GRAPH NEURAL NETWORKS FOR AERODYNAMIC FLOW RECONSTRUCTION FROM SPARSE SENSING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sensing the fluid flow around an arbitrary geometry entails extrapolating from the physical quantities perceived at its surface in order to reconstruct the features of the surrounding fluid. This is a challenging inverse problem, yet one that if solved could have a significant impact on many engineering applications. The exploitation of such an inverse logic has gained interest in recent years with the advent of widely available cheap but capable MEMS-based sensors. When combined with novel data-driven methods, these sensors may allow for flow reconstruction around immersed structures, benefiting applications such as unmanned airborne/underwater vehicle path planning or control and structural health monitoring of wind turbine blades. In this work, we train deep reversible Graph Neural Networks (GNNs) to perform flow sensing (flow reconstruction) around two-dimensional aerodynamic shapes: airfoils. Motivated by recent work, which has shown that GNNs can be powerful alternatives to mesh-based forward physics simulators, we implement a Message-Passing Neural Network to simultaneously reconstruct both the pressure and velocity fields surrounding simulated airfoils based on their surface pressure distributions, whilst additionally gathering useful farfield properties in the form of context vectors. We generate a unique dataset of Computational Fluid Dynamics simulations by simulating random, yet meaningful combinations of input boundary conditions and airfoil shapes. We show that despite the challenges associated with reconstructing the flow around arbitrary airfoil geometries in high Reynolds turbulent inflow conditions, our framework is able to generalize well to unseen cases.

1 INTRODUCTION

Many engineering applications stand to benefit from the ability to sense and reconstruct fluid flow features from sparse measurements originating at a structure's surface. Flow sensing could be crucial for improvements in the accuracy and resilience of wind turbine and unmanned aircraft controllers. Another possible application is monitoring of wind loaded structures (Barber et al., 2022), where the use of cheap micro-electromechanical systems (MEMS) in combination with novel methods for flow sensing could lead to robust structural health monitoring solutions. In this work, we focus on common aerodynamic structures: we aim to reconstruct the flow around 2-D airfoils. Traditionally, computing the flow around an airfoil requires approaches from Computational Fluid Dynamics (CFD), which are forward-physics simulators. In CFD, the inflow, outflow and wall boundary conditions are set, and over many iterations a solution for the discretized Navier-Stokes PDEs is reached, which then yields a pressure distribution at the airfoil surface. We aim to solve the inverse problem: given only the pressure distribution at the airfoil surface, a solution for the flow field and farfield boundary conditions is to be found. Moreover, our aim is to do so for any airfoil geometry subject to a wide variety of turbulent inflows.

Adopting the notation of Erichson et al. (2020), the problem can be described in the following manner. An airfoil equipped with p distributed barometric sensors is placed in a steady flow of air, providing surface pressure measurements $s \in \mathbb{R}^p$ at multiple locations around its perimeter. The sensors sample from the surrounding flow field $x \in \mathbb{R}^m$ through a measurement operator H:

$$s = H(x) \tag{1}$$

The goal is to construct an estimate of the flow field $\hat{x}$ surrounding the airfoil, by learning from training data a function $\mathcal{F}$ that approximates the highly nonlinear inverse measurement operator G such that:

$$\mathcal{F}(s) = \hat{x} \approx x = G(s) \tag{2}$$

Meshes are an extremely useful tool, indispensable in many engineering domains and especially in CFD. Contrary to Cartesian grid representations, mesh representations offer high flexibility for irregular geometries and allow for variable spatial density. This makes them ideal for discretizing complex physical problems, where one can balance the trade-off between numerical accuracy and computational efficiency in certain regions of interest. Furthermore, meshes can also be described in terms of nodes and edges, i.e. as a graph. In this context, the flow reconstruction problem can be described as follows. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{U})$ is constructed from an airfoil CFD mesh, with m fluid nodes $\mathcal{V}_f$ and p airfoil boundary nodes $\mathcal{V}_a$. The flow features of all $\mathcal{V}_f$ are unknown, whilst the features of $\mathcal{V}_a$ are known. Our aim is then to learn a graph operator $\mathscr{F}$ that estimates the information at the fluid nodes using the information contained at the airfoil boundary nodes, the input graph-level attributes $\mathcal{U}_{in}$ and the edges $\mathcal{E}$:

$$\hat{\mathcal{V}}_f = \mathscr{F}(\mathcal{V}_a, \mathcal{E}, \mathcal{U}_{in}) \tag{3}$$

An ancillary goal is to estimate the global context $\mathcal{U}$ of the graph, as this contains relevant information for applications. Figure 1 provides an description of the flow reconstruction problem in terms of graph learning.

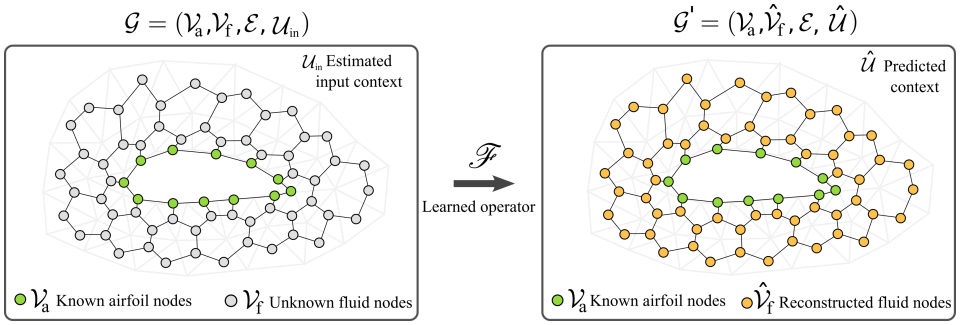

Figure 1: Problem setup. We aim to learn the reconstruction operator $\mathscr{F}$ which estimates properties of the fluid nodes as well as the graph context. This amounts to reconstructing a solution to the Navier-Stokes equations which satisfies the boundary conditions perceived at the airfoil surface.

