SURF: A Generalization Benchmark for GNNs Predicting Fluid Dynamics

Stefan Künzli
ETH Zurich
skuenzli@ethz.ch

Florian Grötschla*
ETH Zurich
fgroetschla@ethz.ch

Joël Mathys*
ETH Zurich
jmathys@ethz.ch

Roger Wattenhofer
ETH Zurich
wattenhofer@ethz.ch

Abstract

Simulating fluid dynamics is crucial for the design and development process, ranging from simple valves to complex turbomachinery. Accurately solving the underlying physical equations is computationally expensive. Therefore, learning-based solvers that model interactions on meshes have gained interest due to their promising speed-ups. However, it is unknown to what extent these models truly understand the underlying physical principles and can generalize rather than interpolate. Generalization is a key requirement for a general-purpose fluid simulator, which should adapt to different topologies, resolutions, or thermodynamic ranges. We propose SURF, a benchmark designed to test the *generalization* of learned graph-based fluid simulators. SURF comprises individual datasets and provides specific performance and generalization metrics for evaluating and comparing different models. We empirically demonstrate the applicability of SURF by thoroughly investigating the two state-of-the-art graph-based models, yielding new insights into their generalization. SURF is available under https://github.com/s-kuenzli/surf-fluidsimulation.

1 Introduction

Fluid simulations have established themselves as indispensable tools for addressing intricate and complex tasks in design and development processes. They play a central role in the construction of Formula 1 cars, where fluid simulations are extensively harnessed to refine external aerodynamics and enhance the performance of combustion engines [1]. Moreover, they enable the accurate simulation of scenarios that are too expensive or even infeasible to conduct in real-world experiments, such as the examination of atmospheric re-entry vehicles [2]. The application of fluid simulations extends even to domains such as weather prediction [3].

The underlying physics governing these phenomena can be modeled by the Navier-Stokes equations. Unfortunately, no closed-form solution is known, and they remain one of the infamous "Millennium Problems" [4]. Thus, computationally expensive iterative numerical solutions of the non-linear partial differential equations are used. The ongoing development of simulation software is geared towards delivering high-quality outcomes in the shortest feasible time frame [5]. Here, learning-based approaches have the potential to accelerate this development by either directly predicting the simulated process or providing a better initial solution that can be refined by classical iterative solvers [6]. Often, this comes with a direct tradeoff between guaranteed precision and rapid processing time.

Due to the common modeling of the simulation environment as irregular meshes, Graph Neural Network (GNN) based approaches have gained substantial traction recently. There, the main research focus lies on *how accurately GNNs can predict* the flow of fluid dynamics. The aim is to create a learned general-purpose fluid simulator. Given the wide range of applications of such a simulator, one of its most essential abilities is to generalize and accurately predict the fluid dynamics in novel and unseen environments as they might arise in the aforementioned development processes. This poses a significant challenge, as training on all possible environments or scenarios is simply infeasible.

*Equal contribution.

S. Künzli et al., SURF: A Generalization Benchmark for GNNs Predicting Fluid Dynamics. *Proceedings of the Second Learning on Graphs Conference (LoG 2023)*, PMLR 231, Virtual Event, November 27–30, 2023.

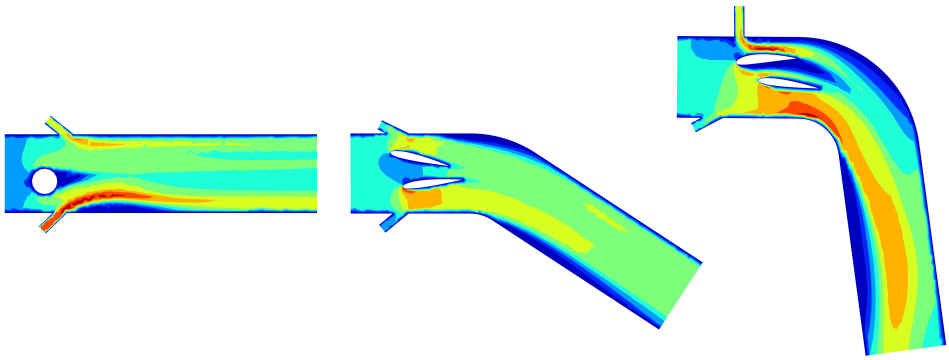

Figure 1: Velocity visualization for three datapoints of the SURF benchmark from SURF-Base, SURF-Topology, and SURF-Full dataset, respectively. The datasets are designed to measure a specific aspect of generalization, e.g., how well models can adapt to new objects, such as different air foils or unseen topologies with turns. Values are displayed contoured to make visual comparisons easier.

As a consequence, such a model has to truly grasp the underlying physical principles in order to generalize rather than only interpolate between observed training data points. To be more precise, such a simulator should be able to adapt to different topologies, resolutions, and thermodynamic ranges. Therefore, we want to address the question *how well GNNs can generalize* to different fluid dynamics environments.

We propose SURF, a benchmark for the simulation of fluid dynamics that focuses explicitly on generalization capabilities to evaluate how well a model grasps the underlying physical principles. SURF contains a diverse set of training and benchmarking datasets, with the explicit aim of adapting to (1) finer mesh resolution, (2) different parameter ranges, (3) topologies, and (4) dynamic simulation environments. To quantify the generalization, we propose a new set of metrics, the SURF generalization scores, measuring the performance of each mentioned aspect. They intend to capture the relative performance loss of a model due to not training on an unseen scenario where it has to rely on its ability to generalize. To demonstrate the utility and applicability of SURF, we empirically evaluate two state-of-the-art models, MGN [7] and EAGLE [8], for mesh-based fluid simulation.

We propose the SURF benchmark to systematically evaluate the ability of learned mesh-based simulators to generalize across the demanding environments of fluid dynamics. SURF's main contributions can be summarized as follows:

- We introduce seven new large-scale datasets to simulate the fluid dynamics behavior. Each dataset is constructed to change specific aspects of the environment to assess generalization performance across different datasets.
- We propose the SURF generalization scores. A set of new metrics which quantitatively capture the ability of a proposed model architecture to generalize to overall or specific changes across mesh resolution, parameter ranges, topologies, or simulation dynamics. This is complemented by the SURF performance score, which can assess the prediction quality of a model.
- We investigate what aspects of the training process help models to generalize by identifying how a training data set should be built such that the generalization of a model improves.
- We conduct a thorough empirical evaluation and comparison of two state-of-the-art GNN-based fluid simulators concerning their ability to generalize.

2 Related Work

In recent years there has been much interest in deploying machine learning for fluid simulation. The different approaches can be divided into the following categories: particle-based, grid-based, mesh-based and operator learning methods. The particle and grid methods are described in the Appendix A, whereas the mesh-based and operator learning methods are described in the following.

Mesh-Based Methods. In industrial contexts, mesh-based fluid dynamics solvers are vital for simulating complex fluid behavior. Due to their ability to handle complex geometries with irregular

meshes. Renowned solvers like ANSYS Fluent and Star-CCM+ or the open-source solver OpenFoam are extensively employed. In aerospace engineering, they are used to optimize aircraft designs for enhanced efficiency and safety. The automotive industry relies on them to simulate airflow around vehicles, reducing drag and improving fuel efficiency. These mesh-based solvers underpin industrial innovation by providing insights into fluid dynamics phenomena crucial for optimal design and performance across diverse sectors.

Meshes directly translate to graph representations, making them amenable to Graph Neural Networks (GNNs). In this context, mesh nodes and edges correspond to graph nodes and undirected edges within the graph structure. Graph Element Networks [9] use this structure, while works that combine graph structures with PDEs also exist [10]. The most relevant architectures for our evaluation are MeshGraphNets (MGN) [7] and EAGLE [8].

MGN [7] follows an encode-process-decode architecture and undergoes testing across diverse physical domains, including scenarios such as 2D incompressible fluid flow around a cylinder or cloth simulation. An extensive series of 1200 distinct simulations are conducted to compile a comprehensive dataset. Systematically varied parameters like inlet velocity, cylinder position, and radius are explored within each simulation, spanning 600 discrete time steps.

EAGLE [8] improves upon this approach by introducing enhancements such as node clustering, graph pooling, and global attention mechanisms. These refinements strengthen the model's ability to capture global dependencies. Simultaneously, a dataset featuring 1,184 simulations depicting 2D unsteady fluid flow originating from a mobile flow source across 600 distinct fluid domains was generated.

Operator Learning. A different approach for predicting solutions of partial differential equations like the Navier Stokes equations are machine learning methods which learn an operator mapping from the input functions to the solutions [11]. Several different neural architectures are investigated for learning operators. Fourier neural operator (FNO) [12] learns the operator in spectral space or transformer models [13] which are based on attention mechanism. Nevertheless, the application of these methods to real-world problems is challenging. For example methods like FNO using Fast Fourier Transform require a uniform regular grid. These challenges are addressed in recent publications. Geo-FNO [14] extends FNO by learning a mapping from an irregular mesh to a uniform mesh. Nevertheless, the adaption to general topologies remains a challenge. A General Neural Operator Transformer (GNOT) [15] for Operator Learning uses heterogeneous normalized attention layer, which is able to handle multiple input functions and irregular meshes.

Fluid Simulation Datasets. Both MGN and EAGLE focus their evaluation on synthetic in-distribution datasets but also consider generalization as an essential quality that is discussed further. EAGLE investigates generalization to different ground topologies and mesh resolutions. MGN changes mesh sizes and tests on airfoils with steeper angles or with higher inflow speeds. More synthetic datasets for the purpose of learning were introduced [16, 17], for example for turbulence simulation [18]. The PDEBench [19] provides solutions to a wide variety of partial differntial equations (PDE), including a single dataset for the Navier-Stokes equation in a unit square with random initial velocities and forcing terms.

None of these works offer dedicated datasets or comprehensive scores designed to compare models for this specific purpose. Our evaluation bridges this gap, creating a unified benchmark to quantitatively analyze their respective generalization capabilities. In addition to velocity and pressure, we add temperature as a solution variable. This enables the models to calculate the fluid-solid heat transfer, an important process for optimizing heat exchangers or analyzing electronic cooling. This also allows the simulation of different thermal fluid properties resulting in different heat propagation in the fluid flow.

