka. 3D Flow Field Estimation around a Vehicle Using Convolutional Neural Networks. In *Proceedings of the 32nd British Machine Vision Conference (BMVC)*, 2021. [1](#)
- [8] Qian Chen, Mohamed Elrefaei, Angela Dai, and Faez Ahmed. TripNet: Learning Large-scale High-fidelity 3D Car Aerodynamics with Triplane Networks, 2025. [1, 2](#)
- [9] Chris Choy, Alexey Kamenev, Jean Kossaifi, Max Rietmann, Jan Kautz, and Kamyar Azizzadenesheli. Factorized Implicit Global Convolution for Automotive Computational Fluid Dynamics Prediction, 2025. [1, 2, 5](#)
- [10] Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim Alabdulmohsin, Avital Oliver, Piotr Padlewski, Alexey A. Gritsenko, Mario Lucic, and Neil Houlsby. Patch n’ pack: Navit, a vision transformer for any aspect ratio and resolution. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2023. Curran Associates Inc. [2](#)
- [11] Mohamed Elrefaei, Florin Morar, Angela Dai, and Faez Ahmed. DrivAerNet++: a large-scale multimodal car dataset with computational fluid dynamics simulations and deep learning benchmarks. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2025. Curran Associates Inc.
- [12] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional Neural Networks for Steady Flow Approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 481–490, New York, NY, USA, 2016. ACM. [1](#)
- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034. IEEE, 2015. [4](#)

- [14] Angelina I. Heft, Thomas Indinger, and Nikolaus A. Adams. Introduction of a New Realistic Generic Car Model for Aerodynamic Investigations. In *SAE 2012 World Congress & Exhibition*. SAE International, 2012. [3](#)
- [15] Jeremy Howard, Sylvain Gugger, and Soumith Chintala. *Deep learning for coders with fastai and PyTorch: AI applications without a PhD*. O'Reilly, Beijing and Boston and Farnham and Sebastopol and Tokyo, first edition edition, July 2020. [2](#)
- [16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7132–7141. IEEE, 2018. [4](#)
- [17] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep Networks with Stochastic Depth. In *Computer Vision – ECCV 2016*, pages 646–661, Cham, 2016. Springer International Publishing. [5](#)
- [18] M. Islam, F. Decker, E. de Villiers, A. Jackson, J. Gines, T. Grahs, A. Gitt-Gehrke, and J. Comas i Font. Application of Detached-Eddy Simulation for Automotive Aerodynamics Development. In *SAE World Congress & Exhibition*. SAE International, 2009. [3](#)
- [19] Sam Jacob Jacob, Markus Mrosek, Carsten Othmer, and Harald Köstler. Deep Learning for Real-Time Aerodynamic Evaluations of Arbitrary Vehicle Shapes. *SAE International Journal of Passenger Vehicle Systems*, 15(2), 2022. [1, 4, 5](#)
- [20] Sam Jacob Jacob, Markus Mrosek, Carsten Othmer, and Harald Köstler. Benchmarking convolutional neural network and graph neural network based surrogate models on a real-world car external aerodynamics dataset. *Computers & Fluids*, 300:106760, 2025. [1, 4](#)
- [21] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. [2](#)
- [22] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3D PDEs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2023. Curran Associates Inc. [1](#)
- [23] Mario Lino, Stathi Fotiadis, Anil A. Bharath, and Chris D. Cantwell. Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 479(2275), 2023. [1](#)
- [24] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin Transformer V2: Scaling Up Capacity and Resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11999–12009, 2022. [2](#)
- [25] Markus Mrosek, Carsten Othmer, and Rolf Radespiel. Reduced-Order Modeling of Vehicle Aerodynamics via Proper Orthogonal Decomposition. *SAE International Journal of Passenger Cars - Mechanical Systems*, 12(3):225–236, 2019. [3](#)
- [26] Markus Mrosek, Carsten Othmer, and Rolf Radespiel. *Variational Autoencoders for Model Order Reduction in Vehicle Aerodynamics*. 2021. [3](#)
- [27] Ken Museth, Jeff Lait, John Johanson, Jeff Budsberg, Ron Henderson, Mihai Alden, Peter Cucka, David Hill, and Andrew Pearce. OpenVDB: an open-source data structure and toolkit for high-resolution volumes. In *ACM SIGGRAPH 2013 Courses*, New York, NY, USA, 2013. Association for Computing Machinery. [3](#)
- [28] Mohammad Amin Nabian, Chang Liu, Rishikesh Ranade, and Sanjay Choudhry. X-MeshGraphNet: Scalable Multi-Scale Graph Neural Networks for Physics Simulation, 2024. [1, 2, 5](#)
- [29] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. [1](#)
- [30] Pratip Rana, Timothy M. Weigand, Kevin R. Pilkiewicz, and Michael L. Mayo. A scalable convolutional neural network approach to fluid flow prediction in complex environments. *Scientific reports*, 14(1):23080, 2024. [1](#)
- [31] Rishikesh Ranade, Mohammad Amin Nabian, Kausubh Tangsali, Alexey Kamenev, Oliver Hennigh, Ram Cherukuri, and Sanjay Choudhry. DoMINO: A Decomposable Multi-scale Iterative Neural Operator for Modeling Large Scale Engineering Simulations, 2025. [1, 2, 5](#)
- [32] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an Open Source Differentiable Computer Vision Library for PyTorch . In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3663–3672, Los Alamitos, CA, USA, 2020. IEEE Computer Society.
- [33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. [4](#)

