Implicit Neural Representation For Accurate CFD Flow Field Prediction

Laurent de Vito [1] **Nils Pinnau** [2 3] **Simone Dey** [4 3]

Abstract

Despite the plethora of deep learning frameworks for flow field prediction, most of them deal with flow fields on regular domains, and although the best ones can cope with irregular domains, they mostly rely on graph networks, so that real industrial applications remain currently elusive. We present a deep learning framework for 3D flow field prediction applied to blades of aircraft engine turbines and compressors. Crucially, we view any 3D field as a function from coordinates that is modeled by a neural network we call the backbone-net. It inherits the property of coordinate-based MLPs, namely the discretization-agnostic representation of flow fields in domains of arbitrary topology at infinite resolution. First, we demonstrate the performance of the backbone-net solo in regressing 3D steady simulations of single blade rows in various flow regimes: it can accurately render important flow characteristics such as boundary layers, wakes and shock waves. Second, we introduce a hyper-net that maps the surface mesh of a blade to the parameters of the backbone-net. By doing so, the flow solution can be directly predicted from the blade geometry, irrespective of its parameterization. Together, backbone-net and hyper-net form a highly-accurate memory-efficient data-driven proxy to CFD solvers with good generalization on unseen geometries.

1. Introduction

There is a recent surge to devise fast deep learning models as a substitute for Computational Fluid Dynamics (CFD) solvers for the prediction of fluid flows (Vinuesa & Brunton, 2021). Indeed, CFD solvers are, though accurate, particularly slow, and this problem gets compounded in design optimization (Popov et al., 2020), possibly under manufacturing uncertainties (Kamenik et al., 2018; Meyer et al., 2019), through numerous calls to the flow solver.

Over few years, we have witnessed CNN-based models, followed by graph-based models and point cloud models for fluid flow prediction. It is worth noticing that those models are not intrusive. This is a key property for their wide-spread adoption in industry. They are data-driven: They consume data produced by expensive CFD solvers for training.

It is natural to re-interpret nodes of a computational mesh as pixels, and make use of Convolutional Neural Networks (CNNs) to solve expensive equations (Xiao et al., 2018; Gao et al., 2021) but this approach is clearly limited to equally-spaced Cartesian meshes (a regular grid in academic use only). Unstructured meshes[1] are widely used in the industry because they can adequately delineate complex geometries and easily deal with localized regions that require different resolution, e.g. cells are small near walls to capture the boundary layer where flow transitions are sharp whereas they are large in the freestream where gradients are smooth. To make unstructured meshes amenable to a treatment with CNNs, they are rasterized into voxel grids and processed using 3D volumetric convolutions (Thuerey et al., 2020; Aulich et al., 2019). With low resolutions, some information is inevitably lost during voxelization. Therefore, a high voxel resolution is required to preserve flow details. However, scalability is poor because the computational cost and memory requirement both increase cubically with voxel resolution. Thus, it is infeasible to train a voxel-based model with high-resolution grids.

A mesh can equivalently be viewed as a graph. So Graph Neural Networks (GNNs) are legitimate candidates for predicting flow fields (Pfaff et al., 2020; Meyer et al., 2021; Baque et al., 2018). Since they rely on message-passing, propagating the information from one node to a distant one requires stacking many graph convolutional layers. Pooling is an effective technique to increase the receptive field of nodes but what is a cheap operation in CNNs turns out to be challenging in GNNs (Grattarola et al., 2021).

[1]MTU Aero Engines AG, Munich, Germany [2]Microsoft, Munich, Germany, nils.pinnau@live.de [3]Work done during an internship at MTU Aero Engines AG, Munich, Germany [4]itestra, Munich, Germany, simi.dey@gmx.de. Correspondence to: Laurent de Vito <laurent.vito@mtu.de>.

ECCOMAS CONGRESS 2024 9^{th} European Congress on Computational Methods in Applied Sciences and Engineering.

[1]We regard body-fitted multi-block meshes as unstructured meshes in opposition to equally-spaced Cartesian meshes.

If the adjacency matrix of the graph is discarded, we end up with a point cloud. Many point cloud techniques are in fact graph techniques in disguise: A graph is built by linking a node to its nearest neighbors. Point cloud models for flow field prediction are uncommon (Kashefi et al., 2021).

Our work departs radically from those approaches that only yield a discrete representation of the flow field. Instead, we rely on an *implicit neural representation* of the flow field: A flow field is viewed as a function from coordinates that is approximated by a neural network. Given an input coordinate, the network is trained to output the value of the flow field at that coordinate. Such neural networks are commonly referred to as coordinate-based networks or simply coordinate networks in the literature. They have become popular since (Tancik et al., 2020) showed how to enable a neural network to learn high frequency functions in low-dimensional problems, typically in 2D and 3D.

Once trained on a simulation, our coordinate MLP, called here the *backbone-net*, offers a compressed mesh-agnostic representation of the flow field it was trained on. Indeed, our backbone-net is small, so its few weights can be stored instead of the full CFD solution. But apart for data reduction (Zhang et al., 2021; Huang & Hoefler, 2022), this is not particularly useful in itself.

In this work, we are concerned with steady CFD flow fields of single 3D blade rows of aircraft engine turbines and compressors. We would like to capture the change in flow solution when the blade geometry is varied. Since the flow solution is equivalently viewed as the weights of the backbone-net, we would like to adapt those weights with the geometry of the blade. This task is per definition devoted to a hyper-network (Ha et al., 2016). Our hypernetwork, here in short *hyper-net*, takes in the blade geometry in the form of a triangulated surface mesh, and yields the weights of the backbone-net. Our solution is thus independent of the blade parameterization. This makes our model even more appealing as a proxy for a CFD solver in an optimization: we can use the same model for different optimizations even if the degrees of freedom of the parameterization change. But there is another advantage in taking the surface mesh as input to our model: some geometrical features, e.g. fillets, small gaps or steps, are hard to describe in a compact form that can be passed to the model. Because our model ingests the surface mesh of the configuration, all those geometrical features are no longer obstacles to a faithful representation of the configurations. All in all, we substitute an optimized CFD flow solver by a single forward pass of a hyper-net. Whereas the backbone-net is trained on a single simulation, the hyper-net is trained on a set of simulations. The hyper-net is end-to-end differentiable and so it is trained using stochastic gradient descent.