3 Preliminaries

Mesh Generation. The geometry is built using Ansys SpaceClaim 2023 R1 [20]. Two different meshes are created with Ansys Meshing [21]. The fine mesh consists mostly of quadrilaterals and a small amount of triangles. At the wall boundaries, an inflation layer with eight elements is defined for a better resolution of the gradient. This mesh is then used for the ground truth calculation. The coarse mesh, consisting of triangles, is used for training and predictions (see Figure 2). The exact meshing

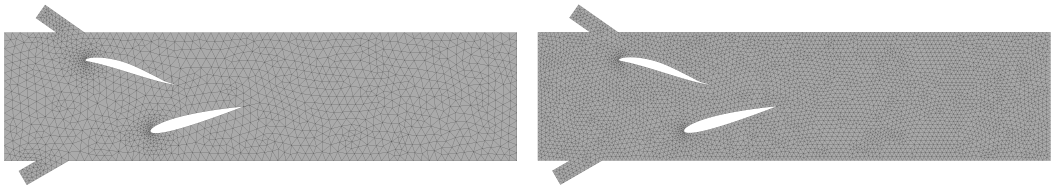

Figure 2: A mesh for a regular datapoint is visualized on the left, while the right datapoint depicts an instance from SURF-Mesh, with an increased mesh resolution by a factor of two.

algorithms used in Ansys Meshing are proprietary, but the triangular meshing of 2D geometries is a well-studied subject [22]. The algorithms differ in the resulting mesh quality (the minimal angle of the triangle should be maximized), the number of created triangles for a certain geometry, and the generation time. Similar meshes could also be obtained with the open source mesher presented in [23]. The result values (velocities, pressure, temperature) at the nodes of the coarse mesh are interpolated from the nodes of the fine mesh using linear interpolation on a triangular grid.

Root Mean Square Error (RMSE). To compare the predictions of a given model, the root mean square error (RMSE) is calculated for velocity, pressure, and temperature. The RMSE for the velocity v for a time horizon H on the test set $\mathcal{D}$ is calculated as:

$$\text{RMSE}_v(x, y) = \frac{1}{|\mathcal{D}|H} \sum_{d}^{\mathcal{D}} \sum_{t=1}^{H} \frac{1}{\sqrt{2}N_d} \sum_{v}^{N_d} ||\mathbf{x}_v^t - \mathbf{y}_v^t||_2$$

Where $\mathbf{x}_v^t$ is the predicted velocity vector at time step t at node v and $\mathbf{y}_v^t$ the corresponding ground truth value. N_d denotes the number of nodes of the datapoint d. The RMSE for pressure p and temperature t are calculated analogously.

4 SURF Benchmark

We introduce SURF, a 2D fluid dynamics benchmark specifically tailored to assess generalization. The benchmark consists of seven new large-scale datasets accompanied by a custom set of metrics to quantify performance and generalization of proposed model architectures. In this section, we describe the design and generation process of the individual datasets and their intended purpose. Then, we define the SURF performance and generalization scores, which allow for a comprehensive evaluation and comparison against other baselines. Our dataset is publically available online[2] with the accompanying code[3] that includes evaluation scripts.

4.1 SURF Datasets

Each of the seven SURF datasets is a large-scale collection of 2D fluid flow simulations with at least 1200 individual datapoints. They are accompanied by dedicated splits so that 80 percent are used for training and 10 percent for validation and test data each. The ground truth for each simulation is computed using ANSYS Fluent [24], a mesh-based solver using the finite volume method [25]. Afterward, the solution calculated on the fine mesh is downsampled using the linear interpolation described in Chapter 3. Every datapoint consists of 300 timesteps and about 1800 nodes that comprise the mesh topology. For the individual parameters, which determine the generation of each dataset, we refer to the Appendix. In the following, we outline the main differences between each of the individual datasets.

SURF-Base. This dataset serves as a basis for all other SURF datasets. It comprises simulated data of 2D flow around a single cylinder which can vary in size with two additional inlets, parametrized through position, angle, inflow velocity, and temperature.

[2] https://huggingface.co/datasets/SURF-FluidSimulation/FluidSimulation
[3] https://github.com/s-kuenzli/surf-fluidsimulation

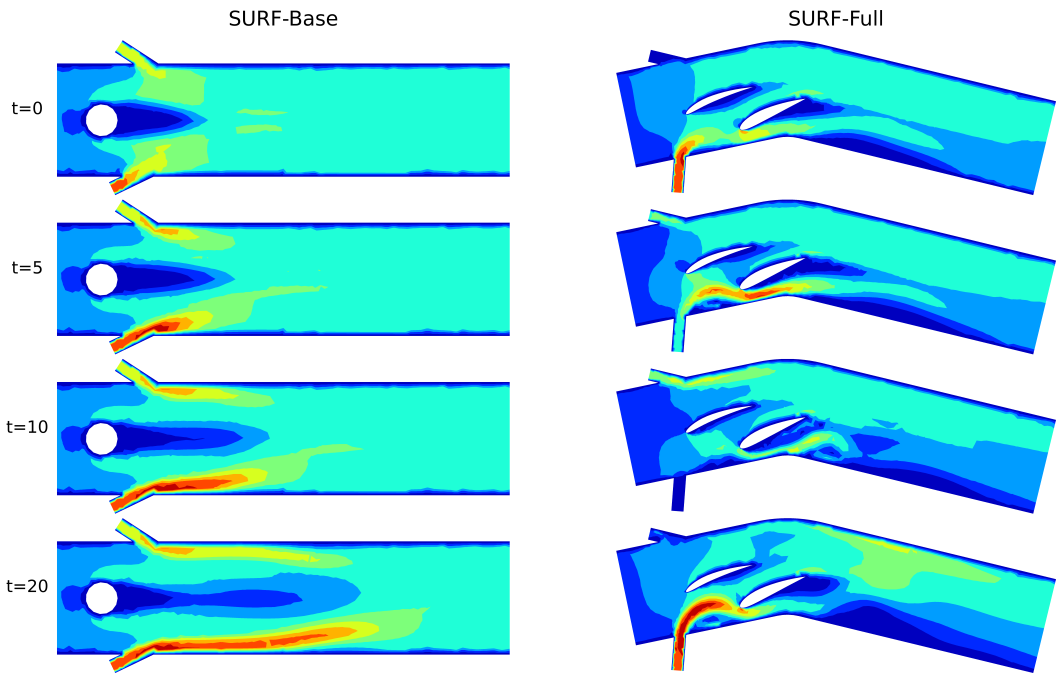

Figure 3: Velocity visualization for ground-truth data from SURF-Base (left) and SURF-Full (right), evaluated at timesteps 0, 5, 10, 20 (top to bottom). Note that on the right, the bottom inlet is active in the beginning, then turned off and turned on again in the last step, leading to a more dynamic simulation environment. Values are displayed contoured to make visual comparisons easier.

SURF-Rotated. The complete computational domain is rotated at an angle of [0°, 360°]. This ensures equal velocity distribution in all directions instead of the predominant velocity in the positive x-direction.

SURF-Range. Compared to the base dataset, the range of most simulation parameters is increased and decreased by a factor of two, including inlet velocity, object wall temperature, the radius of the cylinder, thermal conductivity, and heat capacity. In particular, this impacts the range of the calculated Reynolds number.

SURF-Topology. Two objects are present instead of a single cylinder. Each object is either a cylinder or one of five different airfoils, which vary in their angle of attack and can redirect the flow. The domain is equipped with an elbow angle from 0° to 90°. A flow separation may occur due to the velocity and the elbow angle.

SURF-Dynamic. Both small inlets vary their velocities over time. This introduces more variation of the calculated flow because, without the transient boundary conditions, the flow can converge to a steady state. Instead, the flow varies during the complete time horizon of the simulation.

SURF-Full. The superset of all other dataset generation parameters, except for datapoints with higher mesh resolutions. Therefore, it includes more temporal dynamics, larger ranges, multiple objects with diverse airfoils, and an elbow in the domain. Note that it contains datapoints not present in the other datasets as it is a superset of the generation parameters rather than the actual datapoints.

SURF-Mesh. Generated the same way as the full dataset, but the ground truth is downsampled to a finer mesh. The element size is reduced by a factor of two.

4.2 SURF Metrics

To measure the prediction quality of models on the SURF benchmark, we define a performance score PS. It follows the convention of the N-RMSE metric introduced in [8] and extends it to include

temperature. The time horizon is always fixed at 250 timesteps.

$$\text{SURF}_{\text{PS}} = \frac{1}{3}(\text{PS}_v + \text{PS}_p + \text{PS}_t) = \frac{1}{3}\left(\frac{\text{RMSE}_v}{\sigma_v} + \frac{\text{RMSE}_p}{\sigma_p} + \frac{\text{RMSE}_t}{\sigma_t}\right)$$

Where σ_v denotes the standard deviation of the velocity on the respective dataset. Unless explicitly stated otherwise, the performance score is always reported on the SURF-Full dataset.

