

# DIRECT PREDICTION OF STEADY-STATE FLOW FIELDS IN MESHED DOMAIN WITH GRAPH NETWORKS

**Lukas Harsch, Stefan Riedelbauch**

Institute of Fluid Mechanics and Hydraulic Machinery  
University of Stuttgart

Stuttgart, Germany 70569

{lukas.harsch,stefan.riedelbauch}@ihs.uni-stuttgart.de

## ABSTRACT

We propose a model to directly predict the steady-state flow field for a given geometry setup. The setup is an Eulerian representation of the fluid flow as a meshed domain. We introduce a graph network architecture to process the mesh-space simulation as a graph. The benefit of our model is a strong understanding of the global physical system, while being able to explore the local structure. This is essential to perform direct prediction and is thus superior to other existing methods.

## 1 INTRODUCTION

Using deep learning to model physical problems has become an area of great research interest. Learning the physical equations of a systems can be used as model to control robots (Hoppe et al., 2019) or to solve numerical problems (Raissi, 2018; Sirignano & Spiliopoulos, 2018). In case of computational fluid dynamics (CFD), learned simulations can reduce the computational cost and thus improve the engineering process of, in our case, hydraulic components. During the development process of hydraulic machinery, a large amount of designs has to be simulated and evaluated with respect to their fluid dynamical properties. To speed up this process, stationary simulations are often preferred in the early stage of the design process. For this reason, we propose a method, which enables fast predictions of the fluid dynamics, focusing on predicting the steady-state flow field.

In recent years, deep learning approaches (Tompson et al., 2017; Xie et al., 2018) have shown promising results inferring fluid flow simulations. Data-driven approaches are used to learn the underlying physical behavior of the fluid dynamics from CFD simulations. A generalized solution of the fluid dynamics should be learned to infer the flow field for new problem setups.

A major difference of the algorithms is the way the fluid domain is represented. Modeling complex physical systems in general requires a discretized simulation domain. Mesh-based representations are a common choice to solve the underlying partial differential equations (PDEs) of the system, e.g. the Navier-Stokes equations for CFD. The mesh can be structured as a Cartesian grid to store the flow field. So image processing algorithms can be applied (Guo et al., 2016; Thuerey et al., 2018), which has its limitations for complex structured geometries. In common numerical solvers the flow field is represented as a mesh of arbitrary polygons, which is widely used in mechanical engineering. With the success of graph neural networks (Battaglia et al., 2018), it is possible to use deep learning for modeling such physical problems. Further, the fluid can be treated as particle-based representation, which is common used in smoothed particle hydrodynamics (SPH) and position based fluids (PBF) for simulating free-surface liquids. Particles can be processed as a point cloud. In Schenck & Fox (2018), Ummenhofer et al. (2020) and Sanchez-Gonzalez et al. (2020), this representation is used to model the highly dynamic movement of fluids and the interaction with rigid objects.

In our work, we focus on mesh-based representation on an irregular grid. The combination of a differentiable solver with graph convolutions (GCN) (Kipf & Welling, 2017) is used in De Avila Belbute-Peres et al. (2020) for high-resolution predictions on the meshed domain around airfoils. Our method comes without an additional solver to be more independent from domain knowledge which is necessary for the solver. Pfaff et al. (2021) provides MESHGRAPHNETS, which is especially designed for predicting the dynamical states of the system over time. In section 4 we show, that our method is better suited for steady-state predictions.