The contributions of this paper are as follows:

- We introduce an implicit neural representation for CFD flow fields in turbomachines. As a result, our representation of flow fields on 3D unstructured meshes is compact and continuous. The advantage is threefold: (1) This representation does not change if the mesh is modified as long as the flow field is the same[2]; (2) The interpolation to unseen coordinates is smooth; (3) Because we can query the flow field at any coordinates, the flow field has virtually an infinite resolution, a property that was leveraged in image super-resolution (Klocek et al., 2019).
- Our method does not require extra preprocessing like the time-consuming and approximate interpolation of CFD flow fields onto a Cartesian mesh.
- We demonstrate that our backbone-net, a small coordinate MLP, with discrete Fourier features, can render the full 3D flow field of turbomachine configurations in various flow regimes, from subsonic to supersonic, faithfully. In (White et al., 2020), a mixture of experts was advocated where each mixture is a simple MLP. The rational behind this choice was that the flow solution is too complex to be accurately predicted by a single MLP. We challenge this claim and show that a coordinate MLP with a carefully tailored architecture can adequately predict CFD flow fields. By doing so, we circumvent the difficult problem of initializing mixture of experts (Makkuva et al., 2020).
- We establish the direct mapping of the blade geometry to the aerodynamic flow fields using a hypernetwork. The idea of using a hypernetwork is attractive because it can more generally modulate the (weights of) backbone-net based on any relevant information. In this work, we condition the flow solution on the blade geometry, but our framework can incorporate any other type of side information, e.g. boundary conditions.
- We empirically show that our model generalizes well from few samples.

We focus here exclusively on flow solutions of compressor and turbine blades, but this concept is not limited to internal aerodynamics: it can be applied more broadly to any solution of systems of partial differential equations (Pan et al., 2022).

2. Related Work

Our work is closely related to (Pan et al., 2022): In their work, a hyper-net they call ParameterNet produces the weights and biases of a backbone-net they call ShapeNet based on external factors. ShapeNet is also a coordinate-based MLP. They showed the benefits of this new paradigm

[2]For this property to hold, the flow is in the asymptotic regime and the mesh is modified keeping the mesh resolution unchanged.

in a broad range of applications notably in CFD. Furthermore, they compared this approach against other recent frameworks like DeepONet.

In this work, we leverage a similar framework and apply it in a real-world large-scale industrial setting. Whereas in (Pan et al., 2022) the input to their ParameterNet is a time stamp because their focus is on unsteady simulations, the input to our hyper-net is a blade geometry in the form of a surface mesh. Furthermore, their ShapeNet has sine activation functions (Sitzmann et al., 2020), but our backbone-net is equipped with deterministic Fourier features because we found sine activation functions to underperform.

Convolutional Neural Networks (CNNs). Traditional CNNs can be applied to predict the flow solution on regular grids, either in 2D or 3D (Guo et al., 2016; Aulich et al., 2019; Thuerey et al., 2020; Obiols-Sales et al., 2020; Chen et al., 2021), as well as on irregular structured grids by mapping them to canonical regular grids (Chen & Thuerey, 2021). However, CNNs do not scale well because computational and memory requirements grow cubicly with the 3D grid resolution. As a result, CNNs cannot exploit the detailed geometry of irregular bodies in fluids and cannot render important features such as boundary layers, wakes and shock waves.

Compared to CNNs, our model has a very low memory footprint, so the infrastructure cost for training and deploying our model is reduced. Furthermore, it faithfully predicts important flow features.

Graph Neural Networks (GNN). Opposite to CNNS, GNNs have the potential to work natively on arbitrary meshes. Impressive results haven been obtained recently with GNN-based methods that directly mimic the CFD solver iterative process in computing the solution on a mesh from one time-step to the next (Pfaff et al., 2020; Meyer et al., 2021). However, GNNs scale badly with the mesh size. The problem is that neighboring nodes affect each other in the learning process. For the computation of the embedding of a single node, a GNN needs the embeddings of the neighbors of this node. Consequently, GNNs must learn on the full graph, which is infeasible if the graph is too large. Mini-batching is a solution, but despite advances (Ding et al., 2021; Klicpera et al., 2021), training graph neural networks on large graphs is still challenging. By avoiding graph neural networks, our code is considerably simpler (no need to introduce edges) and more efficient (no information gathering at each node from neighbors) than (Pfaff et al., 2020). (Meyer et al., 2021) proposed a data-driven proxy that necessitates a hierarchy of meshes, which makes it cumbersome to code and certainly precludes its extension to 3D.

The output of GNNs is furthermore inherently discrete: the solution is available only at the nodes of the mesh. With our method, the solution can be queried anywhere in space (and time if time is included as input feature).

Furthermore, most graph-based methods are confined to 2D configurations in subsonic flow regime (Bonnet et al., 2022). We consider 3D configurations in subsonic up to supersonic flow regimes.

Hypernetworks. Hypernetworks are models that generate parameters for other models (Ha et al., 2016). Many recent works rely on hypernetworks (Kang et al., 2017; Sitzmann et al., 2019; Littwin & Wolf, 2019; Spurek et al., 2020; Knyazev et al., 2021; Skorokhodov et al., 2021). This success is rooted in the modularity property of hypernetworks (Galanti & Wolf, 2020). That hypernetworks outperform embedding-based methods was experimentally illustrated in (Skorokhodov et al., 2021).

Physic-Informed Neural Networks (PINNs). The models mentioned above are proxies (also called deep surrogates) to CFD solvers that need training data. On the opposite, physic-informed neural networks (Raissi et al., 2019) solve for the motion of fluid flows just as CFD solvers do. Hence, their generalization power is on par with that of CFD solvers. This property makes them highly attractive, but despite all the hype, they suffer from severe deficiencies: 1/ They have not matured yet as a drop-in replacement for CFD solvers as they currently cannot compete with advanced PDE solvers in terms of accuracy (Wang et al., 2021; Chuang & Barba, 2022) 2/ The laborious effort to encode all the equations to solve, including turbulent and transition models, and the various boundary conditions, is overlooked, though it is a huge undertaking (Du et al., 2022); 3/ Mass, momentum and energy are not conserved, because the solution is solved point-wise, whereas finite volume flow solvers have the mass, momentum and energy conservation property built-in. Those drawbacks hamper the wide-spread use of PINNs.

Implicit Neural Representation (INR). In implicit neural representation, a discrete signal, e.g. the color information on a lattice (an image) or on a low-dimensional manifold (a meshed shape), is represented as a continuous function by a neural network, usually a MLP with the ReLU activation function. Since the input to those networks are the low-dimensional coordinates, those networks are referred to as coordinate-based MLPs. Not only is the representation continuous but also compact. Coordinate-based MLPs have been used to represent images (Stanley, 2007), volume density (Mildenhall et al., 2020), occupancy (Mescheder et al., 2019), signed distance (Park et al., 2019) and have been employed in a variety of other tasks. Coordinate MLPs have difficulty learning high frequency functions (Rahaman et al., 2019). To overcome this limitation, (Mildenhall et al., 2020) propose to map the raw coordinates to deterministic Fourier features whereas in (Tancik et al., 2020) they are mapped

to random Fourier features. (Sitzmann et al., 2019) take another approach and use sinusoidal activation functions.