From a geometric learning perspective, flow reconstruction is a challenging problem for several reasons. The first significant hurdle to overcome is the size of our graphs. We use meshes with high densities close to the airfoil in order to achieve good spatial resolution in these critical regions. Thus, our dataset contains graphs with a mean of around 60'000 nodes, which is an order of magnitude higher than previous mesh-graph learned simulation methods (Pfaff et al., 2020). Moreover, the input information is concentrated in a very localized domain of the graph: the airfoil nodes. It is difficult to propagate the necessary information to reconstruct nodes far from the airfoil with a shallow Graph Neural Network (GNN), meaning that deep GNN architectures with a large number of message-passing steps are required to push this 'information barrier' away from the airfoil nodes. However, deep GNNs go hand in hand with other issues such as high large memory requirements, over-smoothing, and over-squashing.

In this work, we combine a number of existing graph-learning methods to tackle the aforementioned challenges. Our contributions may be summarized as follows:

- We combine unknown Feature Propagation (Rossi et al., 2021) with very deep Grouped Reversible GNNs (Li et al., 2021) to reconstruct flow features at the fluid nodes, whilst additionally gathering contextual farfield information.
- Generalization of 2D aerodynamic flow field learning with GNNs, including (1) arbitrary airfoil geometries, (2) arbitrary turbulent inflow conditions (flow velocity, turbulence intensity, and angle of attack), and (3) simultaneous flow field reconstruction and inference of contextual farfield flow information based only on sparse pressure data on the surface of the airfoil.

- Generation of a unique training dataset of OpenFOAM airfoil CFD simulations with many different geometries and inflow conditions which are parsed to graph structure and made publicly available.
- We gather qualitative and quantitative results on unseen airfoil and flow configurations, and perform a number of experiments to understand the limitations of this framework. In particular, w test and compare three different GNN layer architectures.

2 RELATED WORK

Machine learning methods have recently garnered interest in the fluid mechanics community (Brunton et al., 2020). Fluid-related problems are typically nonlinear, complex and generate large amounts of data, all of which are conditions under which deep learning approaches thrive. Specifically in the context of flow reconstruction from sparse measurements, several neural network approaches can be found in the literature. In an article by Erichson et al. (2020), a "Shallow Neural Network", i.e. a fully-connected network with only two hidden layers, was applied to estimate transient flows from sparse measurements. The authors trained the networks on a single specific geometrical flow configuration, for example the flow behind a cylinder, and then tested on the same configuration at different time steps. We aim to avoid this limitation as our goal is to estimate the flow around any airfoil geometry. The authors compare their findings against typically used proper orthogonal decomposition (POD) methods and note significant improvements in terms of the reconstruction error. This work was then extended to turbulent flow reconstruction around airfoils based on experimental data in Carter et al. (2021), where the results were compared to Particle Image Velocimetry (PIV) measurements. The viability of neural networks over other approaches was further confirmed in the work of Fukami et al. (2020), where multiple methods were pitted against each other to estimate the flow behind a cylinder and an airfoil. Again, here the models were trained and tested on a single flow configuration, while also being dependent on Cartesian geometrical inputs. In Özbay & Laizet (2022), researchers attempt to avoid this limitation by utilizing Schwarz–Christoffel mappings to sample the points at which the flow is reconstructed, thus rendering the method geometry invariant. The authors train multiple neural network architectures on a collection of transient flow simulations around randomly generated 2-D Bezier shapes at a predefined inflow Reynolds number. As inputs for the flow reconstruction, they use multiple pressure sensors on the shapes' surface as well as velocity probes in the wake. Their results indicate that, when compared to a Cartesian sampling strategy, a significant performance boost is achieved for all neural network types, especially in the vicinity of the immersed shape. While this work demonstrates robustness to various geometric configurations, it requires additional velocity sensors and is trained on a singular farfield boundary condition, both of which we aim to avoid and improve upon. Another method which avoids geometrical dependency is reported in Chen et al. (2021). In this work, to which our approach most closely relates to, the authors utilize a Graph Convolutional Network (GCN) (Kipf & Welling, 2016) on graphs constructed from CFD meshes of randomly generated Bezier shapes. The GCN is used to predict the flow around the shapes at a fixed laminar (Reynolds number of 10) inflow condition without using surface measurements. In our approach, we aim to reconstruct a wide variety of turbulent flows given only surface readings, a significantly less constrained problem. We also aim to characterize the global properties of the flow, similarly to Zhou et al. (2021). To our best knowledge, we are the first to attempt to simultaneously reconstruct the flow while estimating turbulent inflow parameters at large Reynolds numbers for arbitrary airfoil geometries.

Graph networks are based on the message-passing framework (Gilmer et al., 2017), where a nodes features are updated by aggregating messages emanating from its neighbors. Many different types of message-passing schemes can be constructed, with some using attention mechanisms (Veličković et al., 2017) and others relying on strong theoretical backgrounds (Xu et al., 2018). Graph learning methods are increasingly being applied to a wide variety of physics problems (Sanchez-Gonzalez et al., 2018; 2020). In Pfaff et al. (2020), the authors successfully demonstrate how GNNs can learn to replicate forward mesh-based physics simulators and are able to predict the evolution of a transient solution. Motivated by these results, our approach is constructed upon the the same basic Encode-Process-Decode network structure. However, a key difference to note is that, contrary to the next-step prediction problem, flow reconstruction has to overcome high amounts of missing information, with known features being extremely localized. To address this hurdle, we turn to graph-based feature propagation methods (Rossi et al., 2021), which is closely related to matrix completion approaches (Monti et al., 2017). Feature propagation is an effective yet computationally inexpensive method for