The main aim of the SURF Benchmark is to quantify the ability of a model architecture to generalize across different environment settings. We measure generalization as the relative decrease in the performance of a model on the respective dataset. The idea is to train a model on an origin dataset and then test how well the performance translates to a different target dataset that differs in one aspect of the environment. Formally, let $\text{GS}_v(\Phi, \mathcal{D}_1, \mathcal{D}_2)$ denote the velocity generalization of model Φ across datasets $\mathcal{D}_1$ and $\mathcal{D}_2$.

$$\text{GS}_v(\Phi, \mathcal{D}_1, \mathcal{D}_2) = \frac{\text{RMSE}_v\left(\Phi_{\mathcal{D}_1}(\mathcal{D}_2), \mathcal{D}_2\right)}{\text{RMSE}_v\left(\Phi_{\mathcal{D}_2}(\mathcal{D}_2), \mathcal{D}_2\right)}$$

Where $\Phi_{\mathcal{D}_1}(\mathcal{D}_2)$ denotes the prediction for the test set of $\mathcal{D}_2$ made by model Φ trained on the train set $\mathcal{D}_1$. The GS scores for temperature and pressure are calculated analogously. Moreover, the full generalization score is the average over all physical quantities:

$$\text{GS} = \frac{1}{3}(\text{GS}_\text{v} + \text{GS}_\text{p} + \text{GS}_\text{t})$$

We identify four aspects fundamental to the ability to generalize: mesh resolution, parameter ranges, topologies, and dynamic simulation environments. Therefore, for each of these, we define a corresponding SURF generalization score, indicating how well the generalization of a single aspect of the environment has been learned:

$$\begin{aligned}
\text{SURF}_{\text{GS-Mesh}}(\Phi) &= \text{GS}(\Phi, \text{SURF-Full}, \text{SURF-Mesh}) \\
\text{SURF}_{\text{GS-Topology}}(\Phi) &= \text{GS}(\Phi, \text{SURF-Base}, \text{SURF-Topology}) \\
\text{SURF}_{\text{GS-Range}}(\Phi) &= \text{GS}(\Phi, \text{SURF-Base}, \text{SURF-Range}) \\
\text{SURF}_{\text{GS-Dynamic}}(\Phi) &= \text{GS}(\Phi, \text{SURF-Base}, \text{SURF-Dynamic})
\end{aligned}$$

For assessing how well a model can generalize to different topologies, such as handling new objects, we can quantify this using the $\text{SURF}_{\text{GS-Topology}}$ score. To simplify evaluation and comparison across different models, we define the SURF_{GS} score to capture the generalization capabilities of all aspects as an average over the more specialized generalization scores.

$$\text{SURF}_{\text{GS}} = \frac{1}{4}\left(\text{SURF}_{\text{GS-Mesh}} + \text{SURF}_{\text{GS-Topology}} + \text{SURF}_{\text{GS-Range}} + \text{SURF}_{\text{GS-Dynamic}}\right)$$

5 Evaluation

We empirically evaluate two state-of-the-art models for fluid simulation, MeshGraphNets (MGN) [7] and EAGLE [8]. While MGN uses a message-passing Graph Neural Network to compute embeddings directly on the given mesh, EAGLE adds a global attention mechanism to enable long-range information exchange. Both models undergo the SURF benchmark to assess their (1) baseline performance score, (2) generalization score, and (3) what training dataset generalizes best.

We adjust both architectures slightly to incorporate the temperature as a parameter, which was not used in the original architectures and thus assumed constant. We use the same loss function for both MGN and EAGLE. More information on the exact training setup can be found in Appendix D.1. All scores are reported as mean values over three runs.

5.1 Performance

To assess the overall accuracy and quality of the predictions, we compare the results of MGN and EAGLE, across three distinct SURF datasets: Base, Rotated, and Full. The performance scores for each model and dataset are summarized in Table 1.

Table 1: Performance scores for velocity, pressure, and temperature after 250 time steps of MGN and EAGLE trained and tested on the SURF-Base, SURF-Rotated, and SURF-Full dataset, respectively. Both models achieve similar performance, but EAGLE predicts the pressure dynamics more accurately than MGN.

Dataset	Model	$\mathbf{PS_v} \downarrow$	$\mathbf{PS_p} \downarrow$	$\mathbf{PS_t} \downarrow$	$\mathbf{PS} \downarrow$
Base	MGN	0.073 ± 0.005	0.142 ± 0.011	0.068 ± 0.003	0.094 ± 0.006
	EAGLE	0.074 ± 0.003	0.090 ± 0.004	0.097 ± 0.006	0.087 ± 0.004
Rotated	MGN	0.041 ± 0.002	0.133 ± 0.003	0.071 ± 0.002	0.082 ± 0.001
	EAGLE	0.061 ± 0.007	0.089 ± 0.005	0.107 ± 0.003	0.086 ± 0.004
Full	MGN	0.165 ± 0.099	0.168 ± 0.088	0.111 ± 0.041	0.148 ± 0.076
	EAGLE	0.163 ± 0.004	0.166 ± 0.011	0.146 ± 0.015	0.158 ± 0.006

Table 2: Generalization scores after 250 time steps of the MGN and EAGLE model trained on the SURF-Base, SURF-Rotated, and SURF-Dynamic and then tested on the test set SURF-Topology, SURF-Range, SURF-Dynamic, and SURF-Full. Note that this is different from the SURF generalization scores.

Training Set	Model	Topology $\downarrow$	Range $\downarrow$	Dynamic $\downarrow$	Full $\downarrow$
Base	MGN	3.10 ± 0.72	1.30 ± 0.07	5.76 ± 0.64	3.32 ± 1.29
	EAGLE	3.68 ± 0.10	1.08 ± 0.09	2.17 ± 0.08	3.73 ± 0.52
Rotated	MGN	1.81 ± 0.38	1.54 ± 0.18	6.73 ± 0.60	1.95 ± 0.72
	EAGLE	1.96 ± 0.08	1.36 ± 0.22	2.41 ± 0.17	1.47 ± 0.06
Dynamic	MGN	3.11 ± 0.56	1.52 ± 0.23	1.00 ± 0.00	3.93 ± 1.52
	EAGLE	4.47 ± 0.24	2.16 ± 0.33	1.00 ± 0.00	5.25 ± 1.76

Velocity. Across all datasets, both MGN and EAGLE models demonstrate comparable performance in predicting velocity. Their scores consistently remain close, indicating a strong and balanced predictive capability in this aspect.

Pressure. EAGLE showcases a notable advantage in pressure prediction across the board. It consistently scores better than MGN, suggesting that EAGLE is better equipped to predict pressure-related outcomes accurately.

Temperature. For temperature prediction, the MGN model emerges as the more proficient one. Across all datasets, MGN consistently outperforms EAGLE in this aspect, suggesting a greater aptitude for capturing temperature-related patterns.

Overall Assessment. Considering all velocity, pressure, and temperature scores together, there is no clear winner between MGN and EAGLE. Both models achieve comparable performance, with EAGLE excelling in pressure prediction and MGN performing better in temperature prediction. This balanced performance across diverse aspects underscores the comparable overall performance of the two models. Furthermore, the performance trends observed in the datasets remain consistent with the models' behavior. This consistency across datasets reinforces the stability of the introduced performance characteristics. Notably, the SURF-Full dataset consistently leads to the highest prediction loss, indicating that the increased complexity of this dataset might pose a challenge for both models.

5.2 Generalization

We investigate the ability of MGN and EAGLE models to generalize across individual aspects such as topologies, mesh resolution, parameter ranges, and simulation dynamics. We use the proposed SURF generalization scores defined in Section 4.2. Recall that the generalization score captures the

Table 3: The computed SURF generalization scores for MGN and EAGLE. While MGN can generalize better to different topologies, EAGLE achieves better generalization on mesh resolution, simulation dynamics, and parameter ranges, resulting in an overall better SURF_{GS} score.

Model	$\text{SURF}_{\text{GS-Mesh}} \downarrow$	$\text{SURF}_{\text{GS-Topology}} \downarrow$	$\text{SURF}_{\text{GS-Range}} \downarrow$	$\text{SURF}_{\text{Dynamic}} \downarrow$	$\text{SURF}_{\text{GS}} \downarrow$
MGN	1.07 ± 0.26	3.10 ± 0.72	1.30 ± 0.07	5.76 ± 0.64	2.81 ± 0.23
EAGLE	1.01 ± 0.05	3.68 ± 0.10	1.08 ± 0.09	2.17 ± 0.08	1.98 ± 0.06

relative drop in performance due to a model not training on the target dataset and therefore having to rely on its ability to generalize.

The computed SURF generalization scores for each aspect are presented in Table 3. Our observation indicates that MGN exhibits better adaptability to unseen topologies compared to EAGLE. However, in an overall assessment, the EAGLE architecture demonstrates superior generalization regarding mesh resolution, parameter ranges, and simulation dynamics, which results in a lower overall SURF_{GS} score.

Impact of Training Dataset on Generalization. We explore what factors enhance training to yield better generalization results in the mentioned architectures. We analyze the GS scores, reflecting the efficacy of inter-dataset training-to-testing transfer. The results are shown in Table 2, and for the complete matrix, we refer to the appendix.

We reaffirm the conclusions drawn from the prior SURF generalization scores: EAGLE excels in generalization across all datasets, except Topology. Notably and maybe surprisingly, training on the SURF-Rotated dataset substantially improves generalization to the SURF-Full dataset for both MGN and EAGLE. Observing dynamic fluid boundary conditions modestly enhances generalization across datasets. Yet the most significant increase comes from the SURF-Rotated training. Recall that this dataset is a rotated version of the base dataset, and while training solely on rotated versions of the same dataset does not augment dataset size, introduce new topologies or dynamics, it remarkably amplifies generalization across all datasets. Illustrated in Figure 4, the model trained on rotated data excels in predicting fluid flow in adapted curved tunnels and amidst more challenging wing profiles. In contrast to a cylinder, a wing profile can redirect the flow. Additionally, no flow separation occurs for low attack angles of an airfoil. With an increasing angle of attack, a flow separation may occur. Correctly predicting the flow separation or a stall is important for calculating the lift of an aircraft because a flow separation leads to a sudden loss of lift.

6 Limitations

The presented benchmark only considers incompressible flows, i.e., the fluid density is assumed to be constant over time and domain. This assumption is reasonable for flows where the velocity is less than a third of the speed of sound of the fluid. For dry air at a temperature of 20°C, the speed of sound is 340 m/s. Hence, air flows with a speed of less than 100m/s can be considered incompressible. Further, the flow in the dataset is always turbulent; there is no laminar flow in the dataset because if laminar flows were included, the velocity range would become very large (>factor 1000). For these velocity ranges, the training is expected to be challenging. Additionally, the flow characteristics are different, further increasing the difficulty for a neural network to learn both fluid regimes.

Another limitation presents itself in the fact that the datasets cannot capture all configurations that could show up in practice. While the presented metrics are well-suited to judge generalization capabilities to different domains, they can only serve as an approximation of how well these models generalize to real-world applications with variations that are not included in our testing. That being said, we expect that our metrics give a good and valuable indication of the generalization power of the tested model.