3. Models

A overview of the full model is given in Figure 1. All models are implemented in Python using PyTorch (Paszke et al., 2017). We emphasize that our implementation is simpler compared to other approaches: The hyper-net is a small residual networks (He et al., 2016) and the backbone-net is a fully-connected network, both do not require any specialized modules and so are easily coded using any deep learning framework.

3.1. The Backbone-Net

The backbone-net is a small coordinate-based MLP that takes in a coordinate vector $\boldsymbol{x}$ and outputs an approximation $\hat{\boldsymbol{y}}$ to the flow features $\boldsymbol{y}$ at that position. Typically, the geometry of a blade is specified by the design vector $\boldsymbol{\theta}$. From the design vector, the 3D computational domain around the blade is defined and meshed, and the flow solution for prescribed boundary conditions is computed at cell-centers by the flow solver. So the dataset for training the backbone-net is $\mathcal{D}^{b-net} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$ with cell-center coordinates $\boldsymbol{x}_n \in \mathbb{R}^{D_x}$ and flow features $\boldsymbol{y}_n \in \mathbb{R}^{D_y}$ for $n = 1, \cdots, N$ given by the flow solver. It is clear that the nature of the mesh — block-structured, unstructured or hybrid — is irrelevant.

The input $\boldsymbol{x}$ is optionally fed into a positional encoder PE as introduced in (Tancik et al., 2020) and the result is linearly transformed after concatenation with the input: $\boldsymbol{z}_0^{(b)} = \boldsymbol{W}_{in}^{(b)}[\text{PE}(\boldsymbol{x})\|\boldsymbol{x}]$. We omit the bias term of the linear transformation for sake of simplicity. $\|$ denotes concatenation. Concatenation of raw data $\boldsymbol{x}$ with Fourier features $\text{PE}(\boldsymbol{x})$ was found beneficial as in (Chen & Zhang, 2019; Jaegle et al., 2021). Opposite to (Tancik et al., 2020), the encoder does not construct random Fourier features. Following (Mildenhall et al., 2020), the positional encoding scheme is deterministic: The mapping contains only on-axis frequencies. The positional encoder has base frequency f_b and the L on-axis frequencies are defined as $f_l = 2\pi(2^l f_b)$ for $l = 0, \cdots, L-1$ (Mildenhall et al., 2020). We experimentally observed that the benefit of a positional encoding with $L > 4$ is negligible though it does not hurt performance.

Following the positional encoder is a stack of fully connected layers with identical layout:

$$\boldsymbol{z}_k^{(b)} = \sigma(\boldsymbol{W}_k^{(b)}\boldsymbol{z}_{k-1}^{(b)}), \quad k = 1, \cdots, K \tag{1}$$

σ designates the GELU activation function (Hendrycks & Gimpel, 2016). We finally insert a linear layer to convert to the expected output: $\hat{\boldsymbol{y}} = \boldsymbol{W}_{out}^{(b)}\boldsymbol{z}_K^{(b)}$. In all experiments, the positional encoder has base frequency $f_b = 0.5$ and the number of Fourier features is $L = 4$. The backbone-net has $K = 6$ layers and the hidden dimension is 112.

After the backbone-net is trained on the dataset $\mathcal{D}^{b-net}$ in a supervised manner, its weights are implicitly dependent on the design vector $\boldsymbol{\theta}$ that specifies the geometry of the blade for which we have the simulation. We would like to make this dependency explicit. Stated otherwise, we would like to directly get the flow solution (or equivalently the weights of the backbone-net denoted collectively by ϕ) as a function of the design vector, or, more generally, as a function of the blade geometry. For that purpose, we introduce a hyper-net that predicts the weights of the backbone-net from the blade geometry. By doing so, the expensive flow solver is replaced with a single forward pass of the hyper-net.

3.2. The Hyper-Net

Since the hyper-net yields the weights of another neural network, namely the backbone-net, the dataset for training the hyper-net is a dataset of simulations (configurations): $\mathcal{D}^{h-net} = \{(\mathcal{G}_m, \mathcal{D}_m^{b-net})\}_{m=1}^{M}$. $\mathcal{G}_m$ designates the geometry of the m-th blade in the form of a triangulated surface. It is equivalent to a graph. Each $\mathcal{D}_m^{b-net} = \{(\boldsymbol{x}_{m,n}, \boldsymbol{y}_{m,n})\}_{n=1}^{N_m}$ is the 3D flow solution for the blade geometry $\mathcal{G}_m$. The number of points N_m in each of those simulations can vary but they are the same in our experiments.

The input to the hyper-net is a fixed-size pseudo-design vector (or blade embedding), $\boldsymbol{\theta}'$, representing the geometry. It is obtained by passing the geometry through a graph neural network (GNN). Typically, a GNN uses a stack of message passing layers (Gilmer et al., 2017) to learn powerful embeddings compared to a plain MLP. However, they are computationally expensive to train. We experimentally found out that we can achieve comparable performance using the simple and much faster PointNet architecture (Qi et al., 2017). It is a MLP followed by global max pooling. This suggests that, at least for the task of predicting the CFD flow fields from the blade geometry (or its embedding), a complex architecture is not crucial. The input, a triangulated surface mesh, is thus treated as a point cloud. This is fortunate as the triangulation of the original surface mesh is not unique. In our experiments, the MLP is a residual network with 4 layers, a hidden dimension of 128 in the main branch and 192 in the residual branch with GELU nonlinearities. The pseudo design vector has dimension 16 which is surely greater than the intrinsic dimension of the datasets we consider experimentally. Indeed, the blade parameterization has at most 17 degrees of freedom, see Tables 3 and 4, and reconstruction of blades is as good as with pseudo design vectors of greater size, see Figure 23.