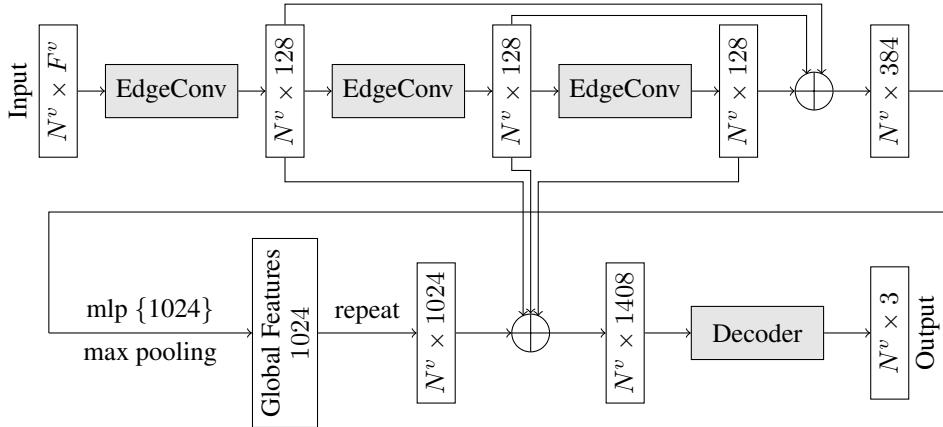


Figure 1: The model architecture to predict the flow field, given the node features of shape  $N^v \times F^v$ . **Top row:** EDGECONV layers as local feature extractors. **Top row:** Pooling, to form an 1D global descriptor. Concatenation of the local and global descriptor and transformation by the decoder.  $\oplus$ : concatenation.

## 2 MODEL

The task is a steady-state prediction, so the model should learn to directly predict the final steady-state velocity and pressure fields, given only information about structure of the meshed domain and additional domain information, i.e. node positions  $\mathbf{u}_i$  and additional quantities  $\mathbf{n}_i$ , respectively. We propose a graph neural network model related to Dynamic Graph CNN (DGCNN) (Wang et al., 2019). Figure 1 shows the visual scheme of our model architecture.

### 2.1 EDGE FUNCTION

The meshed domain can be seen as a bidirectional graph  $G = (V, E)$ , with nodes  $V$  connected by mesh edges  $E$ . Each node  $i \in V$  contains the mesh-space coordinate  $\mathbf{u}_i$ , as well as quantities  $\mathbf{n}_i$  describing the domain, e.g. angle of attack, Mach number or the node type to distinguish between the fluid domain and solid objects like the geometry. In addition, each node is carrying the fluid dynamical quantities  $\mathbf{q}_i$ , which we want to model. The node feature vector  $\mathbf{v}_i = [\mathbf{u}_i, \mathbf{n}_i]$  is defined as the concatenation of the nodes position  $\mathbf{u}_i$  and the quantities  $\mathbf{n}_i$ . The total amount of nodes is defined by  $N^v$  with a node feature vector size of  $F^v$ . The edge feature vectors  $\mathbf{e}_{ij} \in E$  are defined as  $\mathbf{e}_{ij} = h_\Theta(\mathbf{v}_i, \mathbf{v}_j)$ , where  $h_\Theta$  is a nonlinear function with a set of learnable parameters  $\Theta$ .

Extraction both local and global features is essential for the task of directly predicting a global field, since it is necessary to understand the local properties of the mesh as well as its global structure. Therefore, we are using an edge function called EDGECONV introduced by DGCNN, which is defined as  $h_\Theta(\mathbf{v}_i, \mathbf{v}_j) = \bar{h}_\Theta(\mathbf{v}_i, \mathbf{v}_j - \mathbf{v}_i)$ . It is implemented as

$$\mathbf{e}'_{ij} = \text{ReLU}(\Theta \cdot (\mathbf{v}_j - \mathbf{v}_i) + \phi \cdot \mathbf{v}_i), \quad \mathbf{v}'_i = \max_{j:(i,j) \in E} \mathbf{e}'_{ij}, \quad (1)$$

with  $\Theta$  and  $\phi$  implemented as a shared MLP and max as aggregation operation.

The local neighborhood information are captured by  $(\mathbf{v}_j - \mathbf{v}_i)$ , which includes the relative displacement vector in mesh space  $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$  and the difference of the local quantities  $\mathbf{n}_i$ . The global shape structures are captured by the patch center  $\mathbf{v}_i$  including the absolute node position  $\mathbf{u}_i$  and its quantities  $\mathbf{n}_i$ . The MLP is embedding the feature vector  $\mathbf{v}_i$  to a latent vector of size 128. The EDGECONV formulation has properties lying between translation-invariance and non-locality. More details about this model type can be found in Wang et al. (2019).

