Machine Learning for Road Vehicle Aerodynamics

Vidyasagar Ananthan, Neil Ashton, Nate Chadwick, Mariano Lizarraga, Danielle C. Maddix, Satheesh Maheswaran, Pablo Hermoso Moreno, Parisa M. Shabestari, Sandeep Sovani, Shreyas Subramanian, Srinivas Tadepalli, Peter Yu
Amazon Web Services

Abstract

This paper discusses an emerging area of applying machine learning (ML) methods to augment traditional Computational Fluid Dynamics (CFD) simulations of road vehicle aerodynamics. ML methods have the potential to both reduce the computational effort to predict a new geometry or car condition and to explore a greater number of design parameters with the same computational budget. Similar to traditional CFD methods, there exists a broad range of approaches. In particular, the accuracy and computational efficiency of a CFD simulation vary greatly depending on the choice of turbulence model (DNS, LES, RANS) and the underlying spatial and temporal numerical discretizations. Similarly, the end-user must select the correct ML method depending on the use-case, the available input data, and the trade-off between accuracy and computational cost. In this paper, we showcase several case studies using various data-driven ML methods to highlight the promise of these approaches. Whilst these case studies are not comprehensive investigations of the underlying methods and do not include all possible ML approaches (i.e., physics-driven), they highlight the ability of these models to in general predict new designs in near real-time (i.e., less than 5 seconds), after typically less than 1 hour of training on a single GPU. There still exists a need for high quality training data from traditional CFD methods and high-fidelity CFD simulations to validate the ML predictions. Thus, ML approaches should be seen as tools to augment traditional CFD methods rather than to replace them. While this work focuses on preliminary studies, future work will look at more comprehensive real-world/industrial-size calculations for the more promising technologies identified here.

INTRODUCTION

Computational Fluid Dynamics (CFD) is widely used throughout the automotive industry as a key design tool, alongside wind-tunnels and road tests. It is used to design the external surfaces to achieve the required aerodynamic characteristics, i.e., performance and stability. In addition, CFD is used for aeroacoustics [1], thermal loading [2], heating, ventilation and air conditioning (HVAC) [3], soiling and contamination [4], battery design [5] to name a few. Given that CFD is so widely used throughout the automotive industry, there is a constant drive to develop new methods that can provide greater accuracy compared to wind-tunnels and road tests, and greater computational efficiency, i.e., faster and ideally at a lower computational cost [6]. Efforts such as the Automotive CFD Prediction Workshop (AutoCFD [1]) [6, 7, 8] have helped to establish the current state of the art for road-car external aerodynamics, and provided directions for academia and industry on where efforts should be focused to improve both accuracy and computational efficiency.

For almost all of the CFD methods used to date to study road car aerodynamics using finite-volume/element/differences [9, 10]), there exists a link between accuracy, speed and cost. With few exceptions, it is generally not possible to significantly increase accuracy (that is generalizable and not use-case specific) without also increasing the speed and cost. Therefore, academia and industry have in general split their efforts into one or both of the following two directions.

1. High fidelity, accurate, reproducible simulation methods, which provide low levels of error compared to physical testing. These simulations cost from thousands to tens of thousands of dollars per run and take days to weeks to run on 100-1000s of servers. Examples include hybrid Reynolds-Averaged Navier-Stokes (RANS)-Large-Eddy Simulation (HRLES), wall-modelled LES (WM-LES) of external aerodynamics over a complete road-vehicle [11] or directly resolving the aeroacoustics of a component [12].
2. Low-medium fidelity simulation methods, which allow the

[1] http://autocfd.eng.ox.ac.uk

end-user to explore thousands of designs at a lower cost compared to higher-fidelity methods. These need to have an acceptable error compared to physical tests (the best designs from this tool are typically verified using the higher-fidelity tool) and run as fast as possible, with a significantly lower cost than high-fidelity tools. Current state-of-the-art low-medium fidelity design tools still take hours on 10-100s servers at a cost of typically tens to hundreds of dollars (which is case dependent). An example includes a steady-state RANS [6] simulation of a complete road vehicle as part of a Design of Experiments (DoE) workflow exploring 100s of different options. As we discuss later, ML methods are particularly well suited to playing a similar role to these low-medium fidelity methods, given their ability to predict new designs in near real-time.

Significant work has been done to speed up both high and low-medium fidelity approaches by utilizing the latest High-Performance Computing (HPC) hardware (e.g., GPUs, arm64-based processors [13]) and offering software through the cloud to reduce HPC hardware bottlenecks [14]. In addition to improving the computational performance and efficiency of these approaches, there has been work to explore different turbulence modelling approaches [6] (e.g., WMLES, Hybrid RANS-LES) and alternative numerical algorithms e.g., high-order finite-element [15], discontinuous Galerkin (DG) methods [16], and high-order Cartesian finite differences and volumes [17]. Despite these advances, there remains a link between accuracy, cost and time to result. With novel approaches, e.g., Cartesian finite-differences based upon Navier-Stokes [17] or Lattice Boltzmann [18], if the mesh is refined to achieve greater accuracy, there is an unavoidable increase in computational cost, even if the run time itself can be kept low via access to large HPC facilities. Likewise, if reducing computational time and cost is key then accuracy is lost either through the need to move to lower fidelity methods, e.g., RANS. or due to insufficient spatial or temporal resolution.

Whilst continuing improvements in HPC software/hardware and novel algorithms lead to an increase in computational efficiency, in recent years [19] there has been a desire to study whether ML techniques can be an alternative path to achieve computational efficiency (typically ML methods do not directly improve the accuracy compared to traditional CFD methods). In general, two main approaches have emerged: data-driven and physics-driven [20].

- Data-driven: use machine learning techniques to learn the solutions or solution operator of partial differential equations (PDEs) [21], e.g., program the ML model in a supervised manner to minimize the difference between the 'true' solution and the predicted solution. The 'true' solutions can become available by using traditional solvers or by experimental measurements. Examples of these data-driven approaches that are trained on supervised simulation data include message-passing graph neural networks (GNNs) [22, 23], e.g., MeshGraphNets [24, 25, 26] and pure data-driven Neural Operators [27, 28, 29, 30].

  Whilst in these approaches the physics is not explicitly embedded into the ML model (e.g., the loss function), given sufficient training data (which is case-dependent), some suggest that the physics can be implicitly learned because the training data itself is based upon physics [31]. To date, it has not been robustly established if this is true or the amount of training data that would be required. Recent works [32, 33] have also identified failure cases at inference time.

- Physics-driven: use machine learning techniques to solve specific PDEs and study how to incorporate known physics into the learning process. One common approach includes adding the PDE as a soft constraint or regularizer to the loss function in Physics-Informed Neural Networks (PINNs) [34] and Physics-Informed Neural Operators (PINO) [35]. In [36, 37], the authors identify challenges during the optimization of these soft-constrained approaches. Recent work has studied adding these equations as hard constraints including enforcing the PDE in the network architecture [38], the integral form of conservation laws [32] and boundary conditions [33].

In this paper, we mainly focus on using data-driven methods to show the promise of offering significant speed ups in both raw prediction/inference time. At its heart is the hypothesis that this computational efficiency arises from 1) the near real-time of predicting a new design (or boundary condition), i.e., the inference step as referred to by the ML community, using modest CPU or GPU resources resulting in a cost of less than $1. 2) the cost to train the model, which use-case specific, is less than or equal to a low-medium fidelity CFD run, i.e., hours on modest GPU resources at typically a cost of less than $100. Ultimately, the break-even point of when these ML approaches are more cost efficient than traditional CFD approaches is dependent on the amount of training required and whether these simulations need to be run as part of the normal process. For example, consider the scenario when thirty simulations are to be run regardless during a road-car design project. Any further simulations can be run using a trained ML model, and then there is a strong cost savings from any simulations after the first thirty. On the other hand, if only five simulations are going to be run (which is not enough to train a model), the extra cost to run the additional twenty-five simulations to train a model may not be returned if there is not enough demand to run many more design points using the trained ML model. Whilst this paper does not answer this specific question and is the subject of on-going work, we aim to illustrate why there is such strong interest in these methods to motivate further investigations and research.

The paper is organised as follows. Firstly, we describe the datasets used throughout the work. We then focus on three case studies that look at image-based approaches using convolutional neural networks (CNNs), mesh-based using graph neural networks (GNNs) and U-Nets to predict Key Performance Indicators (KPI), i.e., lift and drag coefficients and full flow field, respectively. Lastly, our final case study examines the use of Generative AI in the design and mesh generation process. We then conclude with overall conclusions and future thoughts.

DATASETS

In order to conduct experiments on the various ML approaches, we utilize two datasets that represent simplified versions of CFD outputs used within a typical road-car aerodynamics environment whilst also representing good benchmarking datasets. Table 1 provides an overview of the dataset statistics. See Appendix A for the details on the dataset generation. It should be noted that a limitation of this papers work is that the four methods described in the paper were not run on the same dataset, as they were conducted in different points of time for different purposes. Thus whilst future work will aim to use a single dataset to provide a fairer comparison, in this work we cannot truly compare method I vs II fairly and thus the purpose of the paper is to show that there exists a range of methods rather than trying to state which one method has the best accuracy.

Case Study	Dataset	# Train / Validation	# Nodes/Sample
I - KPI	DrivAer	10 / 3	N/A
I - 2D Slices	DrivAer	10 / 3	N/A
II	DS-DrivAer*	15 / 4	$\sim$17k
III	Motorbike	463/30 (no k-fold x-val)	$\sim$2M (128^3)

* Down-sampled DrivAer dataset.