- [34] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. Concurrent Spatial and Channel ‘Squeeze & Excitation’ in Fully Convolutional Networks. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, pages 421–429. Springer International Publishing, Cham, 2018. [4](#)
- [35] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to Simulate Complex Physics with Graph Networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020. [1](#)
- [36] Bharat Singh, Mahyar Najibi, and Larry S. Davis. SNIPER: efficient multi-scale training. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 9333–9343, Red Hook, NY, USA, 2018. Curran Associates Inc. [2](#)
- [37] Leslie N. Smith. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017. [5](#)
- [38] Mingxing Tan and Quoc Le. EfficientNetV2: Smaller Models and Faster Training. In *Proceedings of the 38th International Conference on Machine Learning*, pages 10096–10106. PMLR, 2021. [2](#)
- [39] Kaustubh Tangsali, Rishikesh Ranade, Mohammad Amin Nabian, Alexey Kamenev, Peter Sharpe, Neil Ashton, Ram Cherukuri, and Sanjay Choudhry. A Benchmarking Framework for AI models in Automotive Aerodynamics, 2025. [1, 5, 8](#)
- [40] The Mosaic ML Team. streaming. <<https://github.com/mosaicml/streaming/>>, 2022.
- [41] Rahul Thapa, Kezhen Chen, Ian Covert, Rahul Chalamala, Ben Athiwaratkun, Shuaiwen Leon Song, and James Zou. Dragonfly: Multi-Resolution Zoom-In Encoding Enhances Vision-Language Models, 2024. [2](#)
- [42] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows. *AIAA Journal*, 58(1):25–36, 2020. [1](#)
- [43] Rui Tian, Zuxuan Wu, Qi Dai, Han Hu, Yu Qiao, and Yu-Gang Jiang. ResFormer: Scaling ViTs with Multi-Resolution Training . In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22721–22731, Los Alamitos, CA, USA, 2023. IEEE Computer Society. [2](#)
- [44] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jegou. Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. [2](#)
- [45] Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, and Hervé Jégou. Three things every one should know about vision transformers. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, page 497–515, Berlin, Heidelberg, 2022. Springer-Verlag. [2](#)
- [46] Thanh Luan Trinh, Fangge Chen, Takuya Nanri, and Kei Akasaka. 3D Super-Resolution Model for Vehicle Flow Field Enrichment. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5814–5823, 2024. [2, 5](#)
- [47] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3349–3364, 2021. [2](#)
- [48] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: octree-based convolutional neural networks for 3D shape analysis. *ACM Trans. Graph.*, 36(4), 2017. [2](#)
- [49] Francis Williams, Jiahui Huang, Jonathan Swartz, Gergely Klar, Vijay Thakkar, Matthew Cong, Xuanchi Ren, Rui long Li, Clement Fuji-Tsang, Sanja Fidler, Eftychios Sifakis, and Ken Museth. fVDB : A Deep-Learning Framework for Sparse, Large Scale, and High Performance Spatial Intelligence. *ACM Transactions on Graphics*, 43(4):1–15, 2024. [2](#)
- [50] Yuxin Wu and Kaiming He. Group Normalization. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part XIII*, pages 3–19, Berlin, Heidelberg, 2018. Springer-Verlag. [4](#)
- [51] Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive Tokens for Efficient Vision Transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10799–10808, 2022. [2](#)