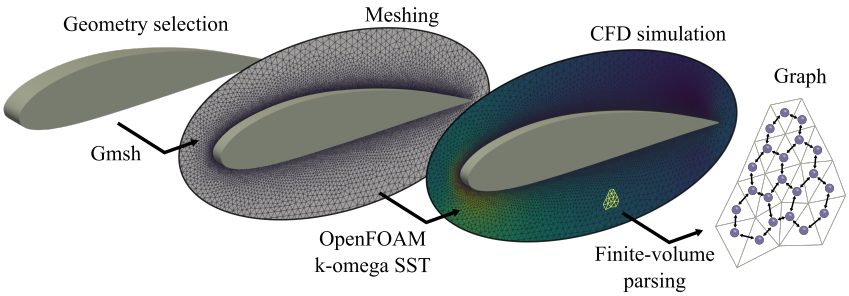

Figure 2: Illustration of the dataset generation pipeline. Airfoil shapes are selected at random from a database, then meshed and simulated in OpenFOAM with random feasible boundary conditions. In the last step, the finite-volume scheme is used to parse the simulation mesh into a graph.

initializing graphs with missing features. We use this method as pre-processing step, through which graphs are passed before being fed into the rest of the GNN model.

Training GNNs for very large graphs is challenging, with typical approaches tending toward minimizing the number of learnable parameters so that the problem becomes tractable (Chen et al., 2020). This often results in relatively shallow GNNs, which could adversely influence the propagation of the information contained at the airfoil nodes through sufficient extents of the graph. Making use of subgraph sampling strategies (Hamilton et al., 2017) is another possible approach, one which also allows for larger/deeper GNNs. However, these methods are not applicable in our case, as we need to feed entire graphs in one pass due to the heterogeneity in information localization. Moreover, subgraph sampling would yield additional difficulties for our ancillary goal of predicting global graph properties for farfield estimation. Recent work by Li et al. (2021) has shown that it is possible to train very deep GNNs on large graphs by making use of Grouped Reversible layers, which reduces memory requirements at the cost of extra computation. This method forms the core of the processing block of the proposed flow reconstruction GNN. Another issue which traditionally characterizes training deep GNNs on large graphs is over-squashing (Alon & Yahav, 2020). Over-squashing is a by-product of a graph's structure, where bottlenecks and tree-like structures (Topping et al., 2021) can cause the latent representation of certain nodes to be overwhelmed by the amount of information needed to be stored. We make use of this information while parsing the simulation meshes into graphs.

3 DATASET GENERATION

In this section we introduce the different elements of our data generation pipeline. In total we generate 1120 converged simulations, which are separated into train, validation and test datasets in a 80/10/10 split. Figure 3 depicts an illustration of this pipeline.

Geometry selection and meshing In our dataset generation pipeline, airfoil shapes are drawn at random from the UUIC database of airfoils (Selig, 1996). Before passing the shape to the meshing algorithm, we carry out some additional interpolation and processing to make sure that the selected airfoil has a sufficient amount of points at the leading-edge as well as a properly defined trailing-edge. Then, we use Gmsh (Geuzaine & Remacle, 2020) to construct an unstructured O-grid type mesh around the selected airfoil. A sizing field is set close to the airfoil in order to make sure that meshes with appropriate y+ values for the CFD wall-functions are generated. An overall sizing parameter is also set for sufficient farfield density, but is adjusted to ensure that an acceptable amount of cells are created ($< 150'000$).

CFD simulations Each mesh is associated with a different inflow configuration. Three parameters control the farfield conditions: angle of attack, inflow velocity and turbulence intensity. These parameters are drawn from probability distributions reflecting realistic atmospheric flows at Reynolds numbers ranging from $2 \cdot 10^5$ to $6.5 \cdot 10^6$, with a mean Reynolds number of around $3 \cdot 10^6$. This mean value is well beyond the typical laminar-to-turbulent transition threshold of around $5 \cdot 10^5$ (Incropera et al., 1996), which greatly increases the difficulty of the flow reconstruction problem. The farfield conditions form the global context vectors of our graphs and are estimated at inference time. We

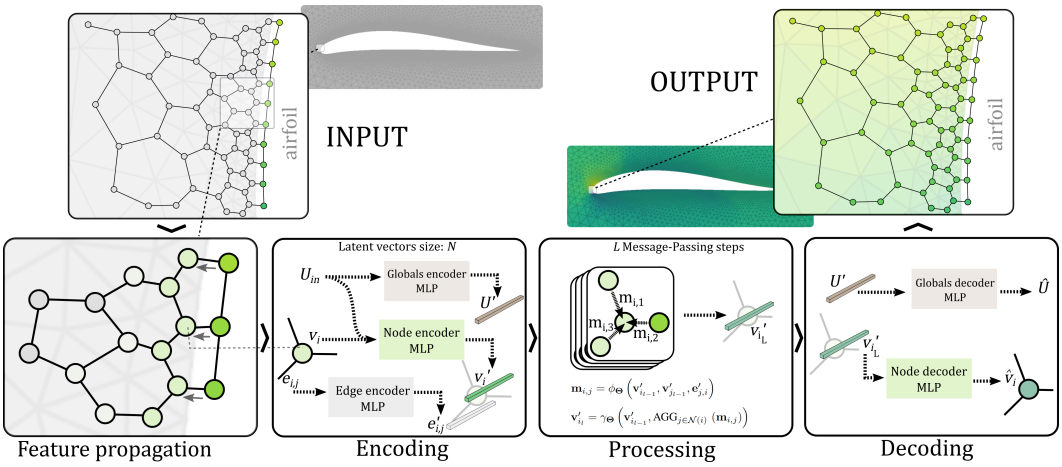

Figure 3: Overview of the GNN-based flow reconstruction pipeline. Feature Propagation is used to initialize the unknown fluid nodes. The graph is then passed through the Encode-Process-Decode pipeline, to obtain the reconstructed flow. During the Process step, node features are updated via message-passing within a deep Grouped Reversible GNN.

simulate the flow around the airfoils using a steady 2-D Reynolds-Averaged Navier–Stokes (RANS) CFD solver with the OpenFOAM software package (Jasak et al., 2007). For turbulence modelling, we select the K-Omega SST model (Menter & Esch, 2001) along with the standard OpenFOAM wall-functions for boundary layer treatment. Only sufficiently converged CFD simulations with pressure, velocity and turbulence residuals below $5 \cdot 10^{-5}$ are kept.