Future Work. The proposed benchmark serves as a starting point for future research to test machine learning methods predicting fluid flows for generalization and overall performance. A possible future direction for the practical applications of learned neural networks on fluid flow predictions could be the application of these architectures to three-dimensional geometries. The biggest hurdle to

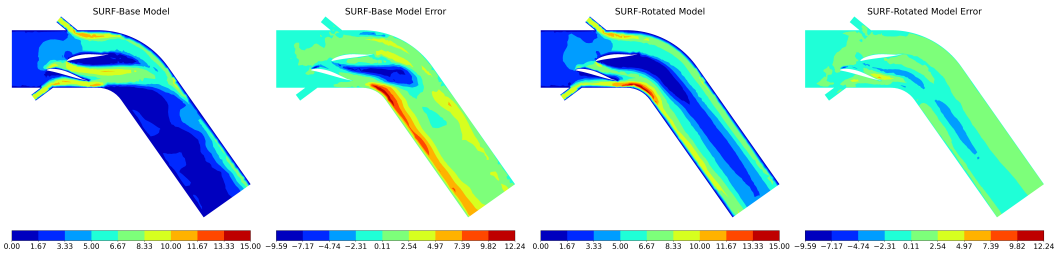

Figure 4: When training a model on the augmented dataset SURF-Rotated (right) instead of the SURF-Base dataset (left), the model exhibits better generalization to new environments and can even predict the flow between objects such as wing profiles more precisely, even though it has never encountered these during training. Values are displayed contoured to make visual comparisons easier.

overcome is the high memory footprint, especially during training. For example, the MGN method used approximately 24 GB of GPU memory during training. By simply extruding the geometry from the SURF-Full dataset by 100mm, the number of nodes in the mesh increases by a factor of 10, therefore requiring considerably more memory. Thus, we face a new challenge for neural fluid simulators: Developing scalable architectures that can be applied to large (or high-resolution) and complex geometries. Another interesting area for future investigations lies in exploring a broader spectrum of architectural choices and potential factors that can impact generalization. This encompasses the characteristics of the training, architecture choices, or even continual learning. SURF stands as a versatile platform well-suited for delving into this research.

7 Conclusion

Fluid simulation is a critical building block in developing and optimizing cutting-edge products, such as turbomachinery, cars, and airplanes. Using machine-learning models to simulate fluid dynamics can lead to faster processing times or more precise results. Given the nature of where such simulators are deployed, it is infeasible to train on all possible scenarios. Therefore, the ability of a learned simulator to grasp the underlying physics and generalize is an essential aspect of judging a given simulator's quality and applicability to real-world scenarios. We introduce SURF, a dedicated benchmark consisting of seven datasets to assess a mesh-based model's performance and generalization capabilities. We present a set of new metrics, which can be used to test and quantify generalization to different domain topologies, dynamic boundary conditions, finer mesh resolution, or a wider range of simulation parameter values.

We demonstrate the applicability of SURF on two state-of-the-art models and perform a comparative analysis. Both models achieve similar performance regarding the overall simulation accuracy. The main objective of SURF is to assess the generalization abilities. Here, the generalization scores show that while MGN generalizes better to unseen topologies, EAGLE can adapt better to the other aspects and achieves a lower unified SURF_{GS} score. Interestingly, training on a rotated domain has a large positive impact on both models, an observation that can be used in future work to improve neural fluid simulators.

Our benchmark focuses on how well learned mesh-based models adapt to new fluid dynamic domains, a crucial property for general-purpose fluid simulators. To achieve this, we introduce seven new large-scale datasets. Alongside these datasets, we introduce a new set of metrics: the SURF generalization scores. These metrics offer a simple way to measure different aspects of generalization and allow for easy comparison between different approaches. Our testing clearly showcases the effectiveness of SURF in evaluating generalization ability. As we move forward, we anticipate that SURF will contribute to a better understanding of generalization and a simpler and unified evaluation within this field.

References

[1] W. Toet. Aerodynamics and aerodynamic research in formula 1. *The Aeronautical Journal*, 117 (1187):1–26, 2013. doi: 10.1017/S0001924000007739. 1

[2] Ioannis Nompelis, Graham Candler, and Raul Conti. A parallel implicit cfd code for the simulation of ablating re-entry vehicles. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, page 1562, 2009. 1

[3] Frank J Zajaczkowski, Sue Ellen Haupt, and Kerrie J Schmehl. A preliminary study of assimilating numerical weather prediction data into computational fluid dynamics models for wind prediction. *Journal of Wind Engineering and Industrial Aerodynamics*, 99(4):320–329, 2011. 1

[4] Keith J. Devlin. *The millennium problems: the seven greatest unsolved mathematical puzzles of our time*. New York: Basic Books, [2002] 2002, 2002. URL https://search.library.wisc.edu/catalog/999935001502121. Includes bibliographical references (pages [229]-230) and index. 1

[5] Niveditha Krishnamoorthy Chris Porter. The computational fluid dynamics revolution driven by gpu acceleration, 2023. URL https://developer.nvidia.com/blog/computational-fluid-dynamics-revolution-driven-by-gpu-acceleration/. 1

[6] Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, volume 38, pages 71–82. Wiley Online Library, 2019. 1

[7] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. *CoRR*, abs/2010.03409, 2020. URL https://arxiv.org/abs/2010.03409. 2, 3, 6

[8] Steeven Janny, Aurélien Béneteau, Madiha Nadri, Julie Digne, Nicolas Thome, and Christian Wolf. Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers, 2023. 2, 3, 5, 6, 18

[9] Ferran Alet, Adarsh Keshav Jeewajee, Maria Bauza Villalonga, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Kaelbling. Graph element networks: adaptive, structured computation and memory. In *International Conference on Machine Learning*, pages 212–222. PMLR, 2019. 3

[10] Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *international conference on machine learning*, pages 2402–2411. PMLR, 2020. 3

[11] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *CoRR*, abs/1910.03193, 2019. URL http://arxiv.org/abs/1910.03193. 3

[12] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *CoRR*, abs/2010.08895, 2020. URL https://arxiv.org/abs/2010.08895. 3

[13] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning, 2023. 3

[14] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries, 2022. 3

[15] Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: A general neural operator transformer for operator learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 12556–12569. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/hao23c.html. 3

[16] Junfeng Chen, Elie Hachem, and Jonathan Viquerat. Graph neural networks for laminar flow prediction around random two-dimensional shapes. *Physics of Fluids*, 33(12), 2021. 3

[17] Xu Han, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Li-Ping Liu. Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113*, 2022. 3

[18] Kimberly Stachenfeld, Drummond B Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation. *arXiv preprint arXiv:2112.15275*, 2021. 3

[19] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning, 2023. 3

[20] 2023. Ansys® Student SpaceClaim 2023 R1, Ansys Inc. 3, 14

[21] 2023. Ansys® Student Meshing 2023 R1, Ansys Inc. 3, 14, 15

[22] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. 02 1970. doi: 10.1142/9789814355858_0002. 4

[23] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry. 4

[24] 2023. Ansys® Fluent, Release 2023 R1, Help System, Fluent Theory Guide, ANSYS, Inc. 4, 13

[25] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000. 4

[26] Joseph Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68: 1703, 07 2005. doi: 10.1088/0034-4885/68/8/R01. 13

[27] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks, 2020. 13

[28] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew M. Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL http://arxiv.org/abs/1806.01261. 13

[29] Zijie Li and Amir Barati Farimani. Graph neural network-accelerated lagrangian fluid simulation. *Computers & Graphics*, 103:201–211, 2022. ISSN 0097-8493. doi: https://doi.org/10.1016/j.cag.2022.02.004. URL https://www.sciencedirect.com/science/article/pii/S0097849322000206. 13

[30] Youquan Liu, Xuehui Liu, and Enhua Wu. Real-time 3d fluid simulation on gpu with complex obstacles. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pages 247–256, 2004. doi: 10.1109/PCCGA.2004.1348355. 13

[31] Mitchell Kenny Zadick Johanne, Kenwright Benjamin. Integrating real-time fluid simulation with a voxel engine. *The Computer Games Journal*, 2016. 13

[32] Nils Thuerey, Konstantin Weissenow, Harshit Mehrotra, Nischal Mainali, Lukas Prantl, and Xiangyu Hu. Well, how accurate is it? A study of deep learning methods for reynolds-averaged navier-stokes simulations. *CoRR*, abs/1810.08217, 2018. URL http://arxiv.org/abs/1810.08217. 13

[33] 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*, KDD '16, page 481–490, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939738. URL https://doi.org/10.1145/2939672.2939738.

[34] Yao Zhang, Woong-Je Sung, and Dimitri Mavris. Application of convolutional neural network to predict airfoil lift coefficient, 2018.

[35] 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, Aug 2019. ISSN 1432-0924. doi: 10.1007/s00466-019-01740-0. URL https://doi.org/10.1007/s00466-019-01740-0. 13

[36] G. G. Stokes. On the Effect of the Internal Friction of Fluids on the Motion of Pendulums. *Transactions of the Cambridge Philosophical Society*, 9:8, January 1851. 14

[37] E. Sashi Menon. *Transmission Pipeline Calculations and Simulations Manual*. Elsevier, 2015. ISBN 978-1-85617-830-3. doi: https://doi.org/10.1016/C2009-0-60912-0. 14

[38] Kenny Q. Ye. Orthogonal column latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association*, 93(444):1430–1439, 1998. ISSN 01621459. URL http://www.jstor.org/stable/2670057. 14

[39] 2023. Ansys® Student Workbench 2023 R1, Ansys Inc. 14

[40] Steeven Janny, Aurélien Béneteau, Madiha Nadri, Julie Digne, Nicolas Thome, and Christian Wolf. Eaglemeshtransformer, 2023. URL https://github.com/eagle-dataset/EagleMeshTransformer. 18

A Extended Related Work

The particle-based and grid-based methods for fluid simulation are described in the following.