Supplementary

A. Comparison of datasets

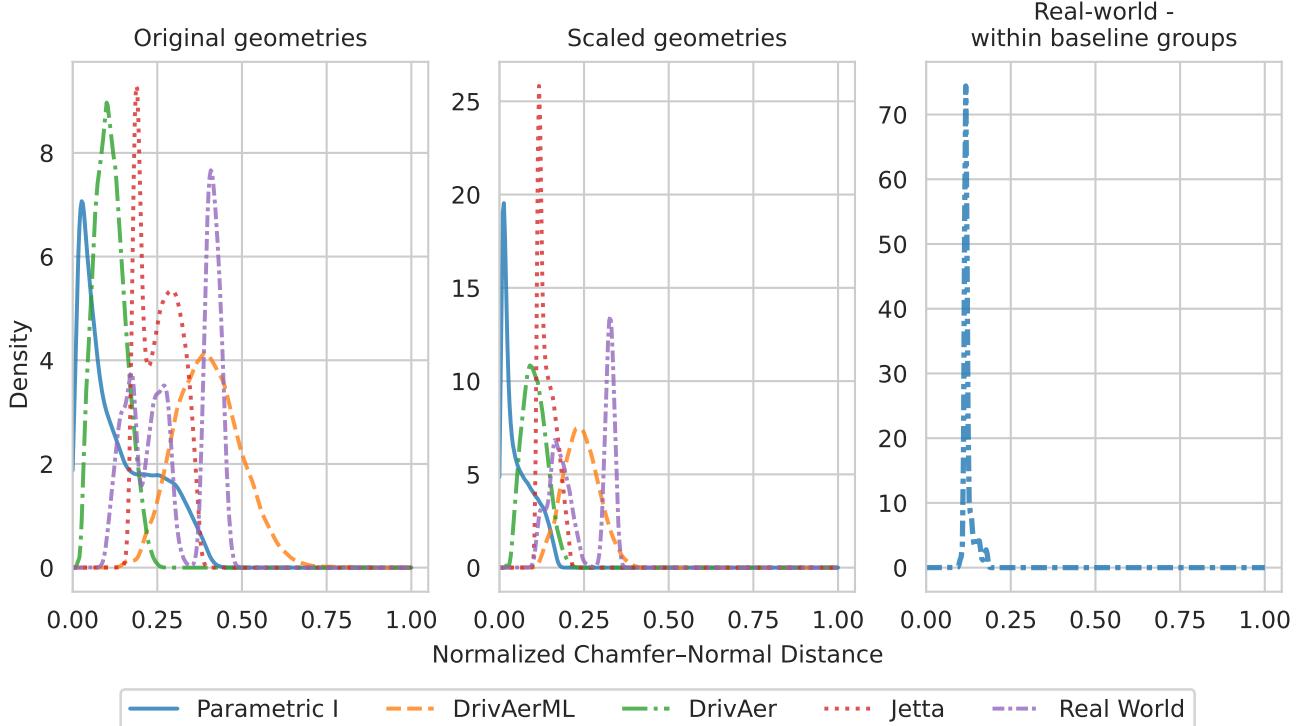


Figure 5. Normalized Chamfer–Normal Distance (NCND) distribution of pairwise samples within a dataset. A larger NCND distance indicates a larger difference between the geometries. The plots compare the NCND distribution in order from left to right: (1) aligned geometries: geometries translated to a common reference; (2) scaled geometries: geometries that are scaled to a unit box to reduce the effect of proportional changes; (3) pairwise distances of real-world dataset calculated only with baseline groups.