Graph parsing Contrary to many other mesh-based physics simulators, CFD solvers such as OpenFOAM are based on finite volume methods. It is a significant difference that should be reflected in the manner in which a mesh is converted into a graph. To do so, we use the cells themselves as the nodes, with bidirectional edges being formed between adjacent cells. This allows us to gather an additional edge feature which is relevant to the underlying physics. Specifically, the length (or surface for 3-D meshes) of the boundary between two cells is used as an edge feature. The benefit of this is twofold: a form of sizing is fed to the network and a quantity relevant to flux computation is set on the edges. For the nodes, we gather 4 types of features: pressure, x-velocity component, y-velocity component and node category (fluid, farfield, wall), the latter of which is inputted as a one-hot vector. The global context features are the farfield conditions (turbulence intensity, inflow velocity and angle of attack). To avoid unnecessary computational overhead, we do not parse the entire CFD domain, which has a radius of 100 airfoil chord lengths, into a graph. Instead we opt to only keep cells within a 1 chord circle centered on the airfoil. Furthermore, we set the airfoil nodes to be located at the meshed airfoil boundary and add bidirectional edges between adjacent airfoil nodes. These additional edges are created with the aim of avoiding tree-like structures in our graphs, as these could potentially cause bottlenecks for the learning process (Topping et al., 2021).

4 Graph Neural Network Framework

4.1 Architecture

For our GNN architecture, we adopt the Encode-Process-Decode logic that is now popular for learning on graph-based physics problems (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020; Godwin et al., 2022), albeit with some notable modifications.

Input features In the flow reconstruction problem, we assume that the global context of the graph is unknown, however some useful physical parameters describing the flow can be estimated. Using Bernoulli's principle, and given that all airfoils are simulated with a zero farfield static pressure, the

farfield inflow velocity magnitude can be initially approximated as:

$$\hat{U}_\infty = \sqrt{\frac{2 \cdot p_0}{\rho}} \tag{4}$$

where ρ is the density of air (constant throughout simulations) and p_0 is the total pressure measured at the stagnation point, which can be estimated by taking the maximum pressure at the airfoil nodes $p_0 = \max(p_{\mathcal{V}_a})$. While Bernoulli's principle is not valid for turbulent flows such as the ones we try to reconstruct, it serves as a good starting point for farfield velocity estimation. Another useful graph property that can be extracted is the normal force coefficient acting on the airfoil. While the lift coefficient is usually used to characterize airfoils, it cannot be calculate as the angle of attack is unknown (a quantity to be inferred from the learned model). Nevertheless, the normal coefficient is directly related to the lift coefficient and brings additional physical information which may aid the network to reconstruct the flow. It can be estimated via the following equation:

$$\hat{C}_n = \frac{\sum_{v_i \in \mathcal{V}_a} p_{v_i} \cdot l_{v_i} \cdot n_{y,v_i}}{p_0} \tag{5}$$

where l is the boundary length and n_y is the y component of the normal boundary vector, both of which are known properties for each mesh cell boundary. We therefore use $\mathcal{U}_{in} = (\hat{U}_\infty, \hat{C}_n)$ as the two-dimensional input context vector.
For the nodes, we only have access to the pressure distribution at the surface of the airfoil, while it is set to 'NaN' values at the fluid nodes. The type of each node is known and is encoded as a one-hot vector, bringing the total number of input node features to four. To account for mesh geometry, the following four edge features are used as inputs: the x and y components of the relative edge direction vector, the edge length, and the cell boundary length value l (see Section 3).

Pre-processing Both the input and target node features are normalized. To avoid data leakage, all pressure features are normalized by the mean and standard deviation of the known airfoil surface pressure distribution, while both components of the velocity target features are normalized by the initial estimated farfield velocity $\hat{U}_\infty$. We use Feature Propagation (Rossi et al., 2021) as a preliminary step before feeding a graph to our GNN. This step is an important part of our framework as it conditions the input graph into a plausible initial state. Essentially, the feature propagator radiates surface pressure information outwards. In most cases, we found 20 feature propagation iterations to be sufficient.

Encoding In the encoding layer, ReLU activated MLPs with two hidden layers and LayerNorm are used to project the input features of the graph into latent vectors of size N. This encoding layer differs to the standard GraphNet encoder (Sanchez-Gonzalez et al., 2020) in that the node encoder MLP takes as input the input node features as well as the latent global vector. We make this modification so that graph-level attributes are taken into account in the construction of the node latent variables, as this is not the case in the processing steps.