### 2.2 ARCHITECTURE

The DGCNN architecture was originally designed for point clouds, using a k-nearest neighbor algorithm to construct the graph. Since the simulation domain is already present as a mesh we rely

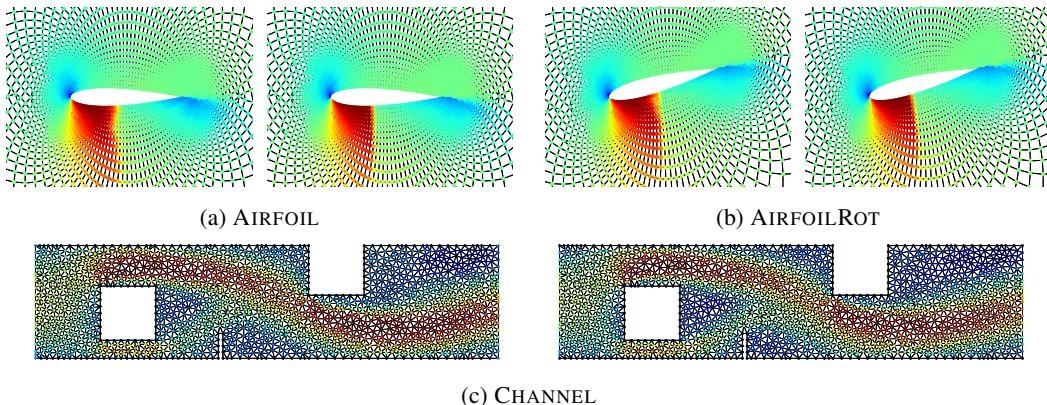


Figure 2: Examples of the predicted velocity field in  $x$ -direction for all three data sets, each with the ground truth simulation to the right. The simulation mesh is displayed in the background. For CHANNEL, the geometry and walls are highlighted in black.

on the graph structure of the mesh instead of recomputing the graph. We use the structure of the DGCNN architecture to increase the ability of extracting local and global properties as displayed in Figure 1. The model has  $L$  layers of EDGECONV. The outputs of the EDGECONV layers can be seen as local feature descriptors. Shortcut connections are included to extract multi-scale features. The features from all EDGECONV layers are concatenated and aggregated by a fully-connected layer. Then, a global max pooling is used to get the global feature vector of the mesh. Afterwards, the global feature vector is repeated and concatenated with the local features. At last, the decoder MLP is transforming the latent features to the desired output  $\mathbf{p}_i$ . Further information about the network and hyperparameter settings can be found in Section A.1.

### 3 EXPERIMENTAL SETUP

For our experiments, three data sets with fluid dynamical simulations on two-dimensional meshes are used. The flow field is represented by the simulation mesh. Each node is holding the fluid dynamical quantities  $\mathbf{q}_i = [\mathbf{w}_i, p_i]$ , with the velocity in  $x$ - and  $y$ -direction  $\mathbf{w}_i$  and the pressure  $p_i$ . All samples in the data set AIRFOIL are steady-state, compressible, inviscid CFD simulations, performed by solving the Euler equations. The same simulation conditions apply to data set AIRFOILROT, which is a modified version of AIRFOIL. The CFD simulations of the data set CHANNEL are performed by solving the steady-state, incompressible, inviscid case of the Navier-Stokes equations.

The first data set AIRFOIL is taken from Belbute-Peres et al. (2020). The simulations are describing the flow around the cross-section of a NACA0012 airfoil for varying angle of attacks  $\alpha$  and different Mach numbers  $m$ . It includes a single geometry setup, so the mesh is identical for every sample in the data set. Thus, the simulated quantities  $\mathbf{q}_i$  vary only due to the parameter setting of  $\alpha$  and  $m$ . The node feature vectors  $\mathbf{v}_i$  include the node positions  $\mathbf{u}_i$ , the global  $m$  and  $\alpha$  and the node type, indicating, whether the node belongs to the airfoil geometry, the fluid domain or the boundary of the flow field. Since the node positions are the same for every simulation, the model predictions mainly depend on the parameter setup of  $\alpha$  and  $m$ . Consequently, it is difficult to show the advantage of our method to capture local and global structures. This data set is used as a baseline to compare our method with the results from literature.