The hyper-net takes in the blade embedding $\boldsymbol{\theta}'$ and outputs

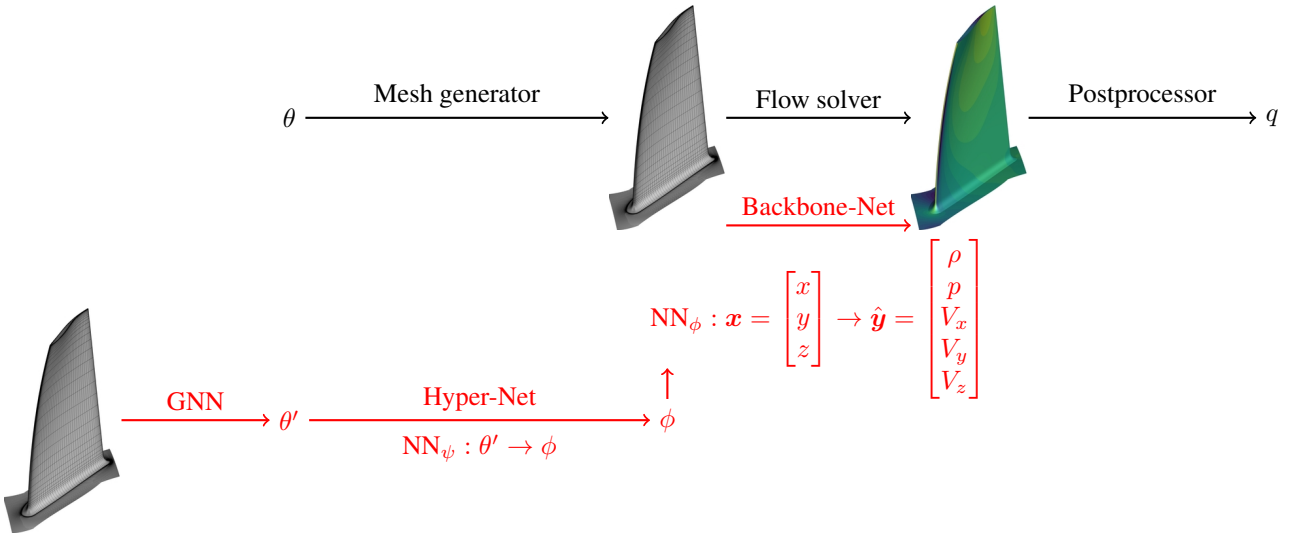

Figure 1. Overview of the backbone-net and hyper-net. At the top in black, we have the usual tool chain: From the blade design vector θ, a 3D mesh is generated (only the surface mesh is depicted), then the CFD solver computes the full 3D flow solution and eventually the postprocessor aggregates the local information at the cell-centers and outputs quantities of interest denoted by q like massflow and efficiency. At the bottom in red, we have our model that comprises the backbone-net and the hyper-net. The backbone-net maps the coordinates of a point of the computational domain to its flow features. It is specialized to a configuration. We make it capable of dealing with any blade by predicting its weights and biases from the surface mesh of a blade given as input. First, a graph neural network (GNN) extracts a fixed-sized vector representation of the blade, θ'. This representation can be thought of as a pseudo design vector. Afterwards, the hyper-net generates the weights and biases, ϕ, of the backbone-net. Finally, the backbone-net yields the 3D flow solution that is compared to the ground truth.

the weights ϕ of the backbone-net. The hyper-net is a small residual network:

$$\boldsymbol{z}_0^{(h)} = \boldsymbol{W}_{in}^{(h)} \boldsymbol{\theta} \tag{2}$$

$$\boldsymbol{z}_k^{(h)} = \boldsymbol{z}_{k-1}^{(h)} + \boldsymbol{W}_{2,k}^{(h)} \sigma(\boldsymbol{W}_{1,k}^{(h)} \sigma(\boldsymbol{z}_{k-1}^{(h)})), 1 \leq k \leq K' \tag{3}$$

$$\phi = \boldsymbol{W}_{out}^{(h)} \boldsymbol{z}_{K'}^{(h)} \tag{4}$$

Notice that we insert a GELU non-linearity right at the start of the residual branch. Experimentally the hyper-net is shallow with $K' = 1$ residual blocks; in each block, the hidden dimension of the main branch is 48 while it is 96 in the residual branch. In the residual branch dropout is enabled to provide additional regularization in the small data regime.

Training a hypernetwork is notoriously difficult (Lorraine & Duvenaud, 2018; Ukai et al., 2018). We follow (Ortiz et al., 2023) and treat the hyper-net predictions as additive changes to the backbone-net. This makes training stable.

Hypernetworks are also prohibitively expensive, in both compute and memory. To avoid an explosion of the number of parameters, only the biases of the backbone-net are predicted by the hyper-net. This is similar to applying shift modulations (Naour et al., 2023). However, by doing so, the performance of our model degrades markedly. To counteract this loss of performance, the hyper-net also predicts the first and the last weight matrices of the backbone-net, $\boldsymbol{W}_{in}^{(b)}$ and $\boldsymbol{W}_{out}^{(b)}$ respectively. The hyper-net is thus small, with 260k parameters in our experiments.

4. Training

We train the backbone-net solo for 300 epochs and the hyper-net for 400 epochs using NAdam (Dozat, 2016) and a cosine learning rate scheduler. The initial learning rate is 0.01 for the backbone-net solo and 0.001 for the hyper-net.

The backbone-net is small and data is plenty, so it is not regularized. But the hyper-net is trained with limited data, so it is regularized with dropout. The dropout rate is a hyperparameter that is found by grid search using Optuna (Akiba et al., 2019).

We train the backbone-net solo with mini-batches. With large batch sizes generalization is poor, in accordance with (Keskar et al., 2016) that showed that large batch sizes are associated to a degradation in model quality, whereas with small batch sizes the training run-time takes longer. We set the trade-off by a batch size of 0.5k. Training takes circa 20 minutes on a single core of an Intel Xeon Platinum CPU for a dataset with around 0.4Mio points. We also train the hyper-net with mini-batches: each batch has 20k points of a single configuration associated to a given blade geometry. Notice that we consider all points of a configuration at each epoch. The training takes less than one day on 32 cores of a CPU for a dataset with circa 130 configurations.

5. Experiments

In our experiments, both qualitative and quantitative evaluations are provided. Visual examination of 3D flow fields is hard, so to avoid cluttered plots we only show slices at constant relative radius r_{S_1}, ranging from 0 (hub) to 1 (tip). $r_{S_1} = 0.5$ is referred to as midspan. See Figure 6 for an illustration.

5.1. Datasets

We consider four different blades, a stator and a rotor from a low-pressure subsonic turbine, `turb-stator` and `turb-rotor`, and a stator and a rotor from a low-pressure transonic compressor, `comp-stator` and `comp-rotor`. Those blades feature complex geometries: Both turbine blades have large fillets, the compressor stator has half-gaps (small gaps in the rear part at the hub and tip, see Figure 6) and the compressor rotor has a tip clearance (small gap between blade and casing). For each of these four original blades indexed by b, we sample its geometry vector $\boldsymbol{\theta}_b$ randomly M_b times by varying radially among other things the leading edge blade angle, trailing edge blade angle and stagger angle, so as to construct a dataset of configurations $\mathcal{D}_b^{h-net} = \{(\mathcal{G}_{b,m}, \mathcal{D}_{b,m}^{b-net})\}_{m=1}^{M_b}$ for training the hyper-net. As a result, all datasets $\mathcal{D}_{b,m}^{b-net}$ of a given series b for $m = 1, \cdots, M_b$ have the same number of points, approximately 0.4Mio. Recall that $\mathcal{D}_{b,m}^{b-net}$ is the dataset from a simulation. The boundary conditions at inlet (total pressure, total temperature and velocity directions) and outlet (back pressure), along with the rotational speed, are hold constant for each series of blades.