Processing For the processing step, we opt to use a deep Grouped Reversible GNN (Li et al., 2021) with L message-passing layers. This architecture makes modifications to the typical GNN architecture by first splitting the input node feature matrix V across the feature dimension into C groups $\langle V_1, V_2, ..., V_C \rangle$, which are then processed into grouped outputs $\langle V'_1, V'_2, ..., V'_C \rangle$ with a Grouped Reversible GNN layer. These outputs are computed as follows:

$$\begin{aligned} V'_0 &= \sum_{k=2}^{C} V_k \\ V'_k &= f_{wk}(V'_{k-1}, A, E) + V_k, \quad k \in \{1, \ldots, C\} \end{aligned} \tag{6}$$

with A the adjacency matrix and E the edge feature matrix.
The Grouped Reversible framework allows for any type of message-passing architecture to be chosen for the GNN layer f_{wk}. We choose to test three popular types of GNN layers: the Graph Attention Network (GAT) (Veličković et al., 2017), the modified Graph Isomorphic Network (GIN) (Xu et al., 2018) which accounts for edge features (Hu et al., 2019), and the Generalized Aggregation Networks (GEN) Li et al. (2020) which modifies the standard GCN with different aggregation schemes while also utilizing edge features.

Decoding Only the nodes and global context are decoded back into feature space, as the edges are not updated. Both decoding neural networks are MLPs with two hidden layers and ReLU activations without any output normalization. At the output of the decoder, we gather for each node the estimated pressure and velocity fields. The output context vector is composed of an updated version of the farfield velocity, as well as an estimation of the inflow angle (angle of attack) and of the turbulence intensity.

4.2 TRAINING

General aspects Our models are trained on a dataset composed of 896 graphs. Models are trained with the Adam optimizer on a single Nvidia GPU with 10GB of VRAM. Due to the size and nature of the graphs, we can only use a batch size of one, albeit with random order shuffling occurring at each epoch. The learning rate is initially set at $5 \cdot 10^{-4}$ and is exponentially decayed by a factor of 0.97.

Loss function We use a multi-component loss function, in order to minimize both the node feature reconstruction error and the context vector prediction error, with L_2 losses for both components. We further experiment with an additional physics-based loss component, which aims to enforce divergence-free velocity reconstruction. The velocity divergence at each fluid node v_i of the bidirectional graph can be computed by summing the mass flux along each edge. The divergence loss then compares for each node this value against zero. We provide more details regarding the implementation of the divergence loss in the supplemental material. The overall loss is then:

$$\mathcal{L} = L_2(\mathcal{V}, \hat{\mathcal{V}}) + \lambda \cdot L_2(\mathcal{U}, \hat{\mathcal{U}}) + \gamma \mathcal{L}_{div} \tag{7}$$

where λ and γ are hyperparameters used to balance the different components. In practice, we usually set λ to 1, and if the divergence loss is used, γ is also set to 1.

5 RESULTS

Our trained models are tested on a dataset comprising of 112 unseen airfoil simulations, each with a different combination of turbulent inflow parameters. We show here qualitative and quantitative results for our models and perform experiments aiming to investigate limitations and improvements of the proposed framework.

Comparison of GNN layers In Table 1, we gather and compare results for the three different types of GNN layers used in the Processor: GAT, GIN and GEN. The architecture of the Encoder and Decoder networks were kept constant, with a latent size for the node, edge and global features of $N = 128$. In the Grouped Reversible Processor, the number of Layers was set to $L = 30$, while the number of groups was chosen as $C = 4$. To obtain a consistent number of learnable parameters, the hyperparameters for each GNN layer type were carefully selected, more information about the different configurations can be found in the appendix.

Table 1: Root mean squared prediction errors averaged over the test dataset for both the flow reconstruction task and the global parameter estimation task. Results are averaged over 3 runs with different initializations. All models have latent size of $N = 128$ and $L = 30$ layers. While the revGAT model performs the best in terms of pressure and y-velocity reconstruction, it is outperformed by the revGIN model when it comes to x-velocity reconstruction, and graph-level attribute prediction.

	Node reconstruction RMSE			Global parameter prediction RMSE		
	pressure [Pa]	x-velocity [m/s]	y-velocity [m/s]	farfield velocity [m/s]	angle of attack [°]	turbulence intensity [-]
revGAT	**77.98** $\pm$ **17.99**	7.96 $\pm$ 1.19	**2.69** $\pm$ **0.53**	0.92 $\pm$ 0.17	4.16 $\pm$ 0.10	0.04 $\pm$ 0.003
revGIN	158.37 $\pm$ 5.16	**6.23** $\pm$ **0.32**	4.43 $\pm$ 0.22	**0.45** $\pm$ **0.06**	4.12 $\pm$ 0.09	**0.03** $\pm$ **0.003**
revGEN	137.51 $\pm$ 17.31	10.81 $\pm$ 0.25	5.47 $\pm$ 0.05	0.46 $\pm$ 0.05	**4.12** $\pm$ **0.01**	0.04 $\pm$ 0.002

Velocity reconstruction One of the more challenging prediction tasks is the estimation of the velocity field away from the airfoil. Accurately capturing velocity shear and recirculation regions is