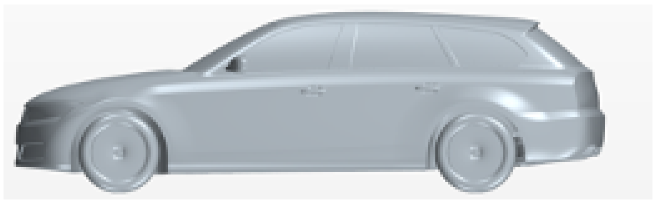
Table 1: Summary of datasets used for each case study.

DRIVAER We use the open-source DrivAer geometry [39], which thanks to the support from major automotive OEMs has grown over recent years to become the defacto open-source model for the automotive aerodynamics community. It contains realistic road vehicle geometry configurations, e.g., body styles (fastback, estate back, notchback), wheel designs, and underbody designs. Figures 1a and 1b illustrate a comparison of different body styles. For this preliminary work, we built up to 22 various body configurations and ran CFD simulations to obtain 3D flow-field outputs, 2D images outputs in the x, y and z directions as well as forces and moments. These 22 variations are listed in Appendix A, Figure 11 and are based upon the existing parts that end-users can download from the DrivAer TUM website [2]. In order to more closely match industry practices, we ran all the simulations using a hybrid RANS-LES approach with grids of up to 300M cells (see Appendix A for details of the simulation setup). It is important to note that the authors in collaboration with other groups, are at the time of writing this paper, in the process to generate a much larger dataset containing 500 variations of the DrivAer model that will help the community to study this case with much higher fidelity. This work to generate a larger dataset is part of the AutoCFD4 workshop and thus interested readers can find more at workshop website [3] as well as future publications on this topic. Considering this note, the work shown in this paper is very much preliminary in nature.

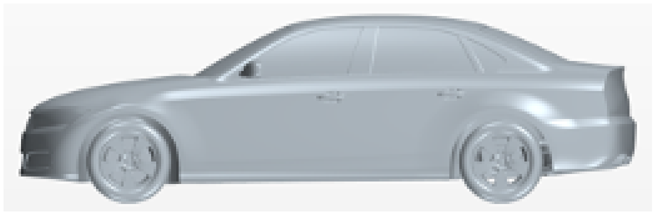
MOTORBIKE The motorbike geometry within the OpenFOAM simpleFOAM tutorial [41] has become a widely used open-source test-case to test CFD approaches. Figure 2 illustrates one base bike model. We follow the simpleFoam tutorial

[2]https://www.epc.ed.tum.de/en/aer/research-groups/automotive/drivaer/geometry/

[3]http://autocfd.eng.ox.ac.uk



(a) Estate with detailed underbody and closed tires.



(b) Notchback with detailed underbody and open wheels.

Figure 1: Different body styles in the DrivAer dataset [40].

within OpenFOAM, i.e., $k-\omega$ RANS simulation from OpenFOAM to generate the training data. It is common for industrial RANS simulations of road-cars and motorbikes to contain tens of millions of cells for accuracy purposes. In our simulations, we use a coarser grid given that the main objective is to highlight the ability of the ML model to match the CFD data, rather than requiring high-fidelity CFD data to accurately match experimental or wind-tunnel data. For the motorbike cases, each unstructured mesh is approximately 700k hex-dominated cells, which is converted to a structured grid of $\sim$ 2M (128^3) through interpolation.

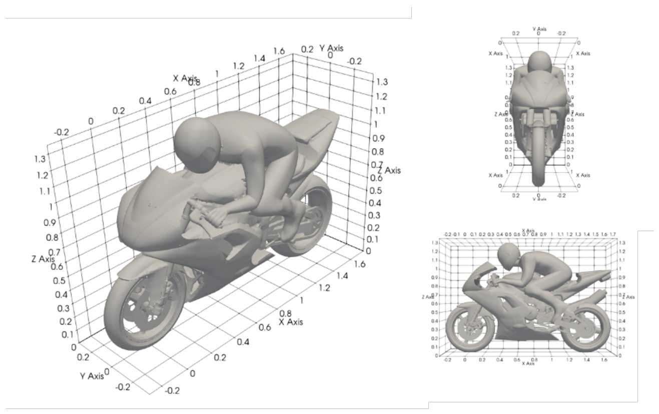


Figure 2: Base bike model geometry and dimensions.

CASE STUDIES

In this section, we provide four case studies ranging from applications of image-based methods to mesh-based methods for KPI and full-flow field predictions as well as the use of GenAI methods to generate new road-car geometries. Table 2 shows the summary of the training and inference costs for each case study and Table 3 shows the Amazon Elastic Compute Cloud (EC2) resources used on Amazon Web Services. See Appendix B for the hyperparameters used in each experiment and Appendix C for additional experimental results including solution profiles, error plots and loss curves.

Case Study	Model Name	Instance Type	Training Time (mins)	Training Cost	Inference Time (secs)	Inference Cost
I-KPI	Modified PointNet	p3.2xlarge	20	1.02 USD	5	0.004 USD
I-2D Slices	PointNet+CNN-VAE	p3.2xlarge	240	12.24 USD	5	0.004 USD
II	MeshGraphNets	g5.16xlarge	11	0.75 USD	2.2	0.0025 USD
III	U-Net	p3.2xlarge	720 (180 per variable (x4))	36.72 USD	1.5	0.0012 USD

Table 2: Summary of training and inference cost for each case study.

Amazon EC2 Instance Name	GPU/CPU type	GPU memory	CPU memory	CPU cores	cost in USD per-hour
p3.2xlarge	x1 Nvidia Tesla V100	16GB	61GB	8 vCPUs	3.06*
g5.16xlarge	x1 Nvidia A10g	24GB	256GB	64 vCPUs	4.096*
hpc6a.48xlarge	Dual-Socket AMD Milan	N/A	384GB	96 CPUs	2.88**
c4.8xlarge	Haswell E5-2666 v3	N/A	60GB	36 vCPUs	1.591*

* on-demand pricing in us-east-1
** on-demand pricing in us-east-2

Table 3: Details of Amazon EC2 instances used for dataset generation & ML training and inference.

CASE STUDY I: IMAGE-BASED PREDICTION OF THE DRIVAER DATASET USING POINTNET

Problem Statement & Objectives Within the automotive and motorsport sectors, it is common practice to automate the post-processing of CFD cases to obtain both forces and moments, i.e., C_l, C_d as well as select flow parameters, e.g., the velocity, vorticity and total pressure coefficients (CpT) on 2D slides in the x, y and z planes. These quantities are typically exported as images in common formats and help end-users to visualize the flow-field in a repeatable manner with lower storage requirements compared to visualizing the full 3D solution. Because of limited data storage, often the full solution is deleted after a certain time and only the images are kept for future analysis. For this reason, in this case study we explore how to use these images (as well as the geometry) as inputs to the ML algorithm. The aim being to predict an unseen geometry and produce both forces and moments as well as 2D slices of velocity and the CpT. We use the DrivAer dataset in Table 1 with the STL of the vehicle and the png images of the resulting flow-field along the x-direction as inputs to the model.

Computational Implementation and Results

KPI prediction We adapt the PointNet [42] algorithm, which is generally used for object classification, to perform regression on point cloud data. This is achieved by replacing the softmax classification loss with the Mean Squared Error (MSE) loss. The model, which we refer to as Modified PointNet, takes as an input a fixed size scattered point cloud ((x, y, z) points)

and outputs the C_d value (or in theory any relevant force or moment). During training, at each epoch we randomly sample 2048 points on the surface of each CFD mesh in the batch. We train the model for 500 epochs with an Adam optimiser on an Amazon EC2 p3.2xlarge instance (see Table 3). For the pure KPI prediction, training completed in under 20 minutes, resulting in a cost of training of 1.02 USD (see Table 2).

To validate the model, we perform the k-fold cross validation procedure with $k = 15$ different train/validation data splits. The validation set accuracy metrics are as follows: C_d is predicted with a Mean Percentage Error (MPE) of 1.72% and a Mean Absolute Error (MAE) of 4.6×10^{-3} averaged over the folds.

We then test the model on three unseen test datasets. Table 4 shows the Mean Squared Error (MSE) and the MAE for the C_d values for geometries in this test set.

Figure 3 visualizes the actual against the predicted C_d values and shows that the accuracy is valid for a number of datapoints. The time to predict the unseen geometries is 5 seconds on an Amazon EC2 p3.2xlarge instance (see Table 3), resulting in an inference cost of 0.004 USD (see Table 2).

We generate the training dataset using a hybrid RANS-LES transient method that takes more than 24 hours to run on several thousands cores of an Amazon EC2 hpc6a.48xlarge instance (see Table 3) at a cost of $>$ 1000 USD per run. Hence, the inference and training costs represent a very small fraction of the total cost. It is also worth comparing the cost of the Modified PointNet against a steady-state RANS version of this DrivAer case, which can be run in 8 hours on 384 cores (using the Amazon EC2 hpc6a.48xlarge instance) with a coarser mesh, i.e., $\sim$ 60M cells. Even in this scenario of the traditional CFD simulation costing $\sim$ 100 USD per run, the total training and inference cost of the ML method is still $100\times$ cheaper. Given the ML approach only predicts the KPI, whereas the CFD solver produces the whole 3D flow-field, it is arguably not a truly fair comparison. From a pragmatic point of view of a designer only caring about the downstream task of obtaining the drag or lift coefficient, it is still a useful comparison.