All 3D steady compressible flow solutions $\mathcal{D}_{b,m}^{b-net}$ were generated using the Navier-Stokes solver TRACE[3] (Becker et al., 2010) with the Wilcox $k-\omega$ turbulence model (Wilcox, 1988). For turbine blades, the $\gamma - Re_\theta$ transition model (Langtry & Menter, 2009) was enabled.

Our model outputs $\hat{y} = [\rho, p, V_x, V_y, V_z]$. To those five primitive variables, it is possible to add quantities related to the turbulence model like the (log of the) turbulent viscosity, but ultimately we are interest in quantities like massflow, efficiency and flow turning, whose derivation relies only on the five primitive variables our model predicts.

All input and output features are linearly scaled into $[-1, 1]$ based on the extreme values computed from samples in the training set only.

5.2. Data Augmentations

The stators and rotors are extracted from axial flow turbomachines. So the flow solution is equivariant under a translation along the x-axis. Furthermore, the flow solution of a configuration is the same if the configuration is rotated about the x-axis, under the condition that the velocity components V_y and V_z are rotated accordingly. A blade has no canonical position in space we could set it in. Hence our model would fail to generalize to configurations in unknown positions and experience a performance drop at test time. To overcome this difficulty, we let our model learn those properties using data augmentations (as is usually done in image classification (Alex et al.)). We re-center each configuration at $x = 0$ (by translation) and $y = 0$ (by rotation) — thereby defining approximately a reference position – and at training time we apply a small random rotation (between -5° and +5° in all our experiments) about the x-axis to the configurations. Since the extension of each configuration (of a given series) in the x direction is the same, it is not necessary to apply a random translation along the x-axis. At test time, we simply put the configurations in the reference position. This data augmentation mitigates overfitting and makes the hyper-net robust against a departure from the reference position at inference.

5.3. Evaluation Metrics

In regression the choice of a differentiable cost function usually defaults to the mean squared error (MSE) but we use the mean absolute error (MAE) as evaluation metric since MAE produces results that are visually better, as is the case in many image application tasks (Zhao et al., 2016; Isola et al., 2017). Figure 2 illustrates the advantage of MAE over MSE in predicting flow solutions.

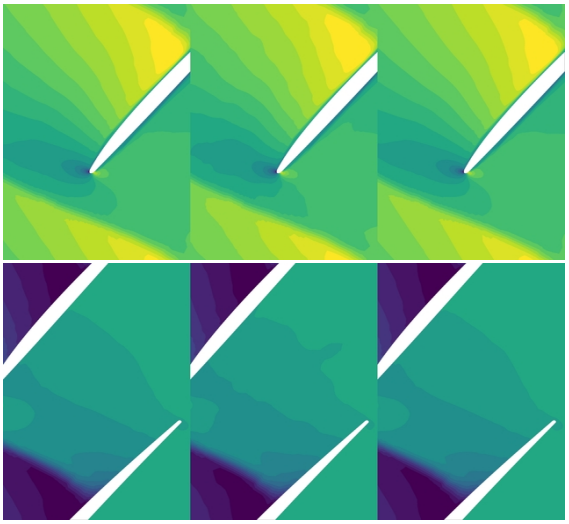

Figure 2. Left: Predictions with MAE loss; middle: predictions with MSE loss; right: ground-truth, for the backbone-net trained solo on `comp-rotor`. We show the axial velocity component V_x at the leading edge (top row) and the pressure p at the trailing edge (bottom row), both at $r_{S_1} = 0.8$.

[3]The non-commercial TRACE solver is developed jointly by DLR and MTU.

5.4. Backbone-Net Solo

We check the capability of the backbone-net by training it solo on various configurations. For each configuration, we choose a 80/20 train/test split. The training set is further divided into training and validation sets (90/10 split). A summary of our results for the compressor and turbine configurations is presented in Tables 1 and 2, respectively.

Our backbone-net is rather small, with roughly 80k parameters. It would be straightforward to scale up this model to get much better results than the ones we will present in this section without severely overfitting the training dataset because a training dataset has circa 0.4Mio points. However, we must keep in mind that all the weights of the backbone-net will be later predicted by the hyper-net and so we must refrain from building a too large backbone-net, otherwise the hyper-net would require too many parameters. Though small, the backbone-net is expressive enough. Figure 3 clearly illustrates its outstanding ability in accurately predicting the flow fields of the primitive variables. The flow is transonic close to the hub and supersonic close to the tip, exhibiting a strong shock wave. Differences against the ground-truth are barely noticeable visually. Figure 4 shows the ability of the backbone-net in rendering the complex flow solution close to leading and trailing edges.

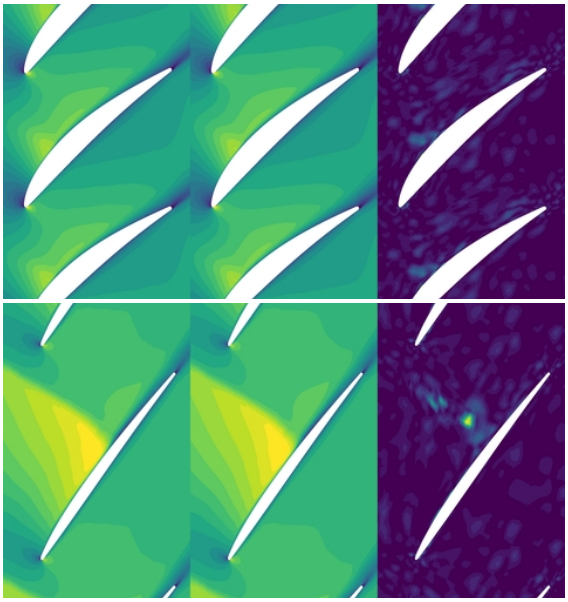

Figure 3. Predictions of the axial velocity component V_x for the configuration `comp-rotor` using the backbone trained solo (left) versus ground-truth (middle); absolute differences on the right. Top row: Near hub ($r_{S_1} = 0.1$). Bottom row: Near tip ($r_{S_1} = 0.8$).

To verify that the interpolation to unseen coordinates is smooth, we conduct the following experiment: we select only 1% and 5% of the samples of the dataset $\mathcal{D}^{b-net}_{\text{comp-rotor}}$ for training. In Figure 5, we see that the flow solution is nevertheless not erratic and flow characteristics are well reproduced. As expected, the quality is not as good as in Figure 3 (bottom row). In this experiment, we clearly have too few data points. Our model was implicitly biased to smooth solutions by keeping the batch size the same as in the experiment with all data points. With only 5% of the samples, results are already very good.