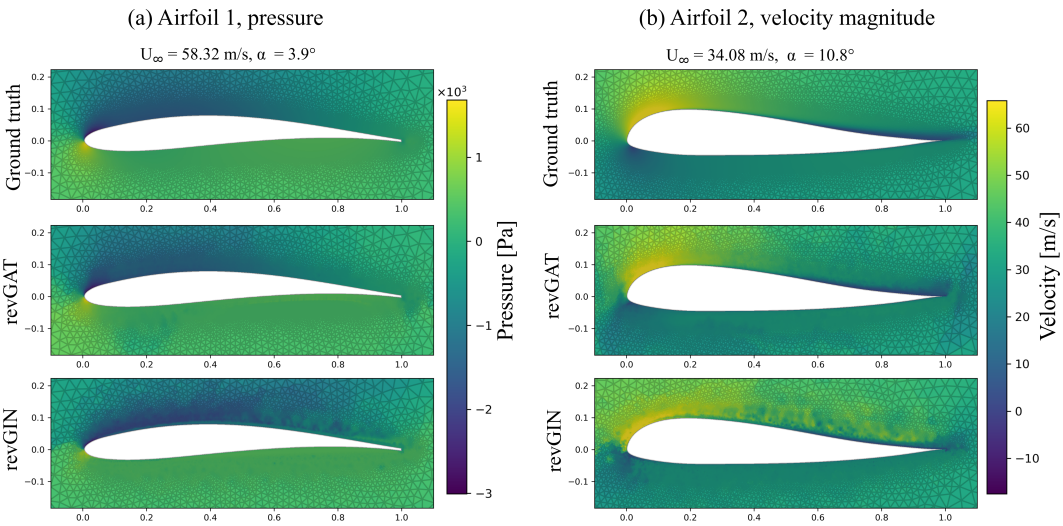

Figure 4: Reconstructed flow around two unseen arbitrary airfoils geometries at different inflow configurations for the revGAT and revGIN models. Figure (a) plots the reconstructed pressure field while the reconstructed velocity field is shown in Figure (b). For each case the ground truth is shown for comparison. Qualitatively, the revGAT model displays fewer artefacts in the solution.

non-trivial even for CFD simulators and is highly dependant on the airfoil shape and the inflow angle. To complete this task sucessfully, the GNN needs to be expressive enough to propagate relevant information throughout the graph. Table 2 summarizes the prediction errors of the velocity field in multiple concentric regions around the airfoil for the three models. We observe that for all three models, the error on the x-velocity decreases further away from the airfoil, but this is not the case for the y-velocity.

Table 2: Root mean squared prediction errors of the velocity components for different concentric regions of the flow, averaged over the test dataset. Each region is defined as the interior of an ellipse with a the length of semi-major axis and b the length of semi-minor (in chord lengths). Results are averaged over 3 runs with different initializations.

	x-velocity [m/s]			y-velocity [m/s]		
	revGAT	revGIN	revGEN	revGAT	revGIN	revGEN
region 1 (a=0.6, b=0.1)	8.26 ± 1.34	**6.46 ± 0.39**	11.44 ± 0.25	**2.48 ± 0.60**	4.15 ± 0.18	5.40 ± 0.03
region 2 (a=0.7, b=0.15)	7.98 ± 1.25	**6.30 ± 0.35**	11.00 ± 0.26	**2.49 ± 0.43**	4.40 ± 0.23	5.52 ± 0.05
region 3 (a=0.8, b=0.2)	7.95 ± 1.24	**6.27 ± 0.33**	10.92 ± 0.26	**2.61 ± 0.55**	4.44 ± 0.23	5.53 ± 0.05

Qualitative results Figure 4 displays some qualitative results for our two best performing models (revGAT and revGIN), compared to the CFD simulation ground truth. These results highlight the fact that the learned model is able to reconstruct flow features well, albeit with some artefacts. As the distance to the airfoil increases, these defects become more apparent. Moreover, some parts of the flow are not well captured. This is the case for flows exhibiting long wakes. On the other hand, we notice that flow features near the leading edge of the airfoil are in general well captured. We provide additional examples of reconstructed flows in the supplemental material.

Farfield estimation Figure 5 displays the graph-level context prediction results evaluated on the test set for the revGIN model. The GNN is able to accurately predict farfield inflow velocity, owing to the good initial farfield estimation provided as an input. For the angle of attack estimation, we observe good results at small angles but less so for larger positive and negative angles. Prediction of the turbulence intensity is however relatively poor, which can be attributed to this variable having a lesser impact on the airfoil pressure distribution. Moreover, this variable is not directly set in the CFD

simulations as it is used to calculate turbulent boundary conditions (kinetic energy k and specific rate of dissipation ω of the $k - \omega$ turbulence model), which makes it more difficult to retrieve in this inverse context.

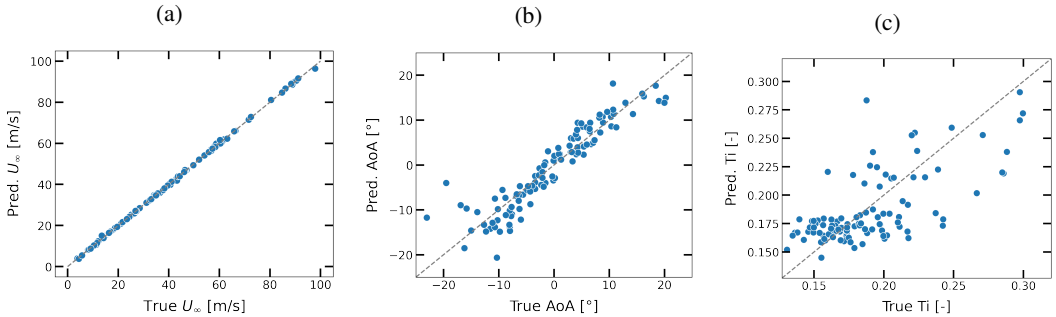

Figure 5: Comparison of the global properties predicted by our revGIN model against the ground truth values, evaluated on the test dataset. The model is able to accurately predict farfield flow velocity (a) and to a lesser extent the angle of attack (b), but it falls short for farfield turbulence intensity prediction (c).

6 Discussion

Our results indicate that the type of GNN architecture chosen in the Grouped Reversible Processor has a clear impact on the flow reconstruction quality. From our comparison, we find that, overall, using Graph Attention Network layers usually yields the best reconstructed solutions. However, we also observe that the Graph Isomorphic Network layer is better able to capture detached flows (see Appendix C). Perhaps a combination of the two could lead to better reconstructed flows, which could be for instance implemented by alternating GIN and GAT layers. Lastly, we find that the performance of the GEN layer to be somewhat underwhelming.