2D image slice prediction For the purpose of 2D image slice prediction, we modify the PointNet approach discussed for the KPI prediction model. Here, we target predicting 2D image slices of the Total Pressure Coefficient (C_pT). Engineers use this quantity to better understand the underlying aerodynamic phenomena affected by the novel design. Precisely, we aim to predict 9 C_pT slices, which are each 192×96 pixels, distributed

	MSE	MAE
Test Run 1	2.19×10^{-6}	1.48×10^{-3}
Test Run 2	4.37×10^{-6}	2.09×10^{-3}
Test Run 3	2.96×10^{-5}	5.44×10^{-3}
Average	1.21×10^{-5}	3.00×10^{-3}

Table 4: Mean Squared Error (MSE) and Mean Absolute Error (MAE) for the predicted C_d values in the test sets using Modified PointNet for the unseen DrivAer test datasets. Test 1,2,3 corresponds to run 3,7 & 11 in Figure 11

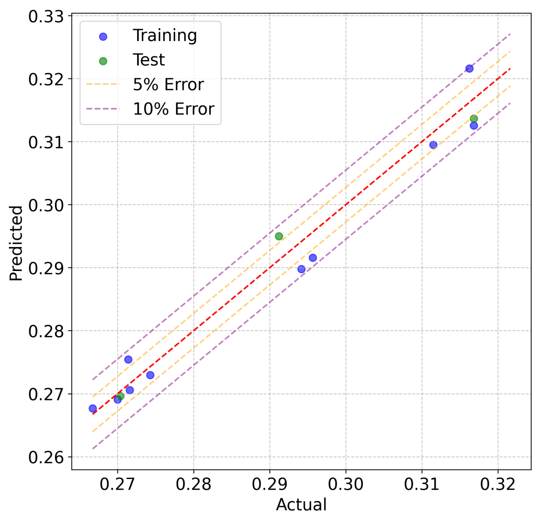


Figure 3: Actual vs Predicted C_d using the Modified PointNet [42] algorithm for the DrivAer Dataset.

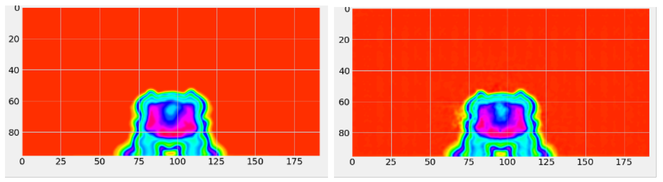


Figure 4: Comparison of the CFD results (*left*) and the Modified PointNet+CNN-VAE reconstructed C_pT slice (*right*).

longitudinally across the vehicles.

First, we build a Convolutional Variational AutoEncoder (CNN-VAE) [43, 44] to compress the $9 \times (192 \times 96)$ image slices into a 64×1 representation. Unlike traditional autoencoders, which map the input onto a latent vector, VAEs map the input data into the parameters of a probability distribution; the mean and variance of a Gaussian. This approach produces a continuous, structured latent space, which allows us to perform image generation by sampling from the probability distribution. We train CNN-VAE trained over 1000 epochs with an Adam optimiser (see Table 6 in Appendix B for a full list of the hyperparameters) on an Amazon EC2 p3.2xlarge instance (see Table 3). Training completed in 4 hours at a cost of 12.24 USD (see Table 2). Note that the time can be decreased by using a more powerful GPU instance.

Next, we train the previously mentioned Modified PointNet model to regress the 64x1 CNN-VAE probability distribution parameters. For inference, we sample the decoder of the CNN-VAE with the predicted probability distribution parameters to obtain the C_pT slices.

Figure 4 shows an example of an image of a reconstructed C_pT slice after compressing and reconstructing it through the CNN-VAE. There is some minor blurriness, especially around the edges of the flow field. This is common with VAEs and arises as a result of the Gaussianity assumption and thus the l_2 loss. See Figure 14 in Appendix C for additional C_pT slices and the corresponding error plots.

Similar to the KPI prediction case, the inference cost is negligible given that it is less than 5 seconds on an Amazon EC2 p3.2xlarge instance (see Table 3) at a cost of 0.004 USD (see Table 2). If we do our cost analysis against a traditional CFD run, we note that the inference time and cost is still many orders of magnitude decreased. The training cost for this 2D image slice prediction becomes more significant as we doing more than just predicting a KPI. It is still $3\times$ cheaper including both training and inference than for a RANS run (and still many times cheaper against a high-fidelity simulation). Clearly, the more predictions that are made the larger the savings over the traditional CFD approach become.

Future Outlook As discussed in the introduction, the targeted area for these data-driven ML methods is as low-medium fidelity methods, e.g., RANS simulations rather than the high-fidelity methods, e.g., HRLES or WMLES [11].

This case study largely confirms this hypothesis as the fidelity of the reconstructed flow field is good enough for a conceptual design phase, where an engineer is examining the high-level effects the geometry has on the flow field. Towards the rear of the vehicle, where complex 3D turbulent effects govern the flow field, the quality of the reconstruction is lower. Based on these results, we expect that with a larger training dataset, more robust image encoding techniques (e.g., VQ-VAE [45]), and potentially the use of some physics-informed regularisation [34] or constraints [32, 33], the quality of the reconstruction can be increased.

A major drawback of this Modified PointNet method is that the point cloud approach can be inefficient in resolving all of the geometry details. More importantly, without transfer learning, inference can only produce data at the 2D slices that was provided during the training, i.e., if the end-user wants to look at a slice that was not computed or trained upon, they cannot. This means the method is less flexible to the end-user. Ideally, the end-user may want to be able to use the underlying mesh from the CFD method and predict both forces and moments as well as the 3D flow field. We explore this mesh-based use-case in our next case study.

CASE STUDY II: MESH BASED PREDICTION OF THE DRIVAER DATASET USING MESHGRAPHNETS

<u>Problem Statement & Objectives</u> In the previous case study, we focused on using images as the input, given the logic that many end-users store their simulation outputs as images. As discussed in the previous subsection, this limits the flexibility of the trained model to predict on slices not in the training set. It also has limits on the resolution, given that the images are often not detailed enough to pick up small flow features. We also discussed how the use of a point cloud to represent the geometry has its disadvantages due to it being inefficient to resolve all of the geometry details (which can be many on a realistic road-car geometry).

In this case study, we focus on using the actual unstructured CFD mesh as the input to the model. Compared to images, this means the training data is comparably larger (i.e., GBs ratheer than KBs). Mesh-based Graph Neural Networks (GNNs) [23, 24] are viable approaches for modeling high resolution unstructured meshes. In particular, MeshGraphNets is a GNN-based method that learns simulation mesh data and is proposed for transient simulation modeling [24]. In this Case Study, we adapt MeshGraphNets for steady-state KPI prediction on mesh-based geometries. Note that steady-state data is more readily available than transient data since saving incremental time steps greatly increases data volumes. In the original Encoder-Processor-Decoder architecture, the Decoder takes the node updates from the processor and maps it to changes in velocity and acceleration using a separately learned fully-connected network. In our architecture, the processor is followed by a global graph max pooling layer. This returns a batch-wise graph-level-output by taking the channel-wise maximum across the node dimension. The output of this layer is of fixed shape equal to the hidden-dimension of the processor. Our decoder is then a simple fully connected layer to the shape of the KPI vector.

We initially set the goal to predict four different aerodynamic Key Performance Indicators (KPI): drag coefficient (C_d), lift coefficient (C_l), rear-axle lift coefficient (C_{lr}), and front-axle lift coefficient (C_{lf}). The model inputs are 19 DrivAer car geometries [40]. The large input mesh size increases training time and limits the number of message passing layers in the model due to the memory constraint. To overcome the memory challenges and reduce the training time, we down-sample the data to ∼17k nodes (see DS-DrivAer in Table 1).

<u>Computational Implementation and Results</u> We implement 5-fold cross validation to generate the train and validation sets. The model is trained on a Amazon EC2 g5.16xlarge instance (see Table 3) for 1000 epochs. We tune the hyperparameters (see Table 7 in Appendix B) through an iterative process considering a trade-off between computational cost and accuracy. The training time for each fold is less than 11 minutes, resulting in a cost of $\sim$ 0.75 USD per fold (see Table 2).

We test the trained model with 3 unseen geometries to demonstrate its generalizability. Figure 5 visualizes the actual vs. predicted values for the unseen data. Table 5 shows the Mean Square Error (MSE) and Mean Absolute Error (MAE) for each

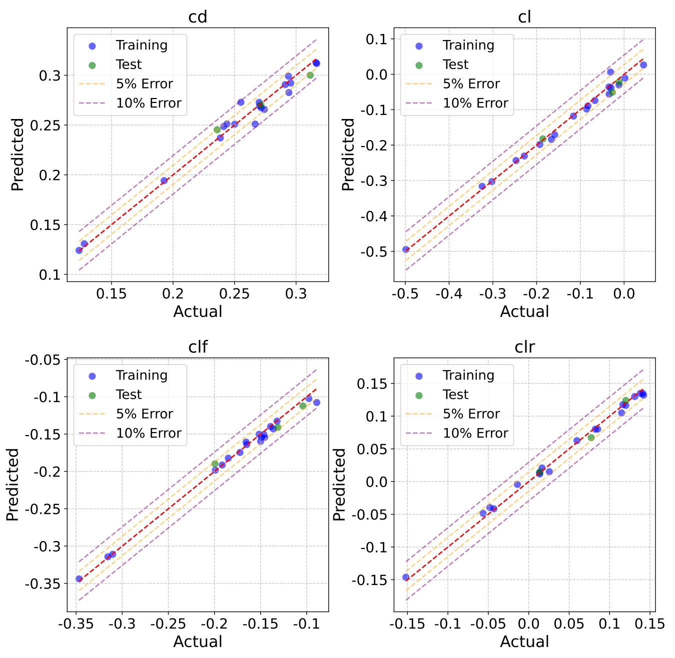


Figure 5: Actual vs. predicted KPIs using the MeshGraphNets model [24] on the down-sampled DS-DrivAer dataset.

KPI. These small errors demonstrate the potential and effectiveness of this model for KPI predictions.

We see that compared to the PointNet approach discussed in the first case study, the accuracy is similar (7.37×10^{-5}) for C_d and the time and cost to predict the model is also similar. The time to train the model is naturally dependent on the number of nodes on the surface. Using the downsampling technique, the training time is slightly lower than that of the PointNet approach. For larger meshes the time (and cost) will increase.