Currently, the only publicly available high-fidelity aerodynamics dataset is the DrivAerML [2] dataset. Although DrivAerNet++ [11] is a large aerodynamics dataset, its simulations are based on Reynolds-Averaged Navier-Stokes (RANS), which carry higher errors compared to the standard high-fidelity simulations in the industry. In this study, we do not evaluate DrivAerNet++ due to its simulation method and licensing restrictions. Both datasets are parametric and have greatly helped improve aerodynamic surrogates. However, we currently lack a publicly accessible aerodynamics dataset that imitates real-world datasets. We compare the datasets used to highlight some of the differences between parametric and real-world datasets.

We compare datasets using the Normalized Chamfer–Normal Distance (NCND), which combines a normalized pointwise Chamfer distance with nearest-neighbor normal cosine dissimilarity. Specifically, we calculate $\text{chamfer_distance}/x + y * \text{cosine_dissimilarity}$ between all pairs of geometries within a dataset. The constants x and y are chosen so the maximum possible normalized value is one and the largest contribution from each component is 0.5; the same constants are used across all plots for comparability. We compute the pairwise NCND distances within each dataset on 100,000-point clouds per geometry sampled via farthest point sampling to keep the computational cost reasonable. We calculate and report the distances in two settings: (i) Aligned geometries - meshes are first translated so that their centroid is at the origin and then shifted along the z-axis so that their minimum z is zero so that all models share a common ground plane; (ii) Scaled geometries - meshes are scaled to fit a unit cube, reducing proportional/placement effects and emphasizing feature-level differences. The scaled geometries tend to have major features (like wheels and mirrors) in similar locations.

Figure 5 shows the pairwise NCND distributions. The original real-world dataset geometries cover a broad span with significant irregularities. In comparison, although some parametric datasets, such as parametric I and DrivAerML, have a large span, they exhibit significantly lower irregularities compared to the real-world dataset. After scaling, average distances,