Particle-Based Methods. The particle-based approach to fluid simulations draws inspiration from smoothed particle hydrodynamics (SPH) as outlined in [26]. SPH represents a mesh-free Lagrangian technique in which particle interactions serve as approximations of fluid dynamics. This approach proves advantageous in modeling situations like free surface flows (such as ocean waves or partially filled pipes) and intricate fluid-solid interactions (like water splashes resulting from a car moving through a water puddle). Notably, these scenarios involve fluid domains where considerable changes occur, even affecting domain boundaries throughout the simulation. This inherent adaptability simplifies the simulation process, eliminating the need for mesh adaptation due to the mesh-free nature of the method. An example of the particle-based approach for fluid simulations with machine learning is "Graph Network-based Simulators" (GNS) [27]. The Graph Network framework [28] is applied, where particles are represented as nodes, and the interactions among the particles are modeled with edges. The dynamic behavior is replicated by learned message-passing on graphs. This approach was adapted by Li and Farimani [29], who represent the advection (viscosity and body forces), pressure, and collisions between particles through three separate graphs.

Grid-Based Methods. Grid-based fluid simulations are relatively fast and mainly used for real-time and visualization applications [30, 31]. Leveraging the grid structure facilitates optimal GPU utilization. Nonetheless, a notable drawback is the consistent resolution across the domain, which may give rise to issues concerning precision and the portrayal of small entities within the flow. Due to the native GPU hardware support, many works on using machine learning methods to predict physics are based on convolutional architectures on regular grids [32–35].

B Physical Background

This section gives a brief background of the physics phenomena governing fluid dynamics, mesh generation, and how the quality and accuracy of a prediction are quantified.

Fluid Simulation. The physical behavior of fluids is described by the Navier-Stokes equations, a set of partial differential equations. They specifically express the conservation of mass and the momentum balance. In this work, we restrict ourselves to incompressible 2D flows with Newtonian fluids. The velocity and pressure field completely defines the flow of an incompressible fluid at a certain point in time. Two quantities are preserved in the fluid flow: mass and momentum. The equation for the conservation of mass for an incompressible, 2D flow is given by [24]:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

Where u and v denote the velocity in the x-direction and y-direction, respectively. The momentum balance equations are:

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left[2\mu\frac{\partial u}{\partial x}\right] + \frac{\partial}{\partial y}\left[\mu\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right]$$
$$\rho\frac{\partial v}{\partial t} + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial y}\left[2\mu\frac{\partial v}{\partial y}\right] + \frac{\partial}{\partial x}\left[\mu\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right]$$

p denotes the pressure, whereas the fluid density and the kinematic viscosity are denoted by ρ and μ. For the calculation of the temperature, the heat transfer has to be calculated. This leads to the energy conservation equation:

$$\rho c_p\left[\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y}\right] = k\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right]$$

k and c_p denote the thermal conductivity and heat capacity, respectively. If the fluid velocities u and v are set to zero, one gets the equation for the heat transfer in solids. Hence, the terms weighted with the fluid velocities describe the convective heat transport.

Reynolds Number. Note that two incompressible flows with the same Reynolds number behave similarly and the velocity and pressure fields only differ by a constant factor. As a consequence, by scaling the solution, the simulation of a single fluid (e.g., water) can be translated to another medium (e.g., air). The Reynolds number is an important dimensionless quantity used to characterize fluid flows. The Reynolds number is the ratio between inertial and viscous forces [36]:

$$\mathrm{Re} = \frac{\rho u L}{\mu}$$

L is a characteristic linear dimension. For example, for a flow around a cylinder, the characteristic length is equal to the diameter of the cylinder, or for a flow in a pipe, it is equal to the diameter of the pipe.

The Reynolds number is often used to predict the transition from laminar to turbulent flow. For a pipe flow with $\mathrm{Re} < 2000$, the flow is classified as laminar, for $\mathrm{Re} > 4000$, the flow is classified as turbulent [37]. In between, the flow is classified as transitional. In laminar flows, the fluid particles follow smooth paths in layers. Particles close to a solid surface move in lines parallel to that surface. There are no swirls in the flow. In contrast, unsteady vortices across large spatial scales interact in turbulent flows.

C Dataset Generation

C.1 Parameters

The 2D geometry, in which the fluid flows, is built from the parameters shown in Figure 5. Small gaps can lead to numerical problems during the solution (meshing and stability of the solver). To prevent too small gaps, the y-position of each object is calculated using a $ObjectyFactor$ ranging from 0 to 1.

$$\begin{aligned} ObjectyPos = & minDistance + objectHeight + \\ & ObjectyFactor \cdot (DomainHeight - 2 \cdot objectHeight \\ & - 2 \cdot minDistance) \end{aligned} \tag{1}$$

Where $minDistance$ is set to 30mm and $objectHeight$ denotes the vertical size of the corresponding object. The transient velocities of Inlet2 and Inlet3 are defined according to:

$$Inletv(t) = InletvMean + InletvAmplitude \cdot \sin(2\pi \cdot InletvFrequency \cdot t) \tag{2}$$

The outlet boundary condition does not allow an influx. Therefore, if necessary, the amplitude of Inlet2 and Inlet3 is reduced to always ensure a net outflow through the outlet.

The data sets and the used range for each parameter are given in Table 4 and 5. Given the range of the cylinder radius and inlet velocity, the SURF-Base and SURF-Base data set cover a Reynolds Number range between 1000 and 126000.

C.2 Process for Generation of Data Sets

The process of generating a data set is outlined in Figure 2. The steps required and the software used in the process are:

- **Parameter Ranges:** Define the range of each parameter (see Table 5 and 4).
- **Latin Hypercube Sampling:** According to the defined range, Latin hypercube sampling [38] is used to generate the samples for the data set. Each sample corresponds to a simulation with a specific geometry and boundary conditions defined by the parameters. These samples are internally referred to as design points (DP). For a sample size of n the Latin hypercube algorithm divides each parameter range into n equally long sub-ranges and ensures that there is only one sample in each sub-range. For two parameters, this is similar to having n rooks on a nxn chessboard without threatening each other.

- **Create Geometry & Mesh:** The geometry is built running in Ansys SpaceClaim 2023 R1 [20]. The creation of the geometry and the mesh for each design point is done in Ansys Workbench 2023 R1 [39], and Ansys Meshing 2023 R1 [21]. Two different meshes are created with Ansys

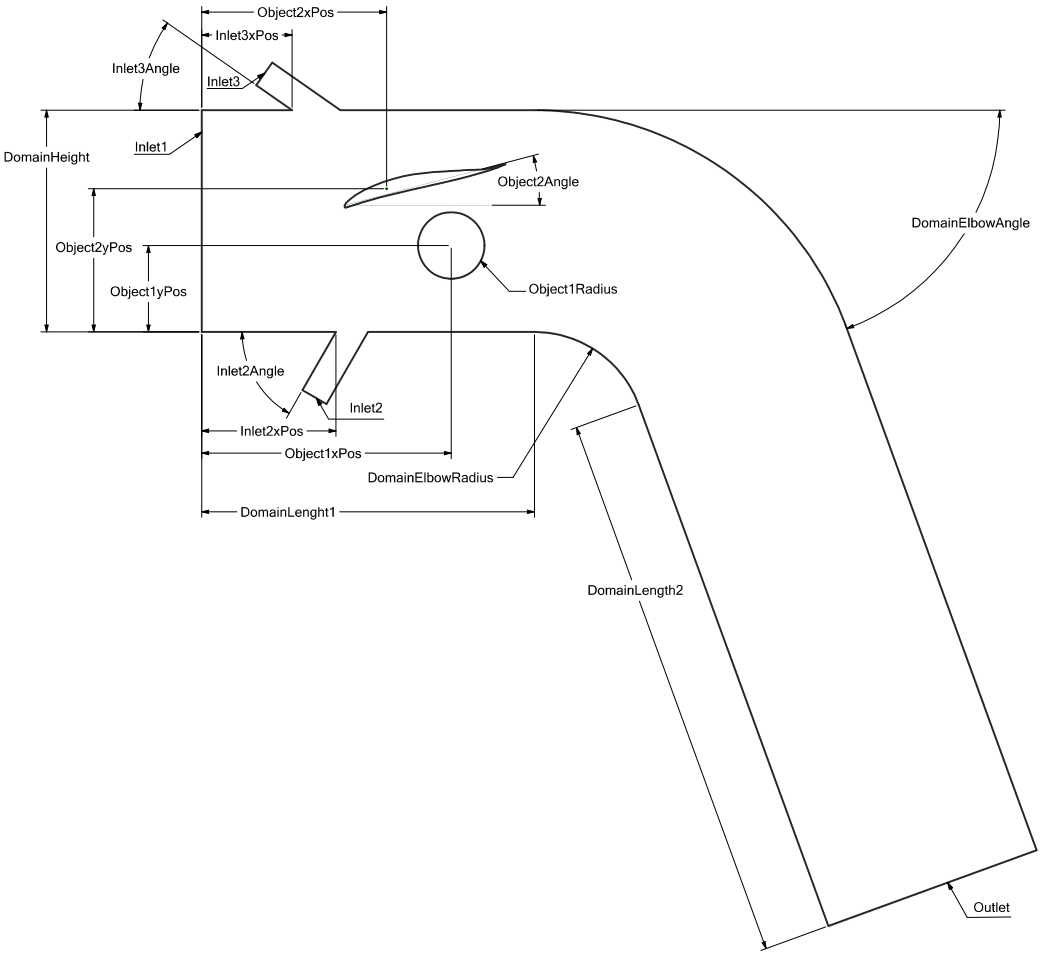

Figure 5: Sketch of how the individual geometry parameters influence the creation of the SURF dataset. Each dataset defines its own range for its generation parameter. For example, The SURF-Base dataset only allows for a cylinder while SURF-FULL can have up to two airfoils.