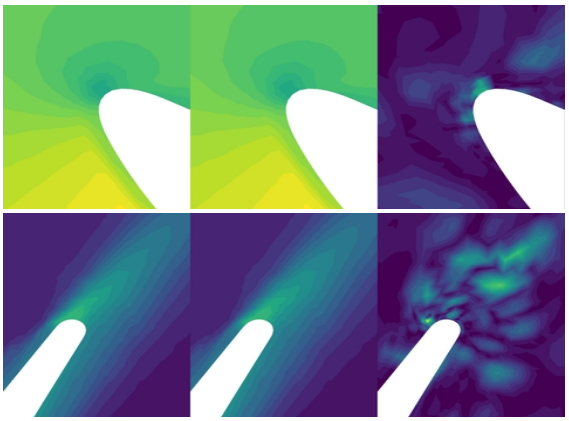

Figure 4. Predictions of the circumferential velocity component V_θ for the configuration `turb-rotor` using the backbone trained solo (left) versus ground-truth (middle) at midspan; absolute differences on the right. Top row: at leading edge; bottom row: at trailing edge.

Table 1. Summary of backbone-net solo training on the compressor datasets. A dataset is a simulation.

DATASET	$\mathcal{D}^{b-net}_{\text{COMP-STATOR}}$	$\mathcal{D}^{b-net}_{\text{COMP-ROTOR}}$
TRAINING LOSS	4.7E-4	7.4E-4
VALIDATION LOSS	5.3E-4	8.6E-4
TEST LOSS	5.2E-4	8.5E-4
#TRAINING SAMPLES (K)	416	307
#TEST SAMPLES (K)	104	77
#EPOCHS	300	300
TRAINING TIME (MINUTES)	23	19

Table 2. Summary of backbone-net solo training on the turbine datasets. A dataset is a simulation.

DATASET	$\mathcal{D}^{b-net}_{\text{TURB-STATOR}}$	$\mathcal{D}^{b-net}_{\text{TURB-ROTOR}}$
TRAINING LOSS	5.8E-4	6.7E-4
VALIDATION LOSS	6.4E-4	7.4E-4
TEST LOSS	6.7E-4	7.4E-4
#TRAINING SAMPLES (K)	346	346
#TEST SAMPLES (K)	86	86
#EPOCHS	300	300
TRAINING TIME (MINUTES)	19	19

5.5. Hyper-Net

The number of parameters of the hyper-net is generally huge, proportional to the number of parameters of the backbone-

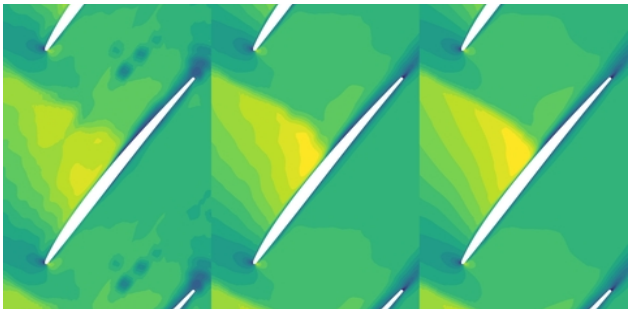

Figure 5. Predictions using the backbone trained solo with only 1% (left), 5% (middle) of the dataset versus ground-truth (right) of variable V_x for `comp-rotor` near tip ($r_{S_1} = 0.8$).

net, the constant of proportionality being the hidden dimension of the hyper-net. This explains why we took great care to have a small but nevertheless accurate backbone-net. By applying shift modulation (see Section 3.2), the number of trainable parameters was reduced to roughly 0.5Mio parameters.

To better assess the generalization performance of the hyper-net in the low data regime, we consider circa 130 training samples. The training set is divided into training and validation sets (85/15 split).

Following (Lee et al., 2019), we consider experimentally two sources of uncertainty: 1/ the fold for training, and 2/ the initialization of the parameters. In Table 3 and 4, we report the mean and standard deviation of the losses calculated over 20 trials using different random seeds and training folds (generated by shuffling the dataset prior to the split). Notice that the hyperparameters for regularizing the model were optimized for a single random split prior to the trials and then frozen. We observe that the model performance is the same across all folds, so we are confident that the observations made henceforth for a random split will hold for any split, in particular the agreement between true and predicted quantities of interest is remarkable, with correlation coefficients close to 1, Figure 16, 18, 20 and 22.

To illustrate that model predictions are at least visually satisfactory, we pick up a test sample with a loss close to the average test loss from the dataset $\mathcal{D}^{h-net}_{\text{comp-stator}}$ and plot the pressure field, Figure 6. From that perspective, predictions are excellent. We proceed similarly with the dataset $\mathcal{D}^{h-net}_{\text{turb-stator}}$ and plot the pressure profiles since they are of uttermost importance for turbomachine designers, Figure 7.

Now we specifically focus on the dataset $\mathcal{D}^{h-net}_{\text{comp-rotor}}$ randomly build around the low-pressure compressor rotor blade `comp-rotor`: it is the most challenging from all datasets we tested our method on since the flow of many configurations in this dataset is highly inhomogeneous radially with a large separation induced by a strong shock wave from

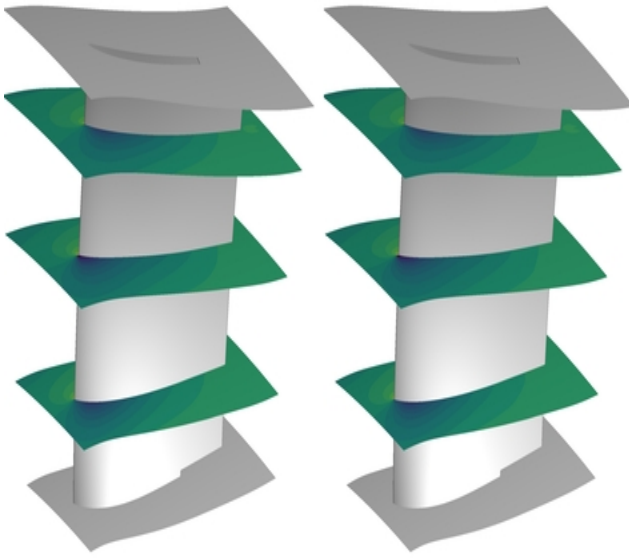

Figure 6. Predictions (left) and ground-truth (right) of the pressure for sample 301 of the dataset $\mathcal{D}^{h-net}_{\text{comp-stator}}$ at $r_{S_1} = 0.2$ (close to the hub), $r_{S_1} = 0.5$ (midspan) and $r_{S_1} = 0.8$ (close to the tip). This test configuration has a loss close to the average test loss.