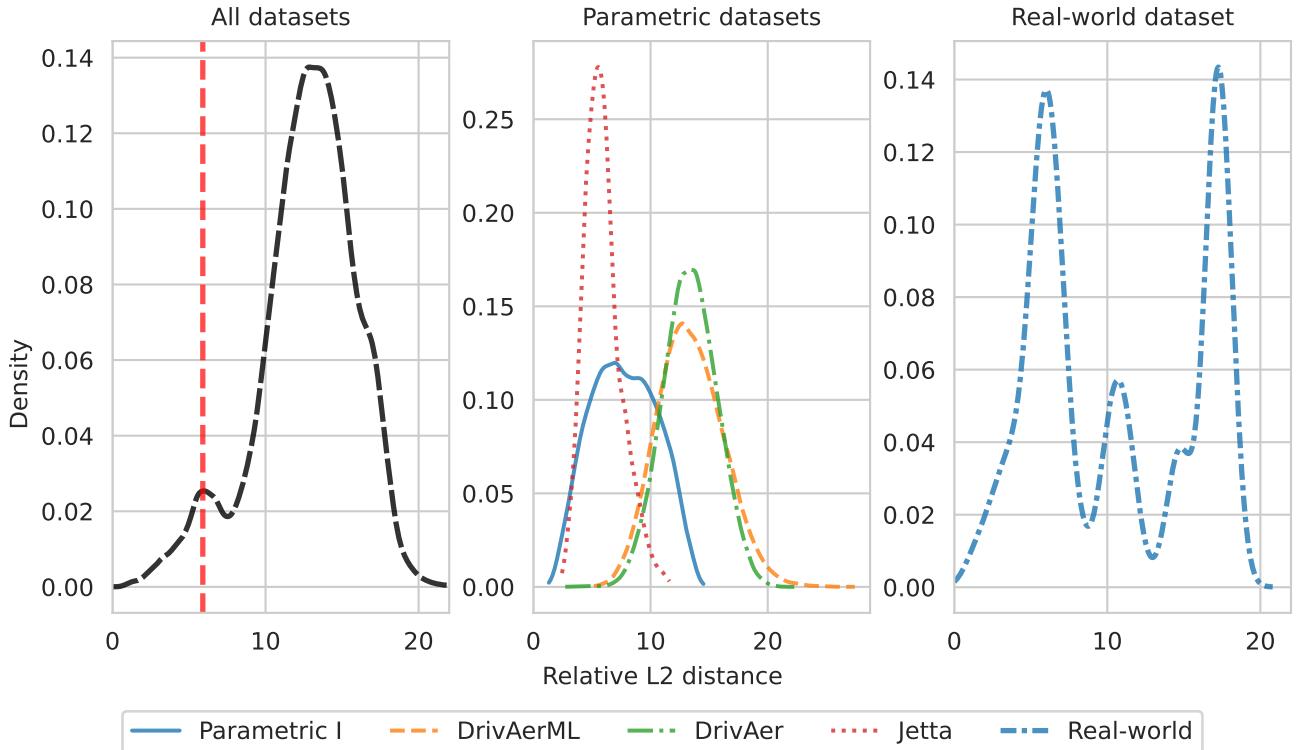


Figure 6. Distribution of pairwise relative L2 distance between time-averaged velocity fields. The plots, left to right, show: (1) all datasets combined: with per-dataset distributions computed from all within-dataset pairs; the vertical red line marks the error that would be obtained by a nearest-neighbor baseline that, for each test sample, predicts the velocity field of its closest training sample; (2) only parametric datasets (3) only the real-world dataset. Larger values indicate larger flow-field differences.

range, and irregularities drop across all datasets. For all parametric datasets except the DrivAer dataset (mean reduction: 6%; span reduction: 14%), the mean drops by at least 40% and the span by at least 48%. Real-world also narrows (mean reduction: 20%; span reduction: 28%), but relative to the original geometry, its distribution stays the broadest and most irregular. This indicates that the real-world datasets have relatively more geometric changes that are not due to scaling or proportional changes. From the distribution calculated only for the samples within baseline groups (a group of geometries created from a single baseline geometry) for the real-world datasets, distances concentrate in a narrow region, indicating that most variation arises across baseline geometries rather than within a single baseline. This is shown to highlight that a majority of the changes in real-world geometries are targeted (as they do not significantly affect the NCND), specific changes that still considerably affect the c_d . The larger and significant proportional changes occur between the baseline geometries. Currently, all publicly available aerodynamic datasets are primarily parametric, and their diversity is driven mainly by large changes in a few geometric features. These datasets do not have the numerous diverse small geometric feature changes present in real-world datasets.

In Fig. 6 we compare the velocity fields by analyzing the distribution of the pairwise relative L2 distance computed within each dataset. Across all datasets, the distribution has a wide span. The distribution for all the parametric datasets yields a roughly normal distribution with varying means and standard deviations. In contrast, the real-world has a larger span with more irregularities and multiple peaks, underscoring that real-world flow fields vary more heterogeneously than the parametric datasets.

B. Implementation details

B.1. PMRT

PMRT consists of three phases: warm-up, pre-training, and fine-tuning. During warm-up, the probabilities are linearly interpolated from an equal distribution to the initial pre-training probabilities. During fine-tuning, the model trains exclusively