	MSE	**MAE**
C_d	7.37×10^{-5}	7.30×10^{-3}
C_l	23.23×10^{-5}	12.54×10^{-3}
C_{lf}	8.44×10^{-5}	9.15×10^{-3}
C_{lr}	3.97×10^{-5}	4.98×10^{-3}

Table 5: MeshGraphNets model error analysis on the 3 unseen test geometries of the DS-DrivAer dataset.

<u>Future Outlook</u> In this case study, we have demonstrated promising results on small-scale examples consisting of leveraging MeshGraphNets [24] for KPI prediction using the surface mesh of the geometry. Various difficulties arise when scaling the MeshGraphNet model to full flow field predictions (outputting node pressure and velocity fields) on industrial-sized real-world 3D cases. In preliminary work, we have so-far identified the following three challenges:

1. Memory constraints in scaling from canonical test cases in academic papers, which typically have a few thousand of nodes per graphs in 2D to 10-100s of millions of nodes per graph in 3D;
2. Most available historical data of already run simulations is

steady state and not transient. This departs from the original assumed implementation of MeshGraphNet [24] with limited examples in the literature of implementing MeshGraphNet on steady state data [**?**];

3. Message propagation in the network needs to travel farther to go the same spatial distance on finer meshes due to the need to pass messages from non-local nodes to local nodes. Decreased message passing layers can lead to larger errors and hinder model convergence as the minimum edge length decreases [23, 25]. This arises both because of the task of predicting steady state (vs. transient) and the larger scale.

We will discuss these challenges as well as initial results in future papers dedicated specifically to this area of large-scale 3D flow-field prediction that is highly relevant to the road-car aerodynamics community.

CASE STUDY III: APPROXIMATING MOTORCYCLE AERODYNAMICS USING 3D U-NETS

Problem Statement & Objectives In the prior subsection, we used the MeshGraphNets [24] approach to compute KPIs and discussed its ability to predict full flow field.

In this case study, we demonstrate an alternative approach to predicting a full flow field case using a U-Net architecture [46, 47] on the Motorbike dataset in Table 1. In contrast to the MeshGraphNets approach, the U-Net approach operates on structured grids, which requires an interpolated resampling of unstructured meshes ($\sim$700k hex-dominated cells) to a ($\sim$2M node) structured grid as an additional step.

This requirement to use structured grids as an input can be seen as one of the disadvantages of the U-Net approach, given how common unstructured meshes are in the CFD community. Nevertheless, it still represents a commonly used ML algorithm that is insightful to explore.

Computational Implementation and Results We train the U-Net using 500 simulations with variants of the base motorbike and a chosen resolution of 2,097,152 points (grid nodes, 128^3) for capturing key geometric features of the bike. For each of the variables, i.e., pressure p and each component of the velocity, i.e., (u_x, u_y, u_z), we train separate models in parallel for computational efficiency. The training data generation takes approximately 30 minutes on an Amazon EC2 c4.8xlarge instance per simulation running in parallel with a total cost of $\sim$ 400 USD. The general runtime for each training session is between 2-3 hours on a V100 GPU (a total runtime of $\sim$ 12 hours across all 4 models) on an Amazon EC2 p3.2xlarge instance (see Table 3) amounting to a total cost of $\sim$ \$36 (see Table 2).

We test the predictions from the trained model on an unseen variant of the original bike, i.e., with differences in roll, pitch, yaw and affine stretches in the x, y, z directions. Inferences take approximately 1.5 seconds (see Table 2) on a Amazon EC2 p3.2xlarge instance (see Table 3), yielding a 800-1200x speedup over the individual CFD simulations used to generate the training data.

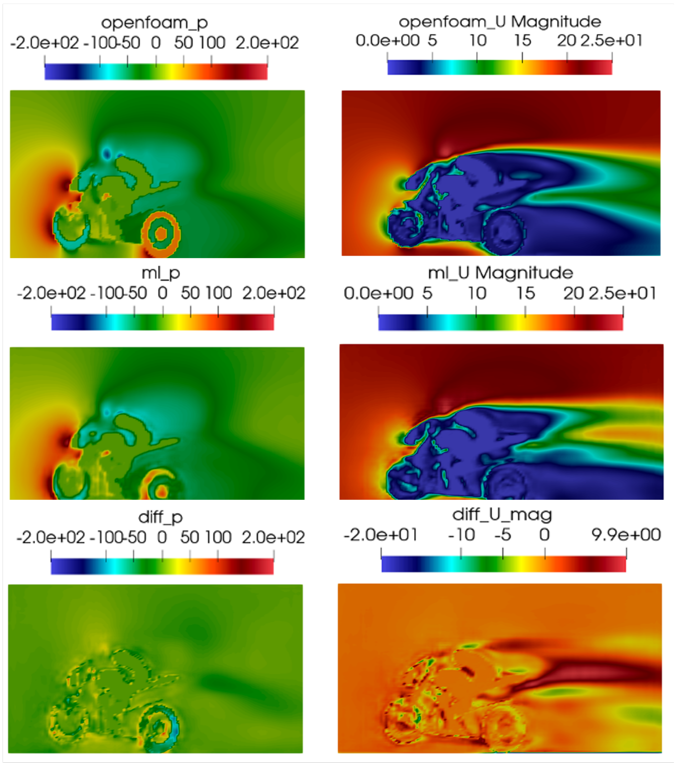


Figure 6: Pressure (*top, left*) and velocity magnitudes (*top, right*) for CFD (*left*), U-Net (*middle*) and error (*bottom*) on a side-view slice of the Motorbike dataset.

Figure 6 shows the side view of the bike. The pressure values show a strong match between the CFD simulations and U-Net predictions, except near the wall regions of the bike. The velocity magnitude map shows qualitatively that the U-Net predictions are able to capture the main flow features; however, there is a sharper structure in the momentum deficit region. The CFD results also show a small recirculation region (small blue area) above the tail of the pipe which is not present in the U-Net predictions. The U-Net predictions generally exhibit sharper transitions in velocity. Improving the velocity regularization to avoid these sharp transitions regions may improve the quality of predictions. For alternate slices, see the front view of the bike in Figure 16 in Appendix C, which shows the strongest correspondence between the CFD results and U-Net predictions, with discrepancies occurring only near the surface of the geometry.

Future Outlook This work presented to date, while preliminary in nature, does highlight the computational efficiency of using an example ML method to predict a full 3D flow-field. Below we discuss the main challenges that we observed during this case study and that are common to other current data-driven approaches.

Purely data-driven approaches, as shown in this case study, can result in physical constraints not being satisfied, e.g., violating the divergence-free (conservation of mass) constraint here. Adding the divergence of velocity to satisfy incompressibility and Laplacian of pressure as a regularization to smooth the pressure contours in the loss function during training ala PINNs [34] or enforcing conservation as a hard constraint [32] may help overcome these violations. In addition, our approach creates a hierarchy of individual models per variable. The physics governing equations (incompressible Navier-Stokes) imply strong interdependencies between these variables. Future work includes exploring a more coupled approach. Lastly, capturing geometric details and internal boundaries is another limitation of the present approach. Due to the voxelization of the solid geometry and surfaces on bike, some surface features are lost. In order to accurately capture intricate surface details, unstructured grid methods using GNNs [22, 23] or MeshGraphNets are being actively explored [24], as discussed in the prior subsection.

For all the previous case studies, producing a large-scale training dataset has been a challenge. Geometry and mesh generation, that is manually creating parametric meshes or CAD models, requires significant human effort. An area that can play a role in making ML methods easier to incorporate into a road-car design process is generating synthetic geometry and meshes using Generative AI techniques (e.g., Stable Diffusion [48] or NeRF [49]). Hence, we explore this area in the next case study.

CASE STUDY IV: GENERATING CAR GEOMETRIES USING STABLE DIFFUSION

Problem Statement & Objectives In this case study, we explore the use of Generative AI (GenAI) in the current pipeline of engineering simulations. The use of GenAI to conduct automotive design space exploration has been explored before [50]. In our case study, we include image-to-3D-mesh generation, which opens the possibility of running full 3D CFD simulations on the generated designs. The generated meshes can then be used in downstream simulations or to train ML models. The goal of this case study is to parametrically modify the aerodynamic design of a road vehicle while retaining design philosophies from existing cars.

Computational Implementation and Results Figure 7 illustrates the overall workflow. Specifically, the process is guided by a physics-informed workflow, where the drag coefficients of intermediate designs are computed using CFD simulations [41]. The initial design is perturbed using Stable Diffusion [48], which takes the original image as input with a prompt to make the design more aerodynamic (purely from text interpretation without physics-guided denoising). The resulting variant images are first converted to point clouds using bootstrapped Neural Radiance Fields (NeRF) [49]. The point cloud data is then reconstructed into a mesh using Neural Kernel Surface Reconstruction (NKSR) [51]. This mesh is then used as input to the CFD simulation. We rank the designs using the resulting drag coefficients. The best design is used in place of the initial image to produce variants in the next generation cycle of the workflow.