A second data AIRFOILROT set is constructed, by rotating the whole simulation domain of AIRFOIL by the value of  $\alpha$ . This can be interpreted as  $\alpha = 0$ , but with a rotated placement of the simulation mesh and its flow field. Here,  $\alpha$  will be discarded from the feature vector, since it is equal for all samples. Now, the model has to rely more on the geometric structure for predicting the flow field. All samples in the data set still have the same geometry profile. Nevertheless, the orientation of the airfoil results in strong variations of the flow field, so the model has to be sensitive in understanding the geometric structure.

Table 1: Test root mean square error (RMSE) for all tasks (RMSE  $\times 10^{-2}$ ).

Method	AIRFOIL	AIRFOILROT	CHANNEL
GCN	1.40	-	-
MESHGRAPHNETS	0.95	7.34	13.30
MESHGRAPHNETS (ABS)	<b>0.73</b>	1.28	13.04
MESHGRAPHNETS (ABS+POOL)	1.25	0.99	6.58
Ours	1.40	<b>0.87</b>	<b>5.81</b>

To increase the requirement of understanding the geometric structure, we generate a third data set CHANNEL, which is a two-dimensional channel flow. Generic objects like circles, squares, triangles and rectangle are randomly placed inside the channel. The number of objects varies between one and three, which results in a large variety of fluid dynamics i.e., various flows, back flows, partial blockage and small gaps. The feature vector for CHANNEL only consists of the node positions  $\mathbf{u}_i$  and the node type, i.e. fluid domain, wall or internal object. All samples in CHANNEL are simulated with the same boundary conditions, so there is no additional information encoded in the feature vector. Thus, the focus lies on the investigation for the understanding of the geometric structure. Examples for all three data sets are shown in Figure 2. More details about the data sets are explained in Section A.2.

## 4 RESULTS

We compare our approach with two baseline methods, a base GCN and MESHGRAPHNETS (Pfaff et al., 2021). The results for the base GCN are taken from De Avila Belbute-Peres et al. (2020). They are only available for the AIRFOIL data set. Then, we use a re-implementation of MESHGRAPHNETS for comparison with the data sets studied in this work.

MESHGRAPHNETS was designed for modeling the system dynamics over time. The model's spatial equivariance bias is counteracting the direct prediction of a final flow field. However, the model can achieve good results for AIRFOIL, but fails to predict the flow field for the other data sets. We investigate two extensions of MESHGRAPHNETS, indicated by the suffix (ABS) and (ABS+POOL). First, the absolute positions  $\mathbf{u}_i$  is concatenated to the node features (ABS). Second, we include the pooling operation for global feature extraction (ABS+POOL). Both extensions are adopted from the DGCNN architecture and aim to increase the understanding of the global structure. The adapted model is described more detailed in Section A.3.

The visual results of our method are displayed in Figure 2, showing the predicted and the ground truth field of the velocity component in  $x$ -direction. There is almost no visually noticeable difference between the predicted and the ground truth flow field. This shows, that our model is able to produce high-quality predictions. More visual results of the velocity fields are shown in Figure 4.

Table 1 shows the prediction error in terms of RMSE for all five models on the three data sets. In the simplest data set AIRFOIL, with respect to the complexity of the geometric structure, all models successfully learn to predict the flow field. As mentioned in Pfaff et al. (2021) MESHGRAPHNETS outperform the base GCN, which is also the case with our re-implementation. Surprisingly, (ABS) outperforms all the other methods. In terms of RMSE, our method falls behind, but visually the fluid dynamics remain nearly identical to the ground truth.

On the richer AIRFOILROT task, MESHGRAPHNETS was unable to obtain a correct prediction of the flow field. This results in a high RMSE, which is also confirmed by visual investigations. In contrast, the adapted MESHGRAPHNETS capture enough geometric information to successfully predict the flow field. This experiment shows the benefit of using both the local displacement and the absolute node position as features. In this case, our model outperforms the other methods in terms of RMSE.