Meshing[21]. The fine mesh consists mostly of quadrilaterals and a small amount of triangles. Quadrilaterals are preferred because they are numerically more efficient (better solution quality with the same cost compared to triangles). At the wall boundaries, an inflation layer with eight elements is defined for a better gradient resolution. This mesh is used for the ground truth calculation. The coarse mesh, consisting of triangles, is used for training and predictions (cf. Figure 2). A coarse mesh must be used due to the GPU memory footprint of the training and to ensure that the training can still be run on an Nvidia RTX 3090 GPU (24 GB).
Software: Ansys SpaceClaim 2023 R1, Ansys Workbench 2023 R1, Ansys Meshing 2023 R1

- **Prepare for execution:** The fluid simulation setup is defined in a journal file for Ansys Fluent (FluentSimulation.jou). In this file, the simulation settings, boundary conditions, and fluid parameters are defined. We then run the simulation on a compute cluster.
- **Run:** Each simulation is run as a separate job, requesting two cores and 2 GB of memory. A transient incompressible 2D simulation is run with a fixed time step of 0.01s for 300 time steps. The calculated results on the fine mesh are then mapped to the coarse mesh. For the downsampling linear triangular interpolation is used. Due to numerical inaccuracies, some nodes lie outside of the domain of the mesh. For these nodes, the nearest neighbor interpolation is used. The final data structure for the training data set was taken from the Eagle paper. For each DP there is a folder with the two files sim.npz and triangles.npy. sim.npz contains the node coordinates, velocities, pressure, and temperature for each calculated time step. Triangles.npy

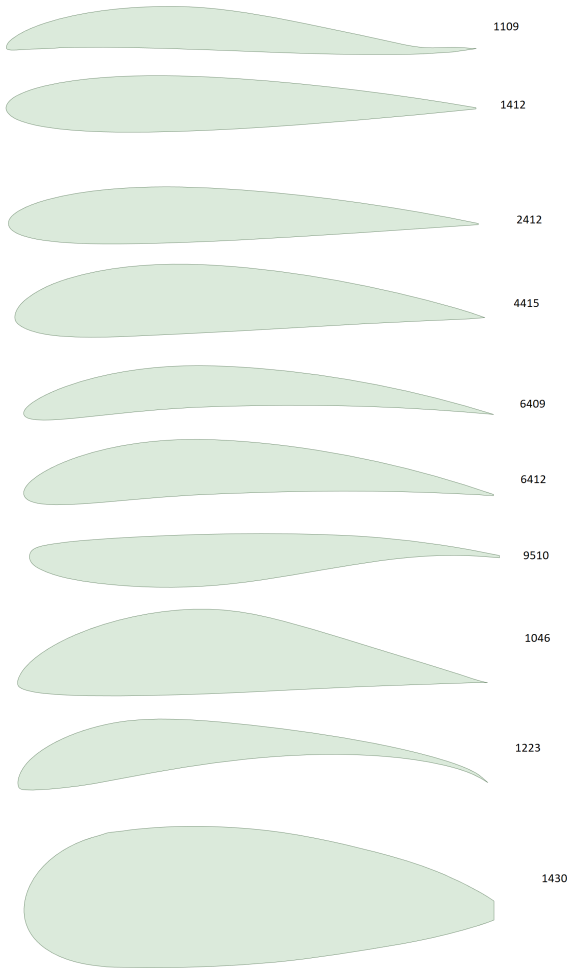

Figure 6: Different airfoil shapes and their corresponding number.

contains the edges connecting the nodes.
Software: Python 3.8.5 and Ansys Fluent 2021 R1

C.3 Dataset Statistics

Before downsampling, each data set has a size of about 2 TB. The size of the downsampled data sets used for the training is between 37 and 53 GB. The calculation of one simulation takes between 7 - 30min to finish. Each dataset is divided into a training, validation, and test set, by randomly shuffling the simulations and selecting 80% of the simulations for the training set and 10% for the validation and training set respectively. Table 7 shows statistics for the main parameters of the generated dataset. Key characterics of the datasets are given in Table 6.

D Baselines

Two Graph Neural Network models are trained on the presented data sets. Because the training data also includes the temperature as a state value, the models are slightly adapted to predict the temperature field in the flow as well.

Parameter	Base	Rotated	Range	Topo	Dynamic	Full
DomainLength	1600	1600	1600	1600	1600	1600
DomainHeight	400	400	400	400	400	400
DomainElbowAngle	0	0	0	[0, 90]	0	[0, 90]
DomainElbowRadius	0	0	0	200	0	200
DomainOrientation	0	[0, 360]	0	0	0	[0, 360]
Inlet2xPos	[150, 450]	[150, 450]	[150, 450]	[150, 450]	[150, 450]	[150, 450]
Inlet2Angle	[20, 45]	[20, 45]	[20, 45]	[20, 90]	[20, 45]	[20, 90]
Inlet3xPos	[150, 450]	[150, 450]	[150, 450]	[150, 450]	[150, 450]	[150, 450]
Inlet3Angle	[20, 45]	[20, 45]	[20, 45]	[20, 45]	[20, 45]	[20, 45]
Object1Type	Cylinder	Cylinder	Cylinder	Cylinder & 5 Airfoils	Cylinder	Cylinder & 10 Airfoils
Object1xPos	[150, 450]	[150, 450]	[150, 450]	[150, 450]	[150, 450]	[150, 450]
Object1yFactor	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 1]
Object1Angle	n/a	n/a	n/a	[-15, 15]	n/a	[-30, 30]
Object1Radius	[45, 75]	[45, 75]	[30, 90]	[45, 75]	[45, 75]	[30, 90]
Object2Type	n/a	n/a	n/a	Cylinder & 5 Airfoils	n/a	Cylinder & 10 Airfoils
Object2xPos	n/a	n/a	n/a	[150, 450]	n/a	[150, 450]
Object2yFactor	n/a	n/a	n/a	[0, 1]	n/a	[0, 1]
Object2Angle	n/a	n/a	n/a	[-15, 15]	n/a	[-30, 30]
Object2Radius	n/a	n/a	n/a	[45, 75]	n/a	[45, 75]

Table 4: The geometry parameter ranges for the generation of the different SURF datasets.

Parameter	Base	Rotated	Range	Topo	Dynamic	Full
Inlet1v	[1, 10]	[1, 10]	[0.5, 20]	[1, 10]	[1, 10]	[0.5, 20]
Inlet2vMean	[1, 10]	[1, 10]	[1, 10]	[1, 10]	[1, 10]	[1, 10]
Inlet2vAmplitude	0	0	0	0	[0, 10]	[0, 10]
Inlet2vFrequency	0	0	0	0	[1, 5]	[1, 5]
Inlet2T	[290, 310]	[290, 310]	[290, 310]	[290, 310]	[290, 310]	[290, 310]
Inlet3vMean	[1, 10]	[1, 10]	[1, 10]	[1, 10]	[1, 10]	[1, 10]
Inlet3vAmplitude	0	0	0	0	[0, 10]	[0, 10]
Inlet3vFrequency	0	0	0	0	[1, 5]	[1, 5]
Inlet3T	[290, 310]	[290, 310]	[290, 310]	[290, 310]	[290, 310]	[290, 310]
Object1T	[450, 800]	[450, 800]	[375, 1300]	[450, 800]	[450, 800]	[375, 1300]
Object2T	n/a	n/a	n/a	[450, 800]	n/a	[375, 1300]
Thermal Conductivity	[0.0258, 0603]	[0.0258, 0603]	[0.013, 1.2]	[0.0258, 0603]	[0.0258, 0603]	[0.013, 1.2]
Heat Capacity	[1.02, 3223]	[1.02, 3223]	[0.5, 6446]	[1.02, 3223]	[1.02, 3223]	[0.5, 6446]

Table 5: The geometry parameter ranges for the generation of the different SURF datasets.

D.1 Training and Loss

Both models are trained with seven-step supervision. The predictions are calculated as follows:

$$\begin{aligned} \mathbf{q}_P^{t+1} &= \mathbf{q}_{GT}^{t} + \mathbf{p}^1 \\ \mathbf{q}_P^{t+2} &= \mathbf{q}_P^{t+1} + \mathbf{p}^2 \\ &\vdots \\ \mathbf{q}_P^{t+7} &= \mathbf{q}_P^{t+6} + \mathbf{p}^6 \end{aligned} \tag{3}$$

The loss is calculated as:

$$loss = \text{MSE}\left(target, output\right) \tag{4}$$

	Base	**Rotated**	**Range**	**Topo**	**Dynamic**	**Full**	**Full Finer**
Number of Datapoints	1200	1200	1200	1315	1200	1241	1241
Time Steps	300	300	300	300	300	300	300
End Time [s]	3	3	3	3	3	3	3
Average Number of Nodes	1287	1288	1293	1751	1287	1787	5433
Dataset Size [GB]	10.9	10.8	10.9	16.0	11.3	16.7	68.3

Table 6: Overview of the different SURF datasets.

Table 7: Statistics for the state variables of the SURF-Full data set.

	Mean	**Standard Deviation**
Velocity	-1	9
Pressure	59	137
Temperature	340	129

with:

$$target = \begin{bmatrix} \mathbf{q}_{GT}^{t+1} - \mathbf{q}_{GT}^{t} \\ \mathbf{q}_{GT}^{t+2} - \mathbf{q}_{P}^{t+1} \\ \vdots \\ \mathbf{q}_{GT}^{t+7} - \mathbf{q}_{P}^{t+6} \end{bmatrix}, output = \begin{bmatrix} \mathbf{p}^{1} \\ \vdots \\ \mathbf{p}^{7} \end{bmatrix} \tag{5}$$

$\mathbf{q}^t$ denotes the state at time step t the subscripts GT and P are used to specify whether the state values are from the ground truth or predicted values, respectively. Note that the target values are always calculated with respect to the previously predicted state and not the state from the ground truth.

The authors of EAGLE show that both models are more robust if trained on more than one time step. Hence, for better comparability, we train both models with seven-step supervision. We use the implementation provided by EAGLE [40] for both models. In this implementation, the loss for the MGN model is calculated differently. In contrast to the loss for the EAGLE model, the $target_{MGN}$ is calculated solely on the ground truth data and is independent of the predicted data. We have found that using the loss given in Eq. 4 gives better results for the MGN model and therefore use the same loss for both models. This also lets us focus more on differences in the architectures.

The learning rate of 10^{-4} is used in the EAGLE paper [8]. As shown in Fig. 7 for the SURF-Full data set, the training works better with a learning rate of 10^{-5}. We therefore use a learning rate of 10^{-5} in our experiments. For MGN we use a learning rate of 10^{-4} with a decay factor of 0.999.