In order to evaluate the methods and workflow, a stock image of

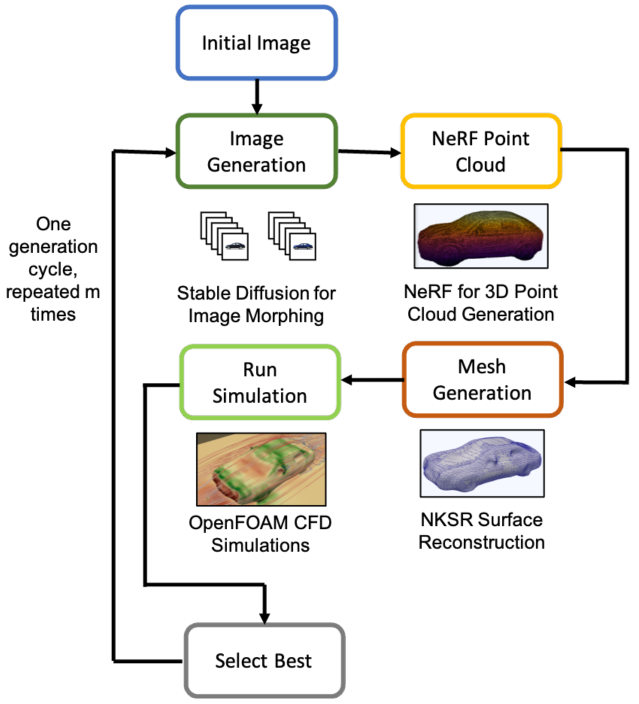

Initial Image
Image Generation
NeRF Point Cloud
One generation cycle, repeated m times
Stable Diffusion for Image Morphing
NeRF for 3D Point Cloud Generation
Run Simulation
Mesh Generation
OpenFOAM CFD Simulations
NKSR Surface Reconstruction
Select Best

Figure 7: Overall workflow for Generative AI automotive design to improve aerodynamics.

a sedan is used as the input image to the Stable Diffusion model (see Cycle 1 in Figure 8). This design progressively changes by subtly adopting different design elements through the generational cycles. Cycle 20 in Figure 8 shows the resulting best design of each generation.

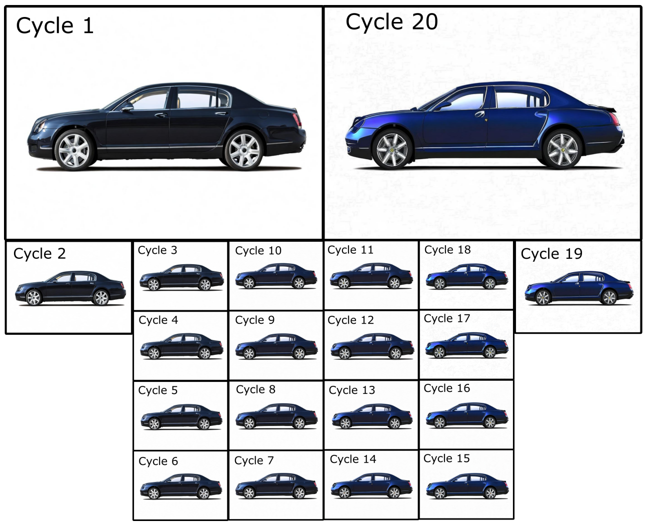


Figure 8: Progressive design changes given a stock image of a sedan showing improvements in drag coefficient.

During the NeRF stage of each cycle, the 2D image generated through Stable Diffusion is converted into a point cloud. Figure 9 shows an example of this process, where the density of the point cloud is highest at the sides and lowest towards the centerline of the car. This is due to information deficiency away from the side-view of the car and interpolation in those regions. Overall, the point cloud captures the rough shape of the car. In this chosen case, the NeRF algorithm predicts an open rear window, and loses detailed information on the side-view mirror.

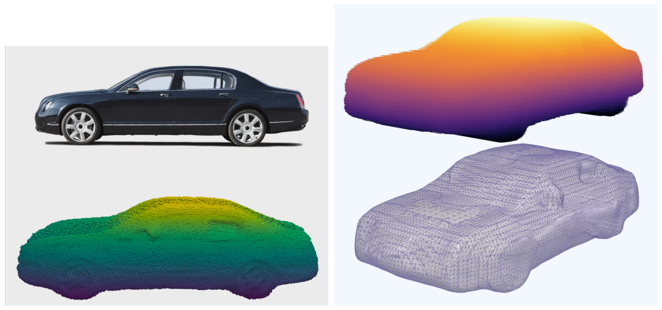


Figure 9: This series of images represent the mesh generation design cycle. From top left, an image is used to create a point cloud (bottom left) using NeRF [49], colored by height. As shown in the top right, once again colored by height, a surface mesh generated from this point cloud using NKSR [51], and shown as a wire frame mesh in the bottom right.

In the next stage, the point cloud is converted into a surface mesh. Figure 9 illustrates an example of the surface reconstruction procedure using Neural Kernel Surface Reconstruction (NKSR) [52]. Compared to CAD quality meshes, the resulting surface topology appears smoothed and not sufficiently detailed. This particularly occurs near the side-view mirrors, which amplifies the discrepancy seen in the NeRF stage. There is also a dent in the rear window, consequent to the observation of an open window during the NeRF stage of the workflow.

Lastly, CFD simulations are run on the resulting 3D mesh to obtain drag coefficients, using similar setup to the previous case study. Figure 10 shows the best drag coefficients for each generation cycle. While the decrease is not monotone, there is a general downward trend in the drag coefficient going from Cycle 1 to Cycle 20, demonstrating the potential for our method to provide gradual design change suggestions that can improve efficiency of automotive vehicles.

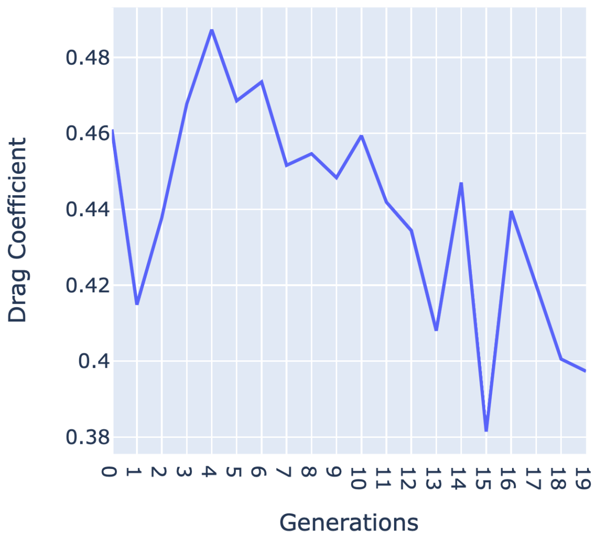


Figure 10: Plot of drag coefficient for each of the best designs across generations.

Future Outlook In this case study, we have shown that visual generative AI models, e.g., stable diffusion, and recent computer vision methods, e.g., NeRF and NKSR, which incorporate generative adversarial networks (GANs) show promise in the conceptual design process. There still remain several limitations in going from an image to a high quality 3D mesh.

The stable diffusion model only provides the side view of the car, hence in the first stage of the pipeline depth variations are not captured. Based on this modified side profile, the GAN embedded within the NERF image-to-3D pipeline captures the depth change based on prior training data from the foundational training dataset with similar side views, introducing a variation in depth, however this is not reliable. Additionally, the complex geometries of the under-body, wheels and wheel wells are not captured in the generated 3D mesh.

By training foundation models to improve the fidelity of reconstruction [53], and improving text-to-3D methods [54], this is expected to improve significantly. Using such tools, non-parametric generative design with a physics-based workflow, can aid conceptual designers in automotive, motorsport and other industries.

CONCLUSIONS

In this paper, we present four different case studies in the emerging area of applying machine learning to augment traditional Computational Fluid Dynamics (CFD) simulations.

Whilst the results of these case studies are preliminary in nature, without a consistent training dataset, these methods do show promise, particularly in terms of computational efficiency compared to traditional CFD methods. In general, we see orders of magnitude improvements in cost-to-solution in the prediction/inference step, which does not vary greatly whether the prediction is a KPI or a full flow field. The training time, which varies depending on what the model is predicting, is still typically less than the cost of a single CFD run. This represents an acceptable cost provided that training data is available. Further work is required to demonstrate the accuracy of these methods. This work includes exploring the addition of physical constraints and the application of these methods to industrial-sized cases. These challenges have not yet been addressed in the available public literature. Overall, this work highlights both the promise of these ML methods as well as the need to explore a number of different algorithms depending on the desired outputs to predict.

ACKNOWLEDGEMENTS

The authors are grateful to the following individuals: Michael W. Mahoney, Shima Alizadeh, Gaurav Gupta, Yuyang Wang, Andrew Stuart, Steven Miller, Samuel Gundry, Sanjeev Kulkarni, Fei Chen and Octavi Obiols-Sales. The authors would like to acknowledge the support of *Amazon Web Services* and *Amazon Science* for providing guidance and computational resources.

REFERENCES

[1] Oettle, N. and Sims-Williams, D., "Automotive aeroacoustics: An overview," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, Vol. 231, No. 9, 8 2017, pp. 1177–1189.

[2] García-León, R. A., Afanador-García, N., and Gómez-Camperos, J. A., "Numerical study of heat transfer and speed air flow on performance of an auto-ventilated disc brake," *Fluids*, Vol. 6, No. 4, 4 2021.

[3] Patidar, A., Natarajan, S., and Pande, M., "CFD Analysis and Validation of an Automotive HVAC System," *SAE World Congress & Exhibition*, SAE, 4 2009.

[4] Gaylard, A. P., Kabanovs, A., Jilesen, J., Kirwan, K., and Lockerby, D. A., "Simulation of rear surface contamination for a simple bluff body," *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 165, No. January 2016, 2017, pp. 13–22.

[5] Ghosh, D., Maguire, P. D., and Zhu, D. X., "Design and CFD Simulation of a Battery Module for a Hybrid Electric Vehicle Battery Pack," *SAE World Congress & Exhibition*, SAE, 4 2009.

[6] Ashton, N. and Van Noordt, W., "Overview and Summary of the First Automotive CFD Prediction Workshop: DrivAer Model," *SAE International Journal of Commercial Vehicles*, Vol. 16, No. 1, 8 2022.