Something to note is that the first few layers of nodes surrounding the airfoil carry a disproportionate amount of relevant information which needs to be propagated outwards to an increasing number of nodes, thus creating artificial tree-like paths within the graph. While the Grouped Reversible framework provides a good way to train deep networks in order to circumvent this, other methods may also feasible. One possible solution might be to use hierarchies such as those implemented in Martinkus et al. (2021). Another option could be to apply a message-passing GNN in an iterative manner, with each step reconstructing increasingly large concentric bands around the airfoil.

Physics-driven learning methods could also lead to potential improvements. For instance, the message-passing framework may well be suited to incorporate elements from the Lattice-Bolzmann method (Chen & Doolen, 1998) as it also functions in a similar two step algorithm (collision and streaming). Another possible option would be to minimize the gradient of the pressure, as described in Taha & Gonzalez (2021).

7 Conclusion

In this work we applied deep graph-based learning techniques to reconstruct pressure and velocity fields around arbitrary airfoil geometries subject to high-Reynolds turbulent flows. We show that, despite the challenges posed by this problem, such as the large graphs and the very localized input information, our Flow-Reconstruction GNN framework is able to provide good reconstructed solutions, and infer contextual farfield flow information. We compared several message-passing architectures within the Grouped Reversible Processor GNN, and found that Graph Attention Network layers yielded the best reconstructed solutions. This work provides a flexible framework which may easily be applied to other mesh-based inverse physics problems, and which may be of significant interest to a number of engineering applications.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Sarah Barber, Julien Deparday, Yuriy Marykovskiy, Eleni Chatzi, Imad Abdallah, Gregory Duthé, Michele Magno, Tommaso Polonelli, Raphael Fischer, and Hanna Müller. Development of a wireless, non-intrusive, mems-based pressure and acoustic measurement system for large-scale operating wind turbine blades. *Wind Energy Science*, 7(4):1383–1398, 2022.

Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

Douglas W Carter, Francis De Voogt, Renan Soares, and Bharathram Ganapathisubramani. Data-driven sparse reconstruction of flow over a stalled aerofoil using experimental data. *Data-Centric Engineering*, 2, 2021.

Junfeng Chen, Elie Hachem, and Jonathan Viquerat. Graph neural networks for laminar flow prediction around random 2d shapes. *arXiv preprint arXiv:2107.11529*, 2021.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Shiyi Chen and Gary D Doolen. Lattice boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364, 1998.

N Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L Brunton, Michael W Mahoney, and J Nathan Kutz. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, 476(2238):20200097, 2020.

Kai Fukami, Koji Fukagata, and Kunihiko Taira. Assessment of supervised machine learning methods for fluid flows. *Theoretical and Computational Fluid Dynamics*, 34(4):497–519, 2020.

Christophe Geuzaine and Jean-François Remacle. A three-dimensional finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, (11):79, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Jonathan Godwin, Michael Schaarschmidt, Alexander Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Veličković, James Kirkpatrick, and Peter Battaglia. Simple GNN Regularisation for 3D Molecular Property Prediction & Beyond. *arXiv preprint arXiv:2106.07971*, 2022.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

Frank P Incropera, David P DeWitt, Theodore L Bergman, Adrienne S Lavine, et al. *Fundamentals of heat and mass transfer*, volume 6. Wiley New York, 1996.

Hrvoje Jasak, Aleksandar Jemcov, Zeljko Tukovic, et al. Openfoam: A c++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pp. 1–20. IUC Dubrovnik Croatia, 2007.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pp. 6437–6449. PMLR, 2021.

Karolis Martinkus, Aurelien Lucchi, and Nathanaël Perraudin. Scalable graph networks for particle simulations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8912–8920. AAAI Press, 2021.

Florian Menter and Thomas Esch. Elements of industrial heat transfer predictions. In *16th Brazilian Congress of Mechanical Engineering (COBEM)*, volume 109, pp. 650, 2001.

Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *Advances in neural information processing systems*, 30, 2017.

Ali Girayhan Özbay and Sylvain Laizet. Deep learning fluid flow reconstruction around arbitrary two-dimensional objects from sparse sensors using conformal mappings. *AIP Advances*, 12(4): 045126, 2022.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Emanuele Rossi, Henry Kenlay, Maria I Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael Bronstein. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. *arXiv preprint arXiv:2111.12128*, 2021.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pp. 4470–4479. PMLR, 2018.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Michael S Selig. Uiuc airfoil data site. 1996. URL `https://m-selig.ae.illinois.edu/ads.html`.

Haithem Taha and Cody Gonzalez. A variational theory of lift. In *APS Division of Fluid Dynamics Meeting Abstracts*, pp. P17–006, 2021.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Kaiwen Zhou, Luanliang Zhou, Simeng Zhao, Xingyu Qiang, Yingzheng Liu, and Xin Wen. Data-driven method for flow sensing of aerodynamic parameters using distributed pressure measurements. *AIAA Journal*, 59(9):3504–3516, 2021.

APPENDIX

A DIVERGENCE LOSS

To compute the divergence (conservation of mass) loss of each node, we first calculate the mass flow along each edge. For a directional edge outgoing from node i to node j, this writes:

$$\dot{m}_{i,j} = \rho l_{i,j} \vec{U}_i \cdot \vec{n}_{i,j} \tag{8}$$

where ρ is the density of the fluid, $\vec{U}_i$ is the velocity vector at node i, $\vec{n}_{i,j}$ is the normalized edge direction vector (which is normal to the original mesh cell boundary) and $\cdot l_{i,j}$ is the surface (length in 2-D) of the cell boundary. The divergence loss at node i with k neighbours can then be computed by summing the mass flows and comparing this with zero using an L_2 distance metric:

$$\mathcal{L}_{div_i} = L_2(\sum_{j=1}^{k} \dot{m}_{i,j}, 0) \tag{9}$$

B ADDITIONAL HYPERPARAMETER DETAILS

For all models, 30 layers were used in conjunction with a latent space size of 128. The GAT model is implemented with 2 attention heads. Moreover a LayerNorm layer is applied to the output of each GAT layer, as it was found that this greatly aided training stability. Both the GIN and GEN implementation makes use of ReLu activated MLPs with two hidden layers and LayerNorm. In total each layer has approximately 4.5k learnable parameters.