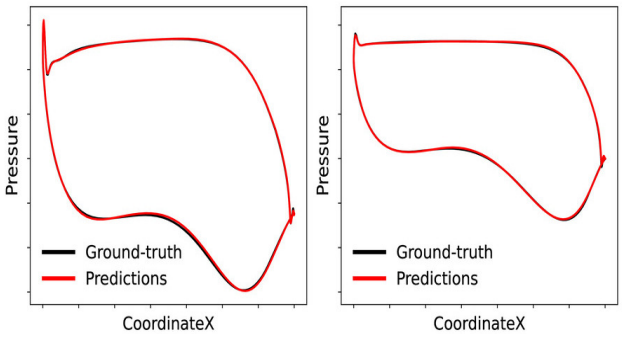

Figure 7. Pressure profiles at $r_{S_1} = 0.2$ (left) and $r_{S_1} = 0.8$ (right) for sample 153 of the dataset $\mathcal{D}^{h-net}_{\text{turb-stator}}$. This test configuration has a loss close to the the average test loss.

midspan outwards. Those are mostly the configurations on the right in Figure 8 because their aerodynamic loss is significantly high. What makes `comp-rotor`, and so all configurations in this dataset, even more particular is that the blade is modelled with a tip clearance; the blade rotates while the casing is non-rotating. This raises severe difficulty in accurately predicting the flow solution.

From the losses reported in Table 3 for both compressor datasets, it is clear that our model overfits the training dataset despite the use of dropout to mitigate overfitting. Notice that our model has a small memory footprint: It does not exceed 12GB even though all samples are read in at the start to avoid IO intensive operations during training. To illus-

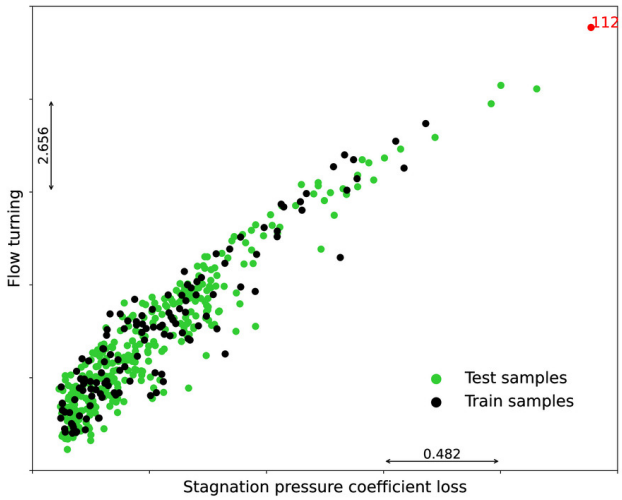

Figure 8. Aerodynamic characteristics of the samples randomly generated from the blade `comp-rotor`.

trate the predictive capability of our model, we consider test sample 112 that clearly lies outside the support of training samples in the dataset, see Figure 8. The geometry of this sample is unconventional and definitively not interesting for designers because of the high aerodynamics losses it generates, but to stress test our model it is the perfect candidate. Its geometry is compared to a more conventional blade in Figure 9. Intuitively, we expect that our model will have trouble with new samples that lie outside the convex hull of the training dataset as sample 112. But the plot is deceptive since it is low-dimensional: It cannot fully convey a sense about how the convex hull of the dataset looks like.

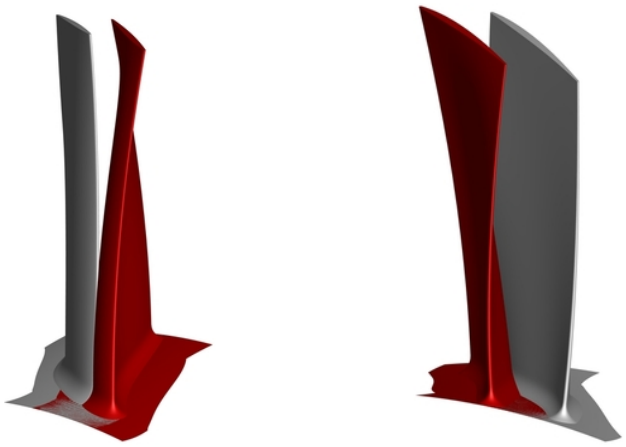

Figure 9. Geometry of sample 112 in red and sample 465 in grey.

Figures 11-14 show the excellent performance of the hyper-net in predicting challenging 3D flow solutions of blades with complex geometry. We can nevertheless observe some minor discrepancies, in particular the predicted show wave from midspan outwards seems to be slightly more smeared out, not as sharply captured as in the CFD simulation. Figure 10 depicts the pressure distribution around the blade. Close to the hub, the agreement between ground-truth and predictions is particularly good. Close to the tip, our model has trouble with the large flow separation on the suction side and with the attached flow on the pressure side. A domain where such a proxy can shine is blade shape optimization because it is cheap and accurate. However, until now we only provided a visual assessment of the model prediction ability. To be useful as a low-fidelity model in a multi-fidelity optimization, a model has to predict quantities of interest also accurately, by far a much harder task. This ability is usually demonstrated with correlation plots: fidelity models should be well correlated, with a correlation coefficient close to 1 (Toal, 2015). Figure 18 reveals that the hyper-net can almost perfectly predict massflow and flow turning. Predicting aerodynamic losses is usually more difficult (Bonnet et al., 2022; Kalaydjian et al., 2023) since it is a highly non-linear function of the predictions of the primitive variables, but the hyper-net tops here with a Pearson correlation coefficient as high as 0.995 on the test set.

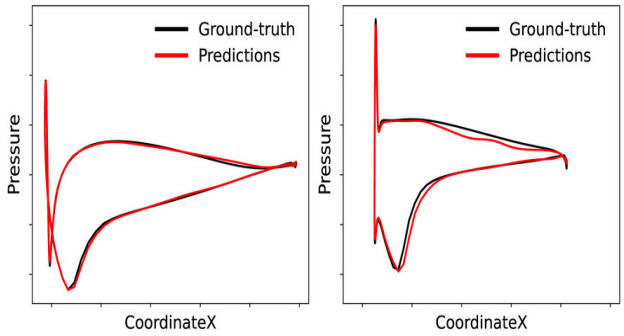

Figure 10. Pressure profiles at $r_{S_1} = 0.2$ (left) and $r_{S_1} = 0.8$ (right) for sample 112 of the dataset $\mathcal{D}^{h-net}_{\text{comp-rotor}}$.