[7] Hupertz, B., Lewington, N., Mockett, C., Ashton, N., and Duan, L., "Towards a Standardized Assessment of Automotive Aerodynamic CFD Prediction Capability - AutoCFD 2: Ford DrivAer Test Case Summary," *SAE Technical Papers*, No. 2022, SAE International, 3 2022.

[8] Page, G. J. and Walle, A., "Towards a Standardized Assessment of Automotive Aerodynamic CFD Prediction Capability - AutoCFD 2: Windsor Body Test Case Summary," *SAE Technical Papers*, No. 2022, SAE International, 3 2022.

[9] LeVeque, R. J., *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, 2002.

[10] LeVeque, R. J., *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, SIAM, 2007.

[11] Ashton, N., West, A., Lardeau, S., and Revell, A., "Assessment of RANS and DES methods for realistic automotive models," *Computers & Fluids*, Vol. 128, 2016, pp. 1–15.

[12] Ashton, N., West, A., and Mendonça, F., "Flow Dynamics Past a 30P30N Three-Element Airfoil Using Improved Delayed Detached-Eddy Simulation," *AIAA Journal*, 2016, pp. 1–10.

[13] Appa, J., Turner, M., and Ashton, N., "Performance of CPU and GPU HPC Architectures for off-design aircraft simulations," *AIAA Scitech 2021 Forum*, No. January, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1 2021, pp. 11–15.

[14] Ashton, N., Sachs, S., Foti, L., and Eberhardt, S., "Towards High-Fidelity CFD on the Cloud for the Automotive and Motorsport sectors," *SAE World Congress*, detroit, 2020.

[15] Hughes, T. J., *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Courier Corporation, 2012.

[16] Dolejší, V. and Feistauer, M., *Discontinuous Galerkin Method, Analysis and Applications to Compressible Flow*, Vol. 48, Springer Series in Computational Mathematics (SSCM), 2015.

[17] Brehm, C. and Ashton, N., "Progress in the development of an immersed boundary method with viscous-wall model for 3D flows," *Eleventh International Conference on Computational Fluid Dynamics (ICCFD11), Barcelona, Spain*, 2018.

[18] Islam, A., Gaylard, A., and Thornber, B., "A detailed statistical study of unsteady wake dynamics from automotive bluff bodies," *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 171, No. October, 2017, pp. 161–177.

[19] Vinuesa, R. and Brunton, S. L., "Emerging Trends in Machine Learning for Computational Fluid Dynamics," *Computing in Science & Engineering*, Vol. 24, No. 5, 9 2022, pp. 33–41.

[20] Lino, M., Fotiadis, S., Bharath, A. A., and Cantwell, C. D., "Current and emerging deep-learning methods for the simulation of fluid dynamics," 7 2023.

[21] Evans, L., *Partial Differential Equations*, Vol. 19 of *Graduate studies in mathematics*, American Mathematical Society, 2nd ed., 2010.

[22] Brandstetter, J., Worrall, D., and Welling, M., "Message Passing Neural PDE Solvers," *International Conference on Learning Representations*, 2022.

[23] Gladstone, R. J., Helia Rahmani, V. S., Meidani, H., D'Elia, M., and Zareei, A., "GNN-based physics solver for time-independent PDEs," *arXiv preprint arXiv:2303.15681*, 2023.

[24] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W., "Learning Mesh-Based Simulation with Graph Networks," *International Conference on Learning Representations*, 2021.

[25] Fortunato, M., Pfaff, T., Wirnsberger, P., Pritzel, A., and Battaglia, P., "MultiScale MeshGraphNets," *arXiv preprint arXiv:2210.00612*, 2022.

[26] Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Vinyals, O., Stott, J., Pritzel, A., Mohamed, S., and Battaglia, P., "GraphCast: Learning skillful medium-range global weather forecasting," 2023.

[27] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A., "Neural Operator: Graph Kernel Network for Partial Differential Equations," *arXiv preprint arXiv:2003.03485*, 2020.

[28] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A., "Fourier Neural Operator for Parametric Partial Differential Equations," *International Conference on Learning Representations*, 2021.

[29] Gupta, G., Xiao, X., and Bogdan, P., "Multiwavelet-Based Operator Learning for Differential Equations," *Advances in Neural Information Processing Systems*, Vol. 34, 2021.

[30] Li, Z., Kovachki, N. B., Choy, C., Li, B., Kossaifi, J., Otta, S. P., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., and Anandkumar, A., "Geometry-Informed Neural Operator for Large-Scale 3D PDEs," 9 2023.

[31] Kadambi, A., de Melo, C., Hsieh, C., Srivastava, M., and Soatto, S., "Incorporating physics into data-driven computer vision," 2023.

[32] Hansen*, D., Maddix*, D. C., Alizadeh, S., Gupta, G., and Mahoney, M. W., "Learning Physical Models that Can Respect Conservation Laws," *Physica D: Nonlinear Phenomena*, Vol. 457, 2024, pp. 133952.

[33] Saad, N., Gupta, G., Alizadeh, S., and Maddix, D. C., "Guiding continuous operator learning through Physics-based boundary constraints," *International Conference on Learning Representations*, 2023.

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

[35] Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A., "Physics-informed neural operator for learning partial differential equations," *arXiv preprint arXiv:2111.03794*, 2021.

[36] Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W., "Characterizing Possible Failure Modes in Physics-Informed Neural Networks," *Advances in Neural Information Processing Systems*, Vol. 34, 2021, pp. 26548–26560.

[37] Edwards, C., "Neural Networks Learn to Speed Up Simulations," *Communications of the ACM*, Vol. 65, No. 5, 2022, pp. 27–29.

[38] Négiar, G., Mahoney, M. W., and Krishnapriyan, A. S., "Learning differentiable solvers for systems with hard constraints," *International Conference on Learning Representations*, 2023.

[39] Heft, A. I. and Adams, N. A., "Experimental and numerical investigation of the drivaer model," *Proceedings of the ASME 2012 Fluids Engineering Summer meeting*, 2012, pp. 1–11.

[40] of Munich, T. U., "DrivAer Model," `http://www.drivaer.com`, 2018.

[41] Jasak, H., Jemcov, A., Tukovic, Z., et al., "OpenFOAM: A C++ library for complex physics simulations," *International workshop on coupled methods in numerical dynamics*, Vol. 1000, 2007, pp. 1–20.

[42] Charles, R., Su, H., Kaichun, M., and Guibas, L. J., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, Los Alamitos, CA, USA, jul 2017, pp. 77–85.

[43] Kingma, D. P. and Welling, M., "Auto-Encoding Variational Bayes," 2022.

[44] Rezende, D. J., Mohamed, S., and Wierstra, D., "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," 2014.

[45] van den Oord, A., Vinyals, O., and Kavukcuoglu, K., "Neural Discrete Representation Learning," 2018.

[46] Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O., "3D U-Net: learning dense volumetric segmentation from sparse annotation," *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, Springer, 2016, pp. 424–432.

[47] Lee, K., Zung, J., Li, P., Jain, V., and Seung, H. S., "Superhuman accuracy on the SNEMI3D connectomics challenge," *arXiv preprint arXiv:1706.00120*, 2017.

[48] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., "High-resolution image synthesis with latent diffusion models," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10684–10695.

[49] Pavllo, D., Tan, D. J., Rakotosaona, M.-J., and Tombari, F., "Shape, Pose, and Appearance from a Single Image via Bootstrapped Radiance Field Inversion," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[50] Arechiga, N., Permenter, F., Song, B., and Yuan, C., "Drag-guided diffusion models for vehicle image generation," 2023.

[51] Jiahui Huang, Z. G., Atzmon, M., Litany, O., Fidler, S., and Williams, F., "Neural Kernel Surface Reconstruction," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[52] Huang, J., Gojcic, Z., Atzmon, M., Litany, O., Fidler, S., and Williams, F., "Neural Kernel Surface Reconstruction," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4369–4379.

[53] Gao, J., Shen, T., Wang, Z., Chen, W., Yin, K., Li, D., Litany, O., Gojcic, Z., and Fidler, S., “GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images,” *Advances In Neural Information Processing Systems*, 2022.

[54] Poole, B., Jain, A., Barron, J. T., and Mildenhall, B., “Dreamfusion: Text-to-3d using 2d diffusion,” *arXiv preprint arXiv:2209.14988*, 2022.

[55] Dubey, P., Pramod, M. Y., Kumar, A. S., and Kannan, B., “Numerical simulation of flow over a racing motorbike using OpenFOAM®,” *AIP Conference Proceedings*, Vol. 2277, AIP Publishing, 2020.

[56] Xiang, Y., Mottaghi, R., and Savarese, S., “Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild,” *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.

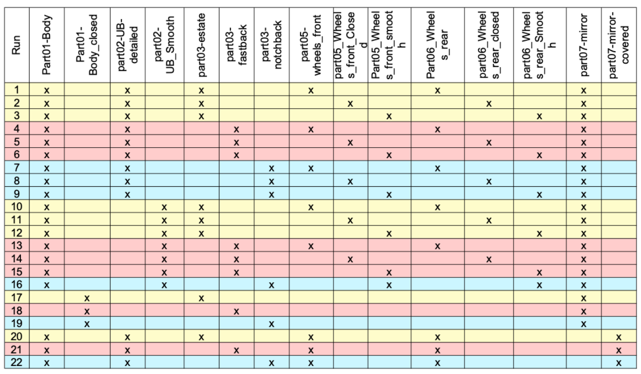
[57] Hodgkinson, L., van der Heide, C., Roosta, F., and Mahoney, M. W., “Monotonicity and Double Descent in Uncertainty Estimation with Gaussian Processes,” *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[58] John, V., Linke, A., Merdon, C., Neilan, M., and Rebholz, L. G., “On the divergence constraint in mixed finite element methods for incompressible flows,” *SIAM review*, Vol. 59, No. 3, 2017, pp. 492–544.