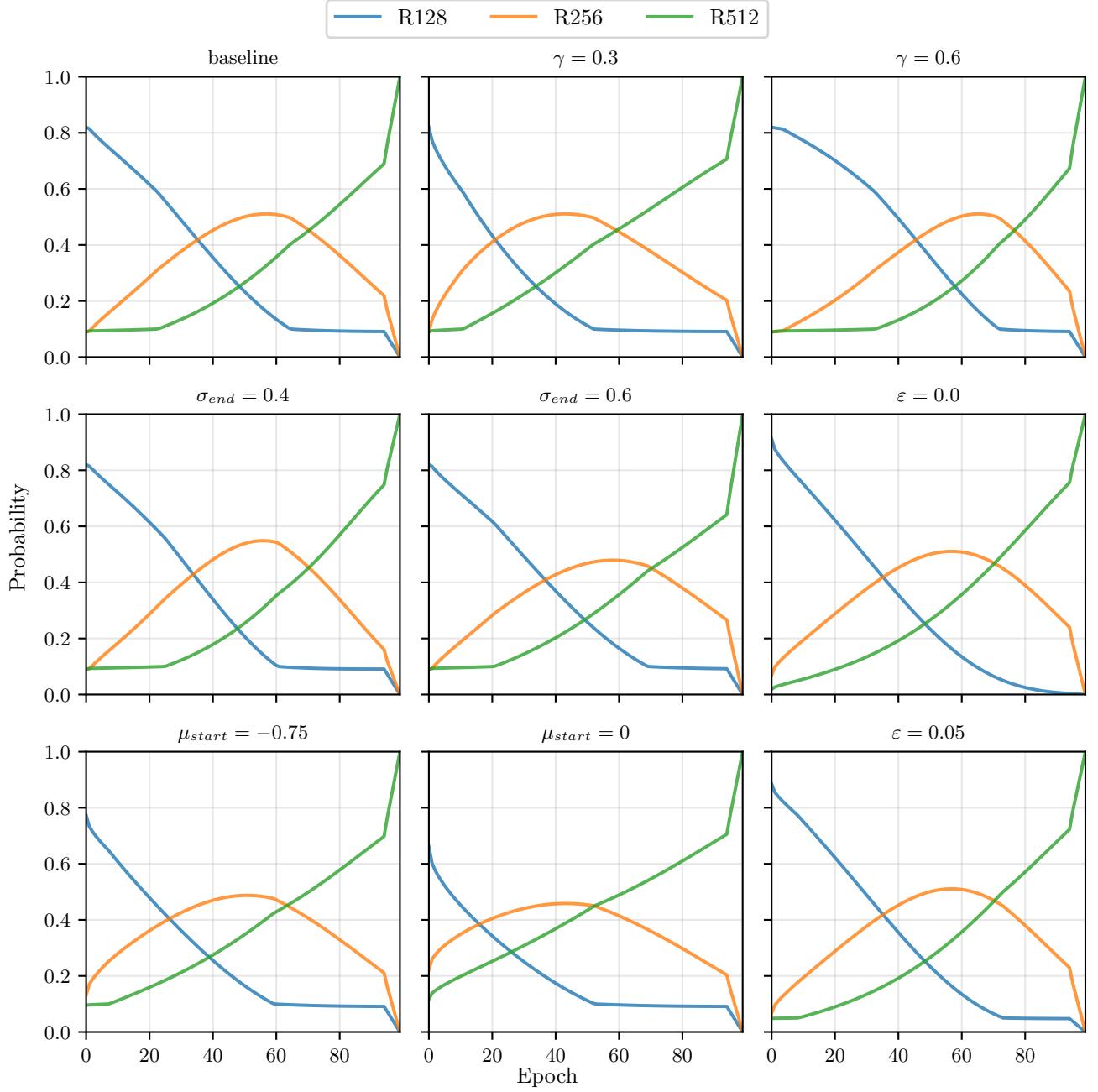


Figure 7. PMRT schedule some hyperparameters variations. Each subplot shows the sampling probabilities for three resolutions across the pre-training epochs with the subplot title showing the parameter that is changed relative to the baseline. The baseline configuration (top-left) uses $\gamma=0.45$, $\sigma_{end}=0.5$, $\epsilon=0.1$, and $\mu_{start}=-1.5$.

on the highest resolution, R512. The pre-training schedule is generated by discretizing a Gaussian distribution over the resolution indices, where over the pre-training epochs the mean moves from low toward higher resolutions while the standard deviation shrinks. Here, we provide the detailed implementation of the pre-training phase.

The pre-training schedule operates over a set of R resolution levels, indexed by $r \in 0, \dots, R-1$. In our case, $R = 3$, corresponding to resolutions R128 (0), R256 (1), and R512 (2). The pre-training phase consists of E epochs, indexed by $e \in 0, \dots, E-1$. We first normalize the epoch index to the range $[0, 1]$:

$$e_n = \frac{e}{E - 1}. \quad (3)$$

The schedule is driven by mean (μ_e) and standard deviation (σ_e) that varies over epochs, and the transition phase is controlled by γ :

$$\mu_e = \mu_{\text{start}} + (\mu_{\text{end}} - \mu_{\text{start}}) e_n^\gamma, \quad (4)$$

$$\sigma_e = \sigma_{\text{start}} - (\sigma_{\text{start}} - \sigma_{\text{end}}) e_n^\gamma. \quad (5)$$

$$\mu_{\text{start}} = -\frac{R}{2}, \quad \mu_{\text{end}} = R - 1, \quad \sigma_{\text{start}} = \frac{R}{2}, \quad \sigma_{\text{end}} = 0.5. \quad (6)$$