C ADDITIONAL RESULTS

We plot in Figure 6 and Figure 7 some additional results, showcasing the reconstructive abilities of our models, as well as some configurations which are not well captured. Notably, detached flows are not always properly reconstructed, an issue which may stem from either the dataset not holding enough detached flows, or from challenges arising from the GNN architecture.

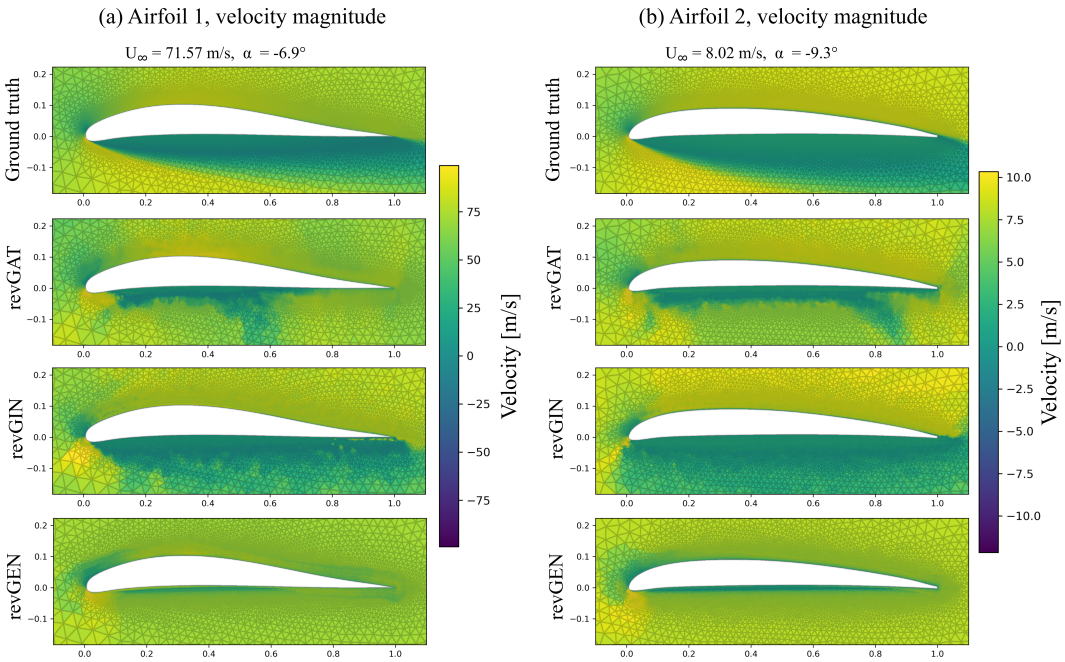

Figure 6: Reconstructed velocity field magnitude around two unseen arbitrary airfoils geometries at different inflow configurations for the revGAT, revGIN and revGEN models. For each case the ground truth is shown for comparison. Qualitatively, the revGIN model is able to better capture the detached flows in the solution.

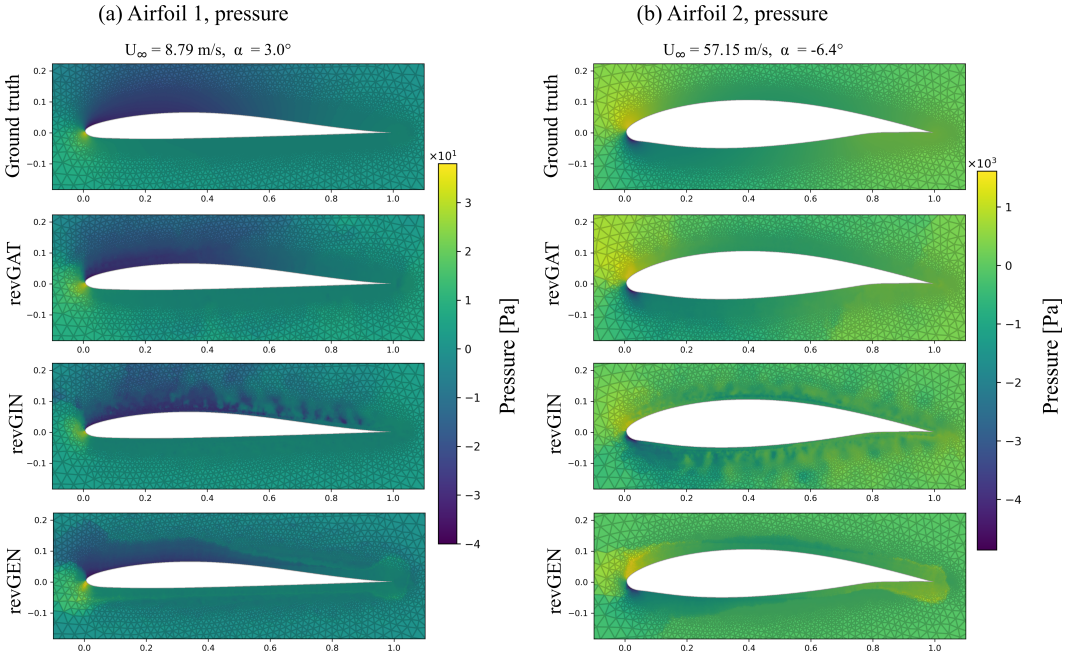

Figure 7: Reconstructed pressure field around two unseen arbitrary airfoils geometries at different inflow configurations for the revGAT, revGIN and revGEN models. For each case the ground truth is shown for comparison. Qualitatively, the revGAT model displays fewer artefacts.