APPENDIX A - DATASET GENERATION

DRIVAER We built 22 different variants of the DrivAer using the differing parts available from the TUM website[4] to cover a broad range of body styles. In particular, the geometries combined fast back, estate back and notch back bodies, closed and open-wheel covers and detailed or flat smooth under bodies. These were then meshed using approximately 300M cells using unstructured prismatic-hex dominant cell types. Simulations were run using a SA-DDES based hybrid RANS-LES approach and run for approximately 2s of flow-time (averaging over the last 1st). The inlet velocity was set to $40ms^{-1}$ with corresponding wheel rotation and ground velocity to match. All other farfield walls were set to a symmetry condition to replicate an effective free-air condition. Outputs were then produced for forces and moments as well as time-averaged qualities for the full flow-field as well as 2D slices in x, y and z directions in *.png* format.

MOTORBIKE We used OpenFOAM 23.06 [41, 55] to compute the flow field around the vehicle. We built an unstructured hex-dominated mesh with prismatic boundary layer cells using blockMesh and SnappyHexMesh from the .obj file we generated in the previous step. We intentionally opted for lower refinement levels than what is typically employed in industry (we can increase these levels as required). Our mesh count was approximately one million cells, on average – this changes slightly depending on the geometry itself. To accelerate the CFD part of the process we restricted ourselves to steady-state RANS simulations using the k-omega SST model (for industrial applications you could extend this to use hybrid RANS-LES or WMLES methods, which have a higher fidelity). Finally, in our setup we used the simpleFoam solver based upon the semi-implicit method for pressure-linked equations (SIMPLE) algorithm. In order to generate the motorbike dataset, the base motorbike geometry is subject to affine deformations and rotations 500 times, with each being a test case. The roll, pitch, yaw, and X, Y, Z stretches are shown in Figures 12 and 13 respectively. The corresponding points within the training data, validation data and the test data point used in the main text, within Figure 6 are shown in these figures.



Run	Part01-Body	Part01-Body_closed	part02-UB-detailed	part02-UB_Smooth	part03-estate	part03-fastback	part03-notchback	part05-wheels_front	part05_Wheels_front_Closed	Part05_Wheels_front_smooth	Part06_Wheels_rear	part06_Wheels_rear_closed	part06_Wheels_rear_Smooth	part07-mirror	part07-mirror-covered
1	x		x		x			x			x			x	
2	x		x		x				x			x		x	
3	x		x		x					x			x	x	
4	x		x			x		x			x			x	
5	x		x			x			x			x		x	
6	x		x			x				x			x	x	
7	x		x				x	x			x			x	
8	x		x				x		x			x		x	
9	x		x				x			x			x	x	
10	x			x	x			x			x			x	
11	x			x	x				x			x		x	
12	x			x	x					x			x	x	
13	x			x		x		x			x			x	
14	x			x		x			x			x		x	
15	x			x		x				x			x	x	
16	x			x			x			x			x	x	
17		x			x									x	
18		x				x								x	
19		x					x							x	
20	x		x		x			x			x				x
21	x		x			x		x			x				x
22	x		x				x	x			x				x

Figure 11: Inputs parts for the 22 DrivAer variants from the TUM website (https://www.epc.ed.tum.de/en/aer/research-groups/automotive/drivaer/geometry/)

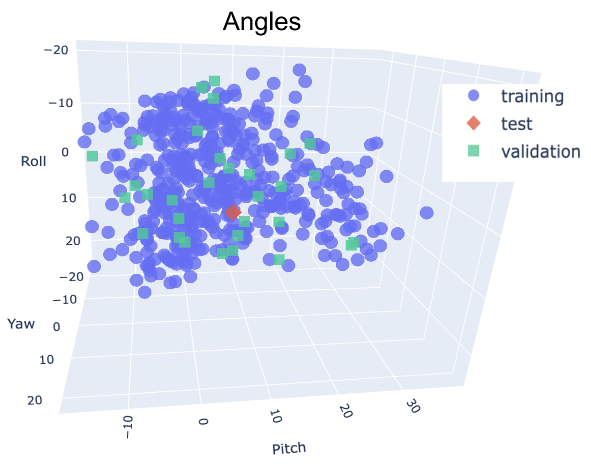


Figure 12: The non-commutative angles used to generate the dataset, showing the split between training, validation and test cases for the motorbike example.

Each motorbike test case was run with a fixed freestream velocity of 20 ms^{-1}.

The SnappyHexMesh utility in OpenFOAM is used to mesh the volume around the motorbike (see Figure 2) and results in a mesh of approximately 700k cells per simulation. Each simulation was run until convergence was reached in both the drag coefficient and the residuals - typically around 2000 iterations.

[4] https://www.epc.ed.tum.de/en/aer/research-groups/automotive/drivaer/

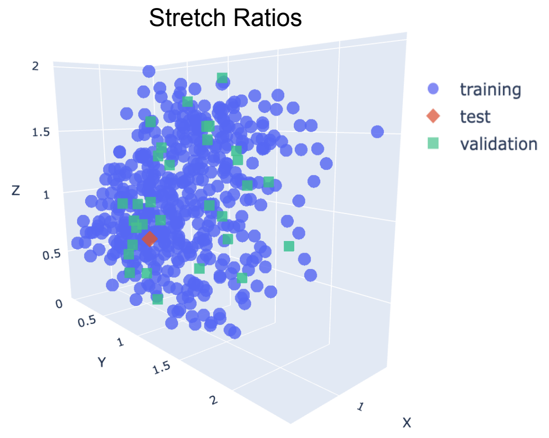


Figure 13: The stretches used to generate the dataset, showing the split between training, validation and test cases for the motorbike example.

APPENDIX B - EXPERIMENTAL DETAILS

CASE STUDY I: IMAGE BASED PREDICTION OF THE DRIVAER DATASET USING POINTNET Table 6 displays the hyperparameters for the PointNet model for the image-based DrivAer case study.

Hyperparameter	Selected Value
Number of encoder layers	8
Latent size dimension	512
Learning rate	5e-4
Weight decay	0
Optimizer	Adam
Batch Size	1
Number of epochs	2000

Table 6: Hyperparameters for the image-based PointNet [42] model in the down-sampled DrivAer (DS-DrivAer) KPI case study.

CASE STUDY II: MESH BASED PREDICTION OF THE DRIVAER DATASET USING MESHGRAPHNETS Table 7 summarizes the hyperparameters for the MeshGraphNet model for the DrivAer KPI prediction case study using the mesh data.

Hyperparameter	Selected Value
Number of message passing layers	10
Number of hidden layers	10
Processor hidden dimension	256
Learning rate	2e-3
Weight decay	0.005
Optimizer	Adam
Batch Size	1
Number of epochs	1000

Table 7: Hyperparameters for the MeshGraphNets [24] model used in the DrivAer case study.

CASE STUDY III: APPROXIMATING MOTORCYCLE AERODYNAMICS USING 3D U-NETS Table 8 shows the hyperparameters for the U-Net model on the Motorbike case study.

Hyperparameter	Selected Value
Encoder Layers	8
Decoder Layers	8
Learning rate	2e-4
Weight decay	1e-5
Optimizer	Adam
Batch Size	5
Max Iterations	9000
Number of epochs	90

Table 8: Hyperparameters the for U-Net [46] model in the Motorbike case study.

CASE STUDY IV: GENERATING CAR GEOMETRIES USING STABLE DIFFUSION Table 9 shows the hyperparameters used in the Generative AI case study. Note that most of the models used here are foundation models, and only NeRF [49] was fine-tuned with the Pascal3D car dataset [56].

Hyperparameter	Selected Value
Stable Diffusion	
Strength	0.35
Guidance Scale	0.55
NeRF	
Inversion Steps	250
Iterations	250
Resolution	310
NKSR	
Voxel Size	0.1
Kernel Dim	4
Tree Depth	4
UNet Maps	32
Optimizer	Adam
Learning Rate	1.00E-04

Table 9: Hyperparameters for the Stable Diffusion 2.1 [48], NeRF [49] and NKSR [52] models used in the generative AI mesh generation case study.

APPENDIX C - ADDITIONAL EXPERIMENTAL RESULTS

CASE STUDY I: IMAGE BASED PREDICTION OF THE DRIVAER DATASET USING POINTNET Figure 14 illustrates a comparison of the predicted C_pT slices (*left*), actual (*middle*) and absolute errors (*right*) for several chosen test samples. We see that the error is concentrated around the boundary of the surface.

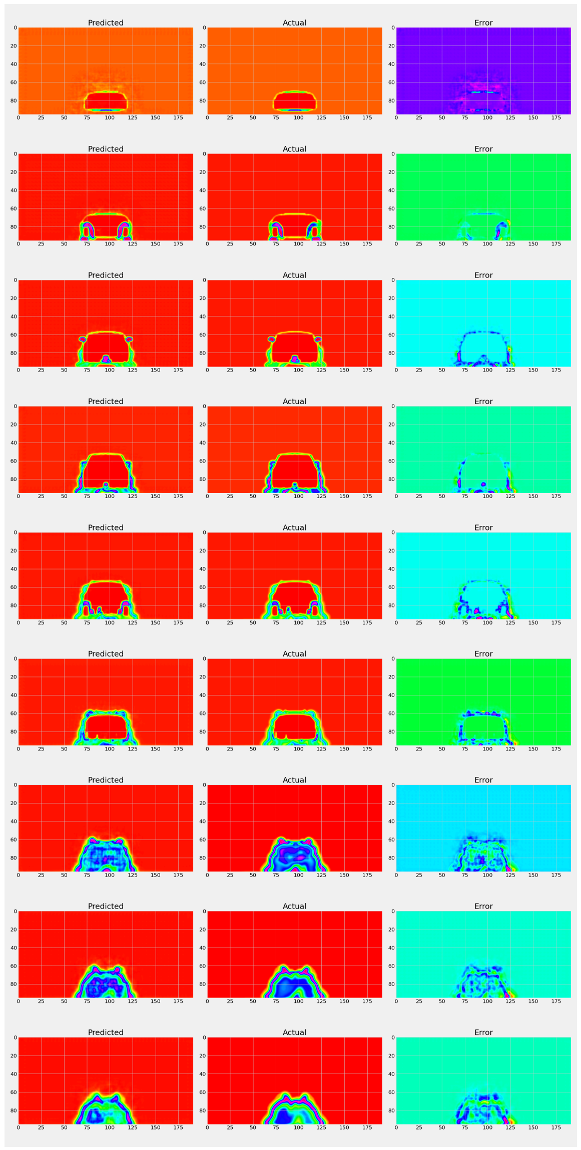


Figure 14: CFD C_pT, PointNet+CNN-VAE predicted C_pT and error analysis for an example in the DrivAer [40] test dataset.