The hyperparameters μ_{start} , μ_{end} , σ_{start} , σ_{end} and γ control the schedule. In our case, the mean moves from (-1.5 → 2) and the μ_{start} is set to $-R/2$, which gives a strong initial emphasis to the lowest resolution (R128). As the training progresses, the σ shrinks and μ moves towards emphasizing the highest resolution. We use a γ of 0.45 and σ_{end} of 0.5 to ensure a balanced mixing of the resolutions. For example, with $E = 200$ pre-training epochs, this leads to an overall sampling ratio of approximately 33%, 35%, and 32% for resolutions R128, R256, and R512. In our case, a higher γ would give more emphasis to the lowest resolution and a lower γ to the highest resolution. Similarly, a lower σ_{end} would emphasize the R512 even more towards the end of the pre-training phase. Figure 7 shows how some hyperparameters affect the probabilities.

The raw probability for each resolution, $p_{e,r}^{\text{raw}}$, is derived from the standard normal cumulative distribution function (CDF), $\Phi(\Phi(x) = \frac{1}{2}(1 + \text{erf}(x/\sqrt{2}))$, as formulated in Eqs. (7) to (9). In Eqs. (10) and (11), the raw probabilities are clipped by a minimum value ε (0.1) and normalized (sum to one) to prevent the probability of any resolution from getting too low.

$$c_{e,0} = 0, \quad (7)$$

$$c_{e,r+1} = \Phi\left(\frac{(r + \frac{1}{2}) - \mu_e}{\sigma_e}\right) \quad (8)$$

$$p_{e,r}^{\text{raw}} = c_{e,r+1} - c_{e,r} \quad (9)$$

$$\tilde{p}_{e,r} = \max(p_{e,r}^{\text{raw}}, \varepsilon) \quad (10)$$

$$p_{e,r} = \frac{\tilde{p}_{e,r}}{\sum_{q=0}^{R-1} \tilde{p}_{e,q}} \quad (11)$$

During the final T epochs (5 epochs) of the pre-training phase (from epoch E-T to E-1), the sampling probabilities are linearly interpolated towards the fine-tuning probabilities p_r^{fine} , that select only the highest resolution:

$$p_{E-T+t, r} = \left(1 - \frac{t}{T-1}\right) p_{E-T, r} + \frac{t}{T-1} p_r^{\text{fine}}, \quad t = 0, \dots, T-1.$$

Finally, during training, at each batch of a given epoch, a resolution is drawn from the distribution defined by the probabilities for that epoch:

$$r_e \sim \text{Categorical}(p_{e,0}, p_{e,1}, \dots, p_{e,R-1}) \quad (12)$$

B.2. Performance

One performance bottleneck we face is loading high-resolution data and applying augmentation. To mitigate these bottlenecks, we use MosaicML Streaming [40] for data loading and Kornia [32] for data augmentation.

All our models were trained using Float32. We have not experimented with mixed precision. All our models, excluding the PMRT R512 c_d + velocity model, do not saturate the GPU; for these models, we do not expect any significant performance gains using mixed precision except for the reduction in memory usage. It might be possible to train R512 PMRT c_d + velocity models at a lower compute cost with mixed precision.

Variant (relative to baseline)	Group: c_d MAE ↓	Group: Relative L2 error ↓
Base: R128- c_d +velocity; 3 decoders	2.4	2.7
Single velocity field decoder	2.5	3.3
No SDF to output convolution skip connection	2.7	2.8
Without multi-head self-attention and transformer	2.7	2.7

Table 4. Ablation of R128 c_d + velocity model. Rows indicate a changed feature, and we report the group c_d MAE and group relative L2 error.

Model	Group: c_d MAE ↓
Best PMRT R512	2.5
PMRT R512: no probability floor	2.6
PMRT R512: no warm-up	2.7
PMRT R512: no fine-tuning on R512	3.0
PMRT R512: low resolution batch size multiplier of 2	2.8
PMRT R512: no low resolution batch size multiplier	3.0
Naive pre-train→fine-tune (hard switch)	3.3
Constant equal probability schedule	3.6

Table 5. Ablation of the PMRT schedule for R512 c_d -only model. The best model uses a probability floor of 0.1, and a batch size multiplier of 4 for lower resolutions. Rows indicate the change compared to the best model. We report the group c_d MAE.