Beside the velocities and pressure, the flow field of a compressible flow should also be represented by the density, but AIRFOIL and AIRFOILROT do not provide values for the density. Nevertheless, including the density as an additional output quantity will increase the complexity of the problem setup and thus, increase the prediction error slightly. Further, it has to be mentioned, that the test

set of AIRFOIL and AIRFOILROT is within the Mach number range of the training set. So the test case is an interpolation task. Since we are not using any additional physical equations or numerical solvers, our model fails to solve difficult extrapolation tasks as presented in De Avila Belbute-Peres et al. (2020).

For the most challenging data set CHANNEL, also (ABS) fails. Only (ABS+POOL) and our method achieve a sufficient low RMSE, but with an improvement of the RMSE with our method. The same findings are indicated by visual investigations of the predicted flow fields. This experiment shows, that our method generalizes well to new unseen geometric setups of the channel flow. In this experiment, both models with a global feature extractor are successful. This shows, that extracting the global features with the pooling operation is necessary to capture a high level of understanding of the global geometric structure. Nevertheless, our method is better in terms of model complexity and processing time, which makes it superior to MESHGRAPHNETS (ABS+POOL).

## 5 CONCLUSION

We propose a mesh-based method to model the fluid dynamics for an accurate and efficient prediction of steady-state flow fields. The experiments demonstrate the models strong understanding of the geometric structure. The method is independent of the structure of the simulation domain, being able to capture highly irregular meshes. Further, our method works well for different kind of systems, i.e. Eulerian and Navier-Stokes systems. Beside this, we show, that the model does not require any a-priori domain information, e.g. inflow velocity or material parameters. Thus, the model can be used for other systems, e.g. viscous fluids or other domains which are represented as field data like rigid body simulations.

## REFERENCES

- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.
- Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018. URL <https://arxiv.org/pdf/1806.01261.pdf>.
- Filipe de Avila Belbute-Peres, Thomas D. Economon, and J. Zico Kolter. Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction. In *International Conference on Machine Learning (ICML)*, 2020.
- Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2402–2411. PMLR, 13–18 Jul 2020. URL <http://proceedings.mlr.press/v119/de-avila-belbute-peres20a.html>.
- 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, pp. 481–490, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939738.
- Sabrina Hoppe, Zhongyu Lou, Daniel Hennes, and Marc Toussaint. Planning Approximate Exploration Trajectories for Model-Free Reinforcement Learning in Contact-Rich Manipulation. *IEEE Robotics and Automation Letters*, PP:1–1, 2019. doi: 10.1109/LRA.2019.2928212.
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning*

- Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL <http://proceedings.mlr.press/v37/ioffe15.html>.
- Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL [https://openreview.net/forum?id=roNqYL0\\_XP](https://openreview.net/forum?id=roNqYL0_XP).
- Maziar Raissi. Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations. *J. Mach. Learn. Res.*, 19:25:1–25:24, 2018.
- Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, 2020.
- Connor Schenck and Dieter Fox. SPNets: Differentiable Fluid Dynamics for Deep Neural Networks. *CoRR*, abs/1806.06094, 2018.
- Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. ISSN 0021-9991. doi: <https://doi.org/10.1016/j.jcp.2018.08.029>.
- 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.
- Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating Eulerian Fluid Simulation with Convolutional Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML’17, pp. 3424–3433. JMLR.org, 2017.
- Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian Fluid Simulation with Continuous Convolutions. In *International Conference on Learning Representations*, 2020.
- Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), October 2019. ISSN 0730-0301. doi: 10.1145/3326362. URL <https://doi.org/10.1145/3326362>.
- You Xie, Erik Franz, Mengyu Chu, and Nils Thürey. tempoGAN: a temporally coherent, volumetric GAN for super-resolution fluid flow. *ArXiv*, abs/1801.09710, 2018.