CASE STUDY II: MESH BASED PREDICTION OF THE DRIVAER DATASET USING MESHGRAPHNETS Figure 15 shows the training (*blue*) and validation loss (*orange*) for the MeshGraphNet [24] model on the down-sampled DS-DrivAer dataset on the KPI prediction downstream task. This plot shows that the training loss is less than the validation loss as expected and that the model has converged around 500 epochs.

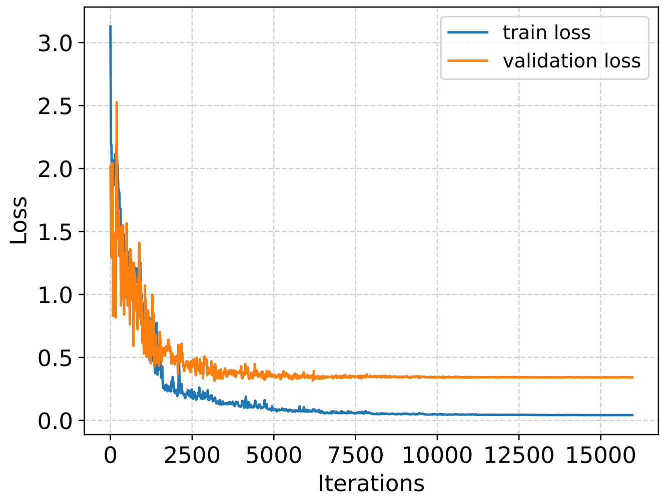


Figure 15: MeshGraphNets model loss for DrivAer dataset.

CASE STUDY III: APPROXIMATING MOTORCYCLE AERODYNAMICS USING 3D U-NETS Figure 16 shows another slice of the front view of the Motorbike, which compares the pressure, velocity magnitudes and errors between the CFD simulation and the U-Net prediction.

Figure 17 shows the training and validation losses for the unknown pressure p over 9000 iterations. We see that the loss decreases from approximately 350 after 10 iterations to approximately 40. There is non-monotonic behavior with a spike in error around 8000 iterations, which may be related to the double descent phenomena [57]. Similarly, this figure shows the asymptote of the validation score of MSE evaluated over the control volume, which includes the entire region of air simulated around the bike, going from approximately 125 to 30.

In incompressible flow, for continuity and mass balance, the flow has to be divergence-free (i.e. divergence of velocity is 0) [58]. This is generally the case in numerical simulations except due to unconverged iterations or numerical artifacts (coarse meshes). However, this information is not present in the ML training as a constraint, hence by plotting the divergence it is possible to quantify the unphysical regions of the ML predictions. This information is shown in Figure 18.

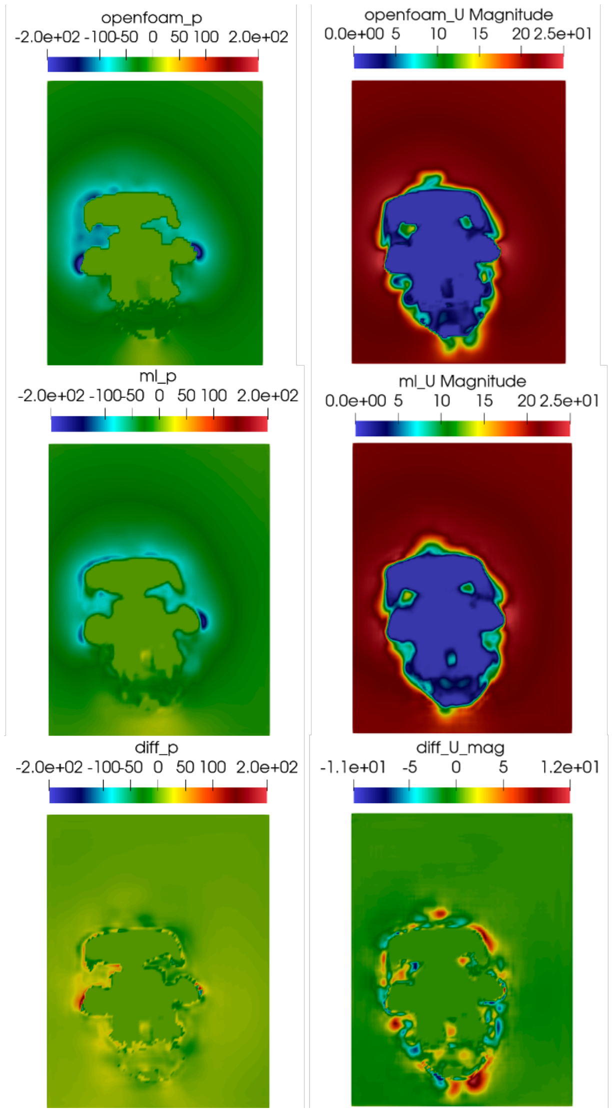

openfoam_p
-2.0e+02 -100 -50 0 50 100 2.0e+02
openfoam_U Magnitude
0.0e+00 5 10 15 20 2.5e+01
ml_p
-2.0e+02 -100 -50 0 50 100 2.0e+02
ml_U Magnitude
0.0e+00 5 10 15 20 2.5e+01
diff_p
-2.0e+02 -100 -50 0 50 100 2.0e+02
diff_U_mag
-1.1e+01 -5 0 5 1.2e+01

Figure 16: Differences in pressure (*left*) and velocity magnitudes (*right*) for CFD (*left*), U-Net (*middle*) and error (*bottom*) on a front-on view slice of the Motorbike dataset.

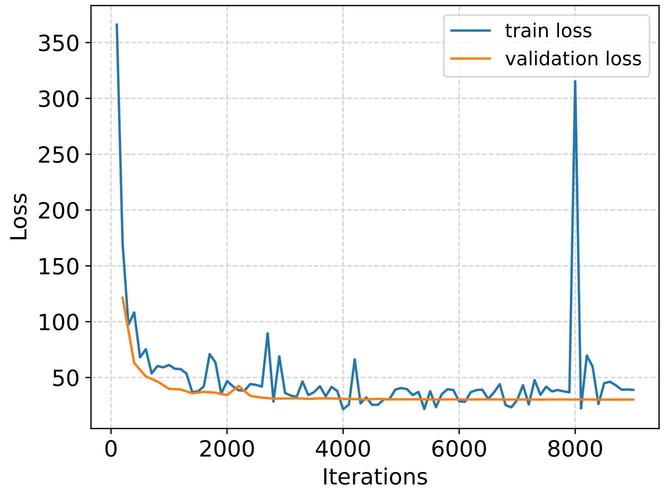


Figure 17: Training losses and validation scores for the pressure p during iterations for the U-Net model on the motorbike dataset.

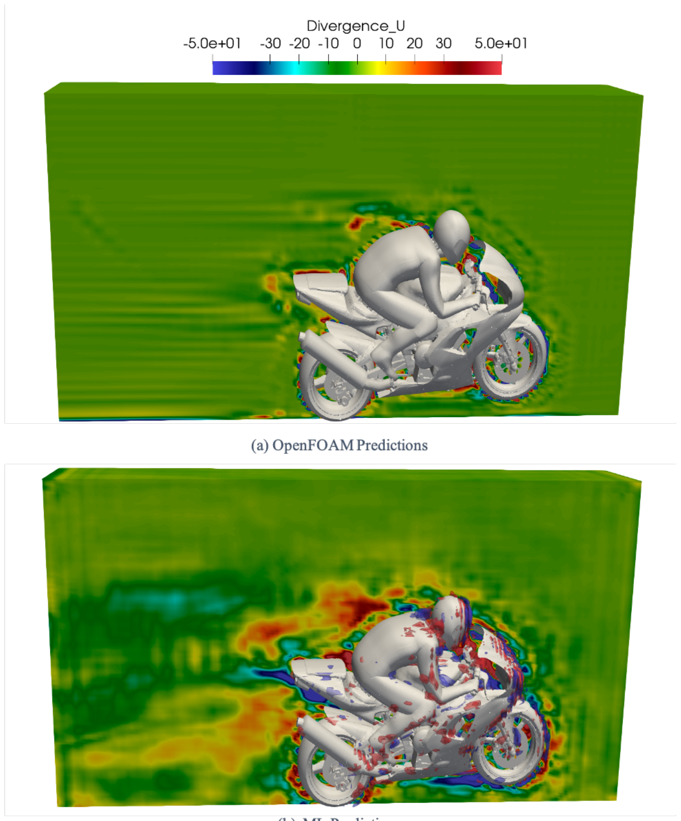


Figure 18: Divergence of velocity shows that the unconstrained U-Net predictions violate the incompressibility condition on the Motorbike dataset. Note that there are numerical artefacts due to the spatial interpolation of the OpenFOAM solution onto a structured grid and first-order finite differences used for calculating the divergence.