D.2 Heat Transfer

In contrast to the data set used in the MGN and EAGLE paper, the SURF datasets include heat transfer and therefore the temperature as a state variable. Hence, the node encoders and decoders are adapted to include the temperature in the state as input value for the node features and output value, respectively. Apart from the flow velocity, two fluid properties characterize heat transfer: heat capacity and thermal conductivity. In Table 5, the range of the two properties used to calculate the data sets is reported. These two properties are fed to the Neural Networks as additional node features to each node. Hence, the node features consist of a concatenation of the following: state (velocity, pressure, temperature), one hot encoding of the node type, and the thermal fluid properties.

D.3 Computational Resources

Trainings were executed on Nvidia Geforce RTX 3090 GPU's with a memory of 24 GB. The MGN model required 24 GB of memory for the SURF-Full data set, where the number of nodes is the highest, while the EAGLE model used 12 GB GPU memory. The 1,000 epochs of the EAGLE model

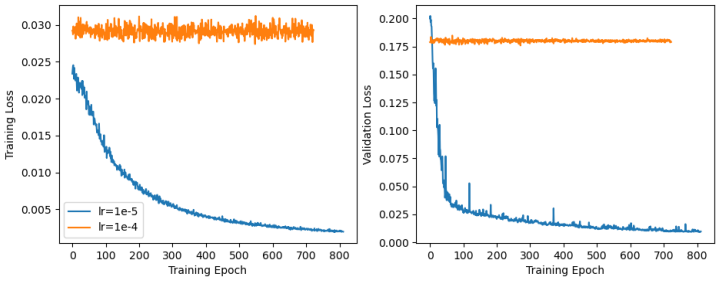

Figure 7: Training and validation loss for the EAGLE model with different training parameters for the SURF-Full data set.

Table 8: Performance scores for velocity, pressure, and temperature after 250 times steps of MGN and EAGLE trained and tested on all datasets.

Dataset	Model	$PS_v \downarrow$	$PS_p \downarrow$	$PS_t \downarrow$	$PS \downarrow$
Base	MGN	0.073 ± 0.005	0.142 ± 0.011	0.068 ± 0.003	0.094 ± 0.006
	EAGLE	0.074 ± 0.003	0.090 ± 0.004	0.097 ± 0.006	0.087 ± 0.004
Rotated	MGN	0.041 ± 0.002	0.133 ± 0.003	0.071 ± 0.002	0.082 ± 0.001
	EAGLE	0.061 ± 0.007	0.089 ± 0.005	0.107 ± 0.003	0.086 ± 0.004
Topology	MGN	0.094 ± 0.019	0.155 ± 0.037	0.087 ± 0.008	0.112 ± 0.021
	EAGLE	0.087 ± 0.006	0.092 ± 0.003	0.110 ± 0.007	0.097 ± 0.002
Range	MGN	0.088 ± 0.004	0.130 ± 0.005	0.070 ± 0.010	0.096 ± 0.005
	EAGLE	0.111 ± 0.012	0.121 ± 0.015	0.084 ± 0.004	0.106 ± 0.008
Dynamic	MGN	0.045 ± 0.008	0.050 ± 0.004	0.035 ± 0.002	0.043 ± 0.004
	EAGLE	0.131 ± 0.020	0.123 ± 0.008	0.109 ± 0.008	0.121 ± 0.011
Full	MGN	0.165 ± 0.099	0.168 ± 0.088	0.111 ± 0.041	0.148 ± 0.076
	EAGLE	0.163 ± 0.004	0.166 ± 0.011	0.146 ± 0.015	0.158 ± 0.006
Mesh	MGN	0.292 ± 0.013	0.294 ± 0.028	0.211 ± 0.012	0.266 ± 0.017
	EAGLE	0.237 ± 0.010	0.248 ± 0.017	0.237 ± 0.016	0.240 ± 0.007

took around 32h. The training for the MGN models was aborted after 48h reaching 700 - 800 epochs. The training for SURF-Mesh required more GPU memory and was done on A6000s with 48 GB of GPU memory for the EAGLE baseline and on A100 80GB GPU memory for the MGN baseline respectively.

E Full Empirical Evaluation

We report the full performance and generalization scores in Table 8 and 9:

F Examples

Figures 8 to 14 depict examples for SURF datapoints.

Table 9: Generalization scores for 250 time steps the MGN and EAGLE model trained on all training sets and then tested on the test set SURF-Base, SURF-Rotated, SURF-Topology, SURF-Range, SURF-Dynamic, SURF-Full, and SURF-Mesh. Note that this is different from the SURF generalization scores.

Training Set	Model	Base ↓	Rotated ↓	Topology ↓	Range ↓	Dynamic ↓	Full ↓	Mesh ↓
Base	MGN	1.00 ± 0.00	9.16 ± 0.54	3.10 ± 0.72	1.30 ± 0.07	5.76 ± 0.64	3.32 ± 1.30	1.55 ± 0.22
	EAGLE	1.00 ± 0.00	6.67 ± 1.44	3.68 ± 0.10	1.08 ± 0.09	2.17 ± 0.08	3.73 ± 0.52	2.33 ± 0.14
Rotated	MGN	1.08 ± 0.07	1.00 ± 0.00	1.81 ± 0.38	1.54 ± 0.18	6.73 ± 0.60	1.95 ± 0.72	1.09 ± 0.11
	EAGLE	1.63 ± 0.13	1.00 ± 0.00	1.96 ± 0.08	1.36 ± 0.22	2.41 ± 0.17	1.47 ± 0.06	1.04 ± 0.05
Topology	MGN	1.52 ± 0.34	7.81 ± 0.54	1.00 ± 0.00	1.48 ± 0.26	7.86 ± 0.92	2.96 ± 1.24	1.50 ± 0.04
	EAGLE	2.49 ± 0.11	4.81 ± 0.28	1.00 ± 0.00	1.89 ± 0.10	3.27 ± 0.26	3.07 ± 0.26	2.22 ± 0.30
Range	MGN	1.94 ± 0.27	11.95 ± 1.63	3.79 ± 0.96	1.00 ± 0.00	7.63 ± 0.75	4.21 ± 1.84	2.36 ± 0.73
	EAGLE	1.89 ± 0.06	8.24 ± 2.08	3.60 ± 0.03	1.00 ± 0.00	2.33 ± 0.20	4.81 ± 2.20	2.81 ± 1.11
Dynamic	MGN	0.58 ± 0.03	9.09 ± 0.35	3.11 ± 0.56	1.52 ± 0.23	1.00 ± 0.00	3.93 ± 1.52	1.94 ± 0.16
	EAGLE	2.14 ± 0.19	10.96 ± 2.49	4.47 ± 0.24	2.16 ± 0.33	1.00 ± 0.00	5.25 ± 1.76	3.99 ± 1.58
Full	MGN	3.53 ± 2.10	3.51 ± 2.26	1.98 ± 0.76	2.41 ± 1.42	6.30 ± 3.08	1.00 ± 0.00	1.07 ± 0.26
	EAGLE Full	5.22 ± 0.49	2.58 ± 0.13	3.50 ± 0.45	4.04 ± 0.94	3.96 ± 0.38	1.00 ± 0.00	1.01 ± 0.05
Mesh	MGN	4.63 ± 0.42	4.50 ± 0.71	3.07 ± 0.97	2.91 ± 0.33	9.03 ± 1.41	1.70 ± 0.71	1.00 ± 0.00
	EAGLE	5.07 ± 0.38	3.63 ± 0.23	3.97 ± 0.16	4.04 ± 0.88	3.80 ± 0.37	1.71 ± 0.10	1.00 ± 0.00

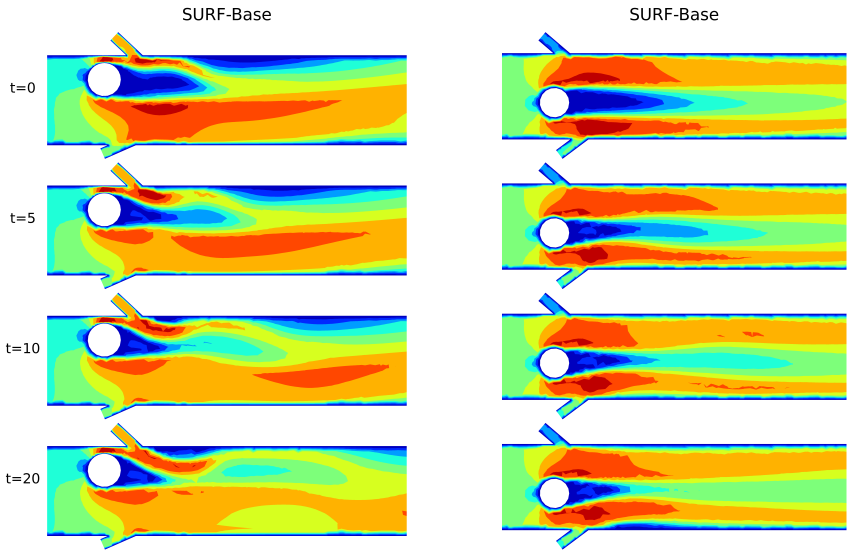

Figure 8: Velocity visualization of two ground-truth datapoints from SURF-Base, evaluated at timesteps 0, 5, 10, 20 (top to bottom).

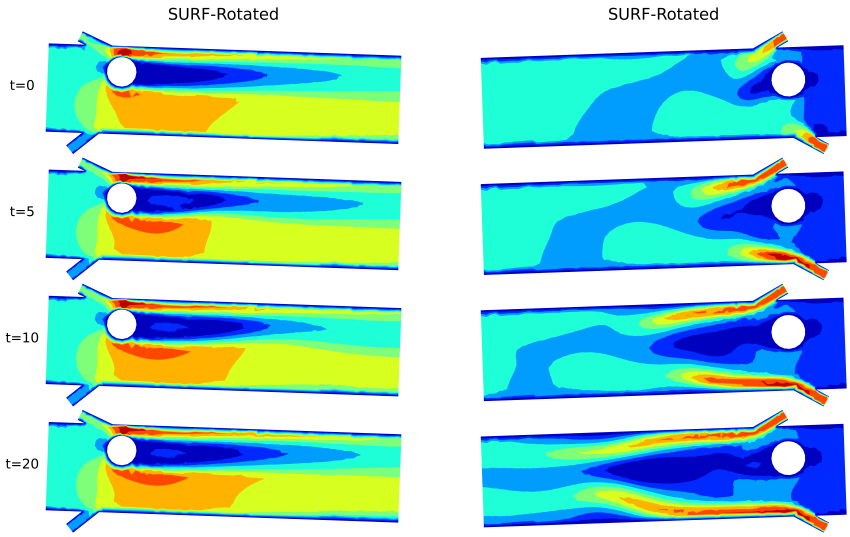

Figure 9: Velocity visualization of two ground-truth datapoints from SURF-Rotated, evaluated at timesteps 0, 5, 10, 20 (top to bottom).