## A APPENDIX

### A.1 ADDITIONAL MODEL DETAILS

All MLPs in the EDGECONV layers are using three shared fully-connected layers of size 128. The two hidden layers come with ReLU activation, followed by a non-activated output layer. The MLPs in the EDGECONV layers are followed by a BatchNorm (Ioffe & Szegedy, 2015) layer. To aggregate the features of all EDGECONV layers, a fully-connected layer of size 1024 is used. The decoder MLP has four fully-connected layers of size (1024, 512, 256, 3) to transform the latent feature vector to the output shape. We are using  $L = 5$  EDGECONV layers, which is a good trade-off between model capacity and computational cost. The whole model consists of 20 fully-connected layers with a total number of approximately 3.3 million parameters.

We train our model in a supervised fashion using the per-node output features  $\mathbf{p}_i$  from the decoder with a  $L_2$  loss between  $\mathbf{p}_i$  and the corresponding ground truth values  $\mathbf{p}_i$ . A batch size of 2 is chosen for AIRFOIL and AIRFOILROT. In CHANNEL the number of nodes is smaller compared to the former data sets, so a batch size of 6 is used. The models are trained on a single RTX 3090 GPU with the Adam optimizer, using a learning rate of  $10^{-4}$ .

## A.2 DATASET

The simulation domain of AIRFOIL and AIRFOILROT is a two-dimensional, mixed triangular and quadrilateral mesh with 6648 nodes. The mesh is highly irregular with varying scale of the edge lengths in different regions of the mesh. The training set is defined by

$$\alpha_{\text{train}} = \{-10, -9, \dots, 9, 10\}, \quad m_{\text{train}} = \{0.2, 0.3, 0.35, 0.4, 0.5, 0.55, 0.6, 0.7\}.$$

The test set is defined similar by

$$\alpha_{\text{test}} = \{-10, -9, \dots, 9, 10\}, \quad m_{\text{test}} = \{0.25, 0.45, 0.65\}.$$

The training and test pairs are sampled uniformly from  $\alpha \times m$ . This results in 168 samples for the training set and 63 samples for the test set, which is split to 32 samples for validation and 31 samples for the test. The samples from AIRFOIL and AIRFOILROT come with a min-max normalization to the value range of  $[-1, 1]$  separately for each velocity component and the pressure.

The simulation domain of CHANNEL is a two-dimensional, unstructured triangular mesh with an equal scale of the edge lengths. Depending on shape and number of objects inside the channel, the number of nodes is variable, but is approximately 1750. This data set has 1000 samples for training and 150 samples each for validation and test. In CHANNEL, we normalize the samples by the maximum velocity and the maximum pressure.

The node features used as inputs as well as the outputs are summarized in the following table.

Method	inputs $\mathbf{v}_i$	outputs $\mathbf{p}_i$
AIRFOIL	$\mathbf{u}_i, n_i, m, \alpha$	$\mathbf{w}_i, p_i$
AIRFOILROT	$\mathbf{u}_i, n_i, m$	$\mathbf{w}_i, p_i$
CHANNEL	$\mathbf{u}_i, n_i$	$\mathbf{w}_i, p_i$

The inputs are defined by the node position  $\mathbf{u}_i$ , the node type  $n_i$ , the angle of attacks  $\alpha$  and the Mach numbers  $m$ , while the outputs are the velocity in  $x$ - and  $y$ -direction  $\mathbf{w}_i$  and the pressure  $p_i$ .

## A.3 MESHGRAPHNETS ADAPTION

For the adaption of MESHGRAPHNETS we are following their Encoder-Processor-Decoder architecture, but with a processor adapted by the idea of DGCNN. Instead of EDGECONV layers, message passing blocks (Battaglia et al., 2018) are used to update node and edge features. Figure 3 shows the visual scheme of the adapted architecture.

**Encoder** The edge feature vectors  $\mathbf{e}_{ij} \in E$  are a concatenation of the relative displacement vector in mesh space  $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$  and its norm  $|\mathbf{u}_{ij}|$  with an edge feature vector size of  $F^e$  and a total amount of edges  $N^e$ . The node feature vectors  $\mathbf{v}_i \in V$  is defined by the quantities  $\mathbf{n}_i$  and can be extended by other features with a node feature vector size of  $F^v$  a total number of nodes  $N^v$ . In the encoder, MLPs  $\epsilon^M$  and  $\epsilon^V$  are used to embed the feature vectors  $\mathbf{e}_{ij}$  and  $\mathbf{v}_i$ , respectively to latent vectors of the size 128.