B.3. Region of interest (ROI)

For the DrivAerML dataset, we extract the velocity fields in an ROI of size $(L_x, L_y, L_z) = (9.28 \text{ m}, 3.84 \text{ m}, 2.66 \text{ m})$. The ROI is anchored relative to the vehicle and CFD domain. The ROI starts 1.0 m upstream of the vehicle, is laterally centered on the vehicle, and extends 2.66 m vertically from the lower bound of the original CFD field (road plane). The R512 cell size ($\approx 0.0181 \text{ m} \times 0.03 \text{ m} \times 0.0208 \text{ m}$) is comparable to or finer than the original CFD cells except for some regions close to the vehicle’s surface and in the wake of the car. While the CFD requires locally finer cells for a stable and accurate unsteady simulation, we find R512 sufficient for predicting and visualizing time-averaged velocity fields.

C. Extended results

C.1. Ablations: architecture

Table 4 shows the ablations of the major architectural design choices. We compare all variants with the baseline R128 c_d + velocity model. When we replace the three velocity field decoders with a single decoder that predicts all three components, there is a negligible increase in the c_d MAE but a significant increase in the relative L2 error. Removing the SDF to output convolution skip connection increases the c_d MAE by 0.3 drag counts and the group relative L2 error by 0.1 drag counts. Removing the multi-head self-attention and transformer layers has no impact on the velocity field prediction but increases c_d MAE by 0.3 drag counts.

C.2. Ablations: PMRT

Table 5 reports ablations of the PMRT schedule for the R512 c_d -only model, with all variants compared against the best PMRT R512 model. Removing the probability floor or the warm-up has a minor increase in the c_d MAE of 0.1 and 0.2 drag counts, respectively. Skipping the fine-tuning phase has a much larger increase of 0.5 drag counts on the c_d MAE; as fine-tuning specializes the model to R512. Reducing the low-resolution batch multiplier to $\times 2$ increased the group c_d MAE by 0.3, and removing it entirely increases it by 0.5. The low-resolution batch size multiplier allows the model to see more low-resolution samples and improves the GPU utilization; the R512 batch size is memory-bound, so using the same batch size at lower resolutions underutilizes the GPU. In the naive pre-train→ fine-tune, we first train on R128, then R256, and

Dataset group	Metric	Baseline R128		Best PMRT R512	
		Without velocity	With velocity	Without velocity	With velocity
DrivAerML	c_d : MAE ↓	3.7	2.5	2.8	2.4
	c_d : MaxAE ↓	17.9	15.8	12.7	8.9
	c_d : R^2 score ↑	0.911	0.958	0.951	0.969
	Velocity: Relative L2 error ↓	–	2.9	–	2.2
Parametric	c_d : MAE ↓	3.2	2.3	2.2	2.5
	c_d : MaxAE ↓	39.8	26.7	23.4	34.0
	c_d : R^2 score ↑	0.982	0.991	0.992	0.988
	Velocity: Relative L2 error ↓	–	2.7	–	2.6
Real-world	c_d : MAE ↓	3.2	2.5	2.6	2.8
	c_d : MaxAE ↓	29.5	10.1	17.7	26.2
	c_d : R^2 score ↑	0.900	0.961	0.950	0.915
	Velocity: Relative L2 error ↓	–	2.4	–	2.5
wall-clock time [h]		3	9	8	24

Table 6. Extended subgroup test metrics for best models trained on all datasets. We report the MAE, MaxAE and R^2 score for c_d , and velocity relative L2 error when applicable. The final row reports wall-clock time on a single NVIDIA H100 GPU.

finally fine-tune it at R512, which increases the group c_d MAE by 0.8. The constant equal probability schedule samples all resolutions equally, which increases the group c_d MAE by 1.1 drag counts.

C.3. Best model metrics in subgroups

Table 6 reports extended subgroup metrics for all the models. We include these for completeness; the interpretation of these metrics does not differ from what was interpreted using the aggregated metrics in the main paper.