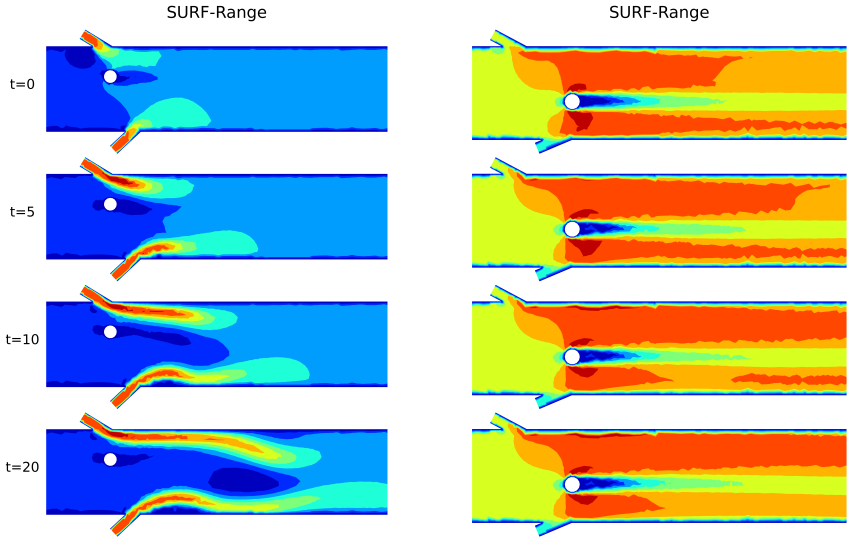

Figure 10: Velocity visualization of two ground-truth datapoints from SURF-Range, evaluated at timesteps 0, 5, 10, 20 (top to bottom).

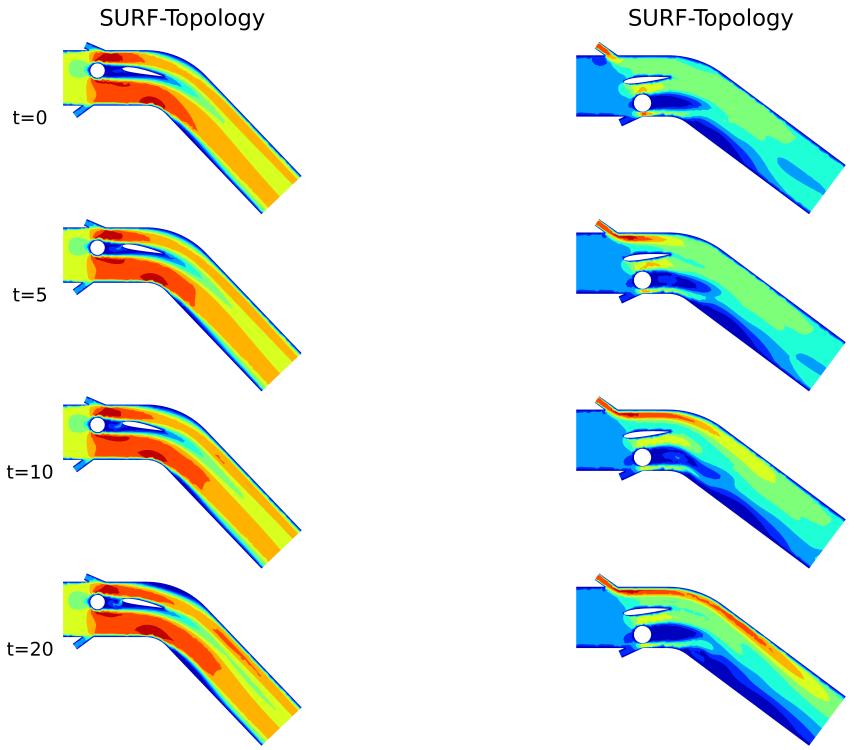

Figure 11: Velocity visualization of two ground-truth datapoints from SURF-Topology, evaluated at timesteps 0, 5, 10, 20 (top to bottom).

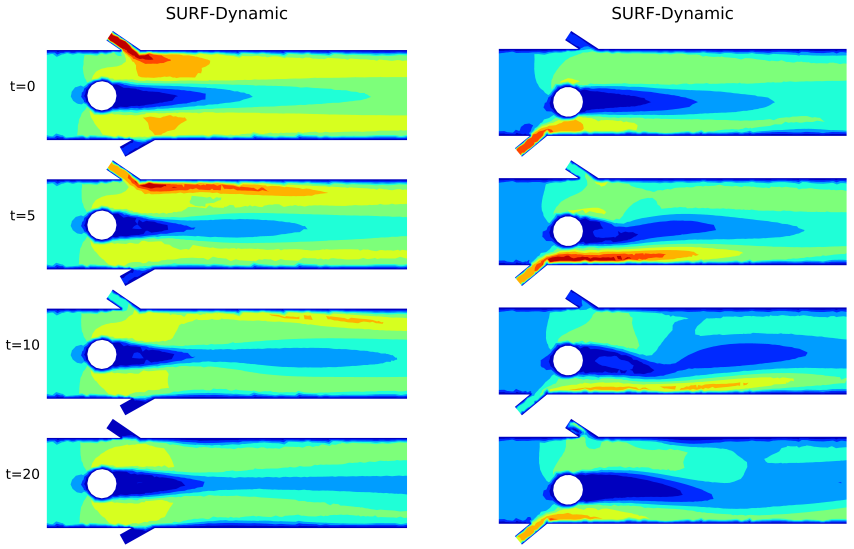

Figure 12: Velocity visualization of two ground-truth datapoints from SURF-Dynamic, evaluated at timesteps 0, 5, 10, and 20 (top to bottom).

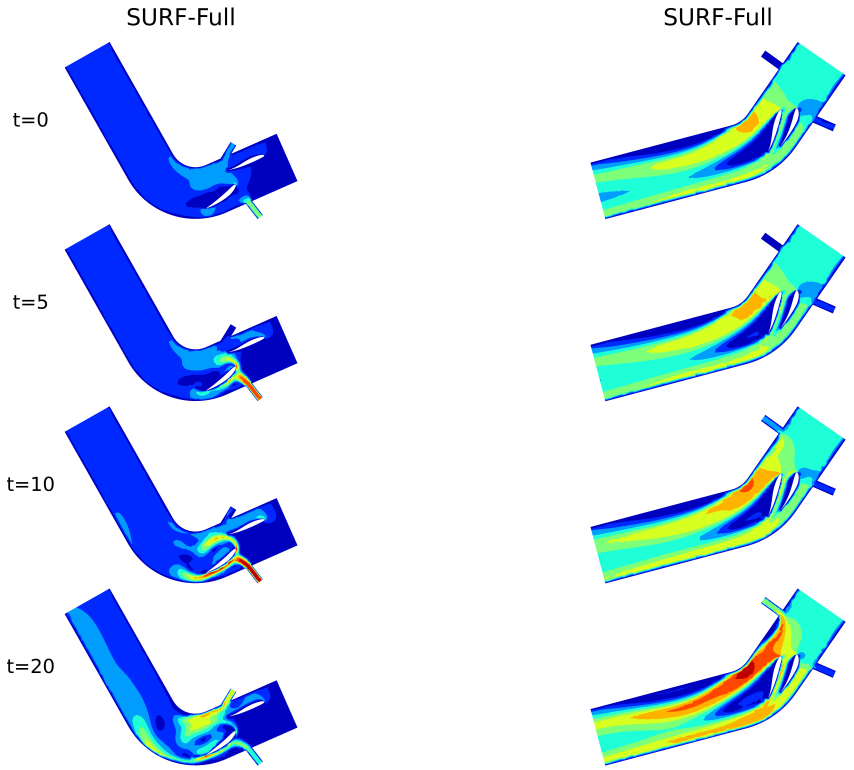

Figure 13: Velocity visualization of two ground-truth datapoints from SURF-Full, evaluated at timesteps 0, 5, 10, 20 (top to bottom).

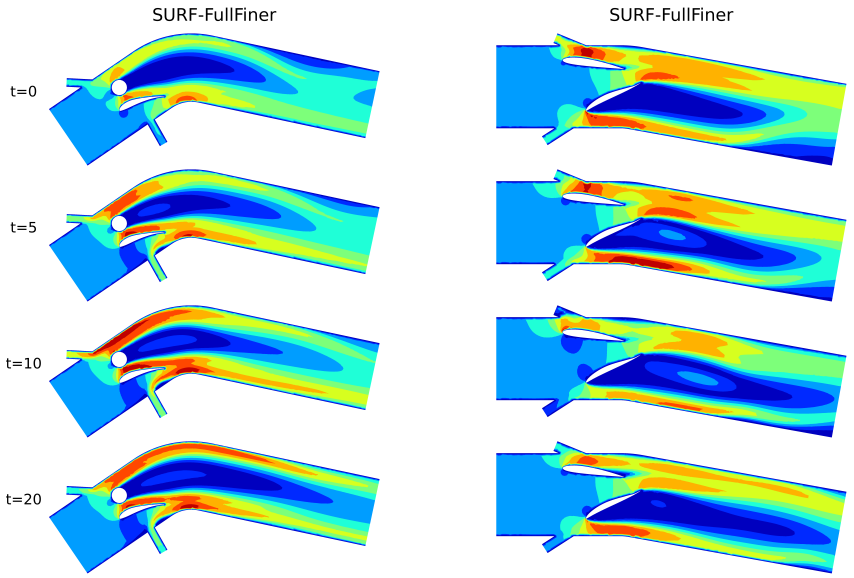

Figure 14: Velocity visualization of two ground-truth datapoints from SURF-Mesh, evaluated at timesteps 0, 5, 10, 20 (top to bottom).