**Processor** The processor uses  $L$  identical message passing blocks to update the edges  $\mathbf{e}$  and nodes  $\mathbf{v}$  into embeddings  $\mathbf{e}'$  and  $\mathbf{v}'$  respectively by

$$\mathbf{e}'_{ij} \leftarrow f^M(\mathbf{e}_{ij}, \mathbf{v}_i, \mathbf{v}_j), \quad \mathbf{v}'_i \leftarrow f^V(\mathbf{v}_i, \sum_j \mathbf{e}'_{ij}), \quad (2)$$

where  $f^M$  and  $f^V$  are implemented as MLPs with residual connections. The MLPs are using the concatenated node and edge feature vectors as input.

**Decoder** Finally, the decoder MLP is transforming the latent node features from the last processor block into the output features. More details about this model type can be found in Sanchez-Gonzalez et al. (2020) and Pfaff et al. (2021).

**Local and global Features** EDGECONV captures both the local neighborhood information ( $\mathbf{v}_j - \mathbf{v}_i$ ) and the global shape structures by  $\mathbf{v}_i$ . Therefore, we are adapting the node features by using a concatenation of the quantities  $\mathbf{n}_i$  and the absolute node position  $\mathbf{u}_i$ . This node features help to capture

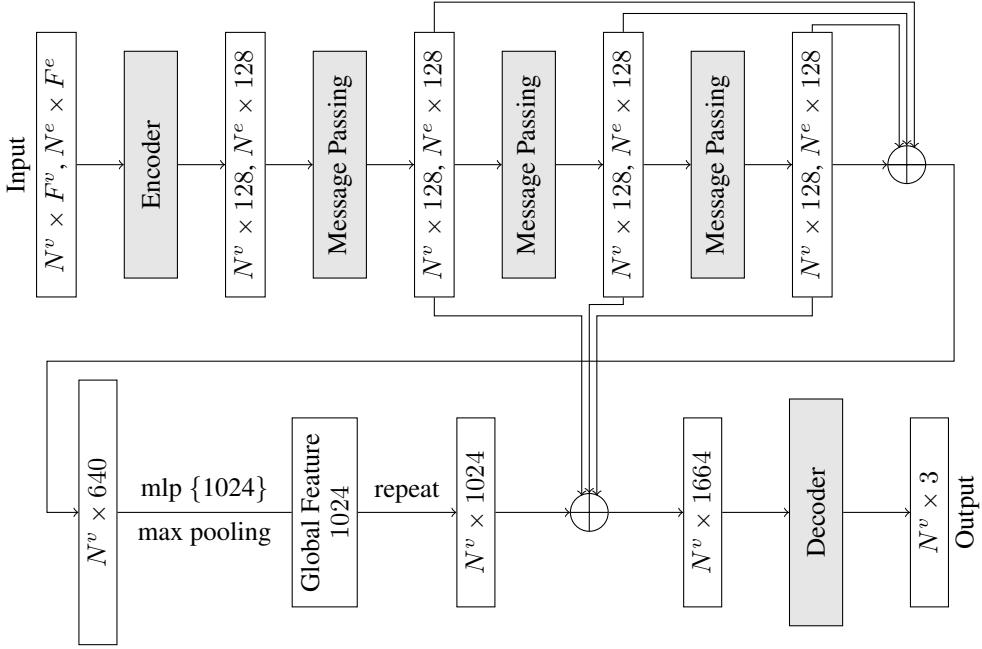


Figure 3: Adapted model architecture using message passing block with local and global descriptor.  
 $\oplus$ : concatenation.

the global shape structure, while the edge features capture the local neighborhood information. This adapted version is called MESHGRAPHNETS (ABS). Further, we adopt the structure of the DGCNN architecture to increase the ability of extracting local and global properties. Instead of stacking  $L$  message passing block, all the message passing blocks are concatenated to calculate the global feature vector. Then, again the local and global features are concatenated and transformed to the desired output by the decoder MLP. This adapted version is called MESHGRAPHNETS (ABS+POOL).

All MLPS in the encoder, message passing blocks and decoder are using three fully-connected layers of size 128. The two hidden layers come with ReLU activation, followed by a non-activated output layer. The MLPs except the decoder are followed by a LayerNorm (Ba et al., 2016) layer. Similar to our method, we are using  $L = 5$  message passing block.

## A.4 ADDITIONAL RESULTS

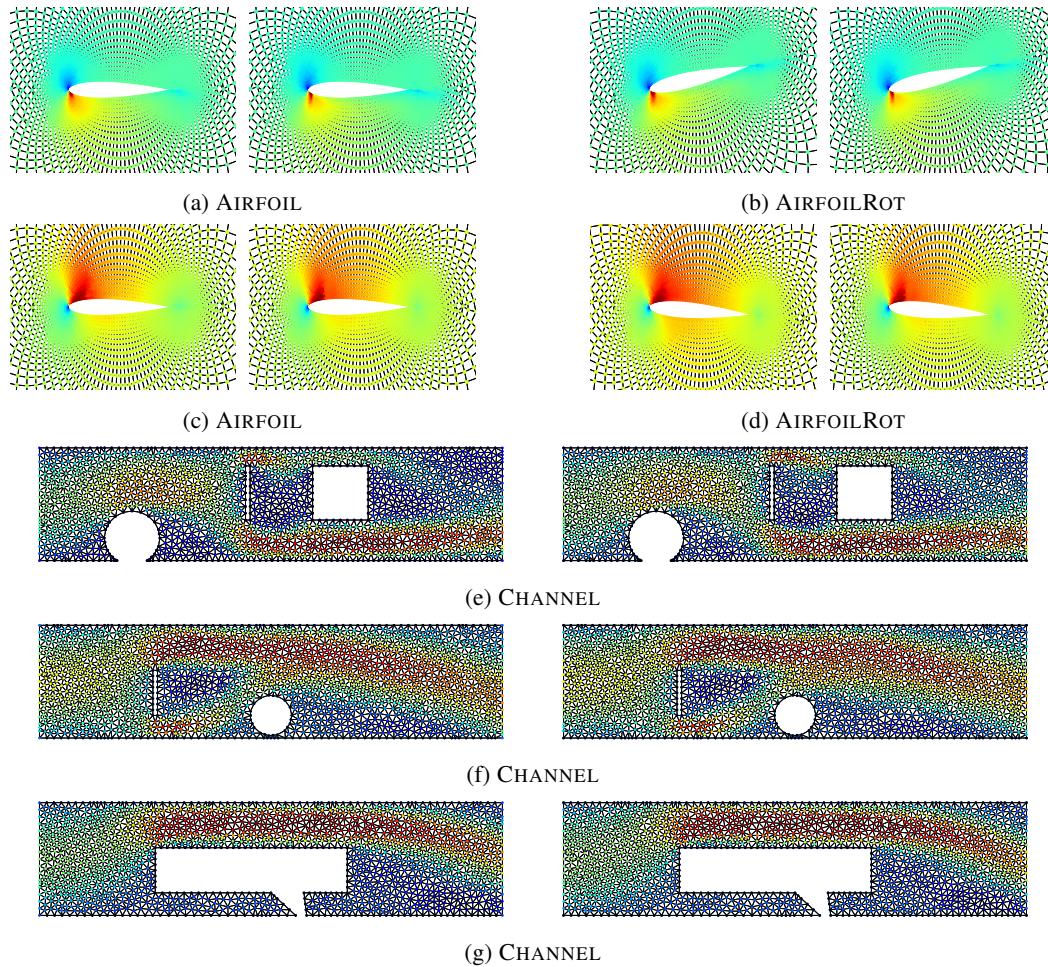


Figure 4: Additional examples of the predicted velocity field in  $x$ -direction for all three data sets, each with the ground truth simulation to the right.