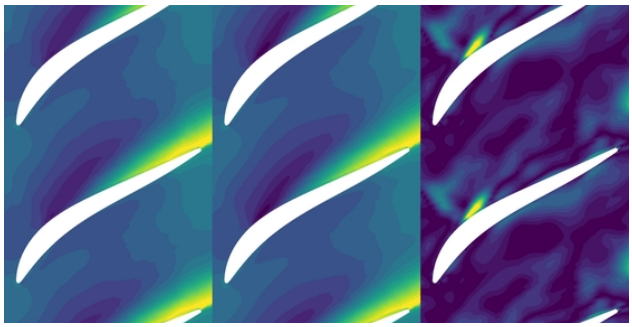

Figure 11. Prediction (left), ground-truth (middle) and absolute differences (right) for the radial velocity component V_r for sample 112 at $r_{S_1} = 0.2$ (near hub).

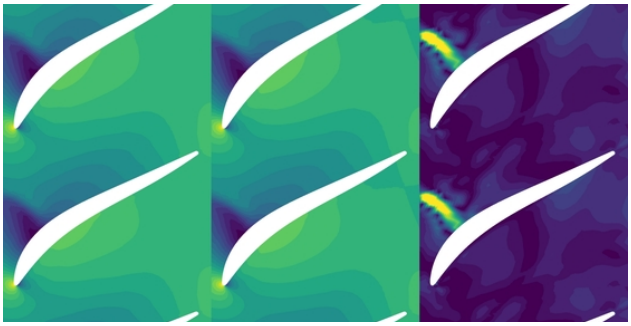

Figure 12. Prediction (left), ground-truth (middle) and absolute differences (right) for the pressure p for sample 112 at $r_{S_1} = 0.2$ (near hub).

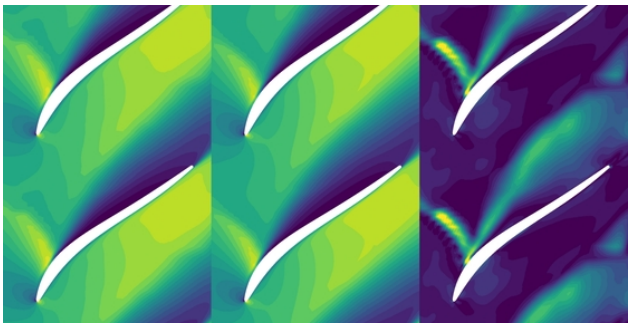

Figure 13. Prediction (left), ground-truth (middle) and absolute differences (right) for the axial velocity V_x for sample 112 at $r_{S_1} = 0.5$ (midpsan).

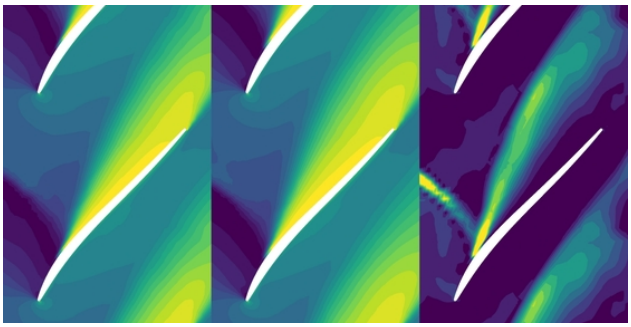

Figure 14. Prediction (left), ground-truth (middle) and absolute differences (right) for the tangential velocity V_θ for sample 112 at $r_{S_1} = 0.8$ (near tip).

6. Conclusion

In this work, we derived a fast and accurate proxy to CFD solvers based on an implicit neural representation. Our model comprises a backbone-net that establishes the mapping of a point in the computational domain to its flow features. It is instantiated as a small coordinate-based MLP. We conditioned the weights of the backbone-net onto the blade geometry using a hyper-net. It is instantiated as a small residual network. By doing so, we could directly predict the full 3D flow solution for any unseen blade geometry. We showed that our model can make accurate predictions of full 3D flows of compressor blades. Our model succeeded even though the flow exhibits large separations induced by strong shock waves and is inhomogeneous from hub to tip. Furthermore, for quantities-of-interest like massflow, flow turning and efficiency aggregated from the model predictions, our model has also very good predictive capabilities. As a result, our model is likely to be useful in future as a low-fidelity model in a multi-fidelity optimization of blade shapes.

7. Acknowledgments

The authors would like to thank MTU Aero Engines for the permission to publish this paper. The research work associated with this publication has been supported by the German Federal Ministry for Economic Affairs and Climate Action under grant number 20X1909E. The funding of the work through the 1st call of the Federal Aviation Research Program VI (LuFo VI-1), grant project title 'DIGIfly', is gratefully acknowledged. The authors are responsible for the content of this publication.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Alex, K., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional networks. In *volume-1; pages-1097–1105; NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing System*.

Aulich, M., Küppers, F., Schmitz, A., and Voß, C. Surrogate estimations of complete flow fields of fan stage designs via deep neural networks. In *Turbo Expo: Power for Land, Sea, and Air*, volume 58585, pp. V02DT46A013. American Society of Mechanical Engineers, 2019.

Baque, P., Remelli, E., Fleuret, F., and Fua, P. Geodesic convolutional shape optimization. In *International Conference on Machine Learning*, pp. 472–481. PMLR, 2018.

Becker, K., Heitkamp, K., and Kügeler, E. Recent progress in a hybrid-grid cfd solver for turbomachinery flows. In *Proceedings fifth European conference on computational fluid dynamics ECCOMAS CFD*, volume 2010, 2010.

Bonnet, F., Mazari, J. A., Cinella, P., and Gallinari, P. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier-stokes solutions. In *Neural Information Processing Systems (NeuRIPS 2022)*, 2022.

Chen, L.-W. and Thuerey, N. Towards high-accuracy deep learning inference of compressible turbulent flows over aerofoils. *arXiv preprint arXiv:2109.02183*, 2021.

Chen, L.-W., Cakal, B. A., Hu, X., and Thuerey, N. Numerical investigation of minimum drag profiles in laminar flow using deep learning surrogates. *Journal of Fluid Mechanics*, 919, 2021.

Chen, Z. and Zhang, H. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019.

Chuang, P.-Y. and Barba, L. A. Experience report of physics-informed neural networks in fluid simulations: pitfalls and frustration. *arXiv preprint arXiv:2205.14249*, 2022.

Ding, M., Kong, K., Li, J., Zhu, C., Dickerson, J., Huang, F., and Goldstein, T. Vq-gnn: A universal framework to scale up graph neural networks using vector quantization. *Advances in Neural Information Processing Systems*, 34, 2021.

Dozat, T. Incorporating nesterov momentum into adam. 2016.

Du, M., Chen, Y., and Zhang, D. Autoke: An automatic knowledge embedding framework for scientific machine learning. *arXiv preprint arXiv:2205.05390*, 2022.

Galanti, T. and Wolf, L. On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*, 33:10409–10419, 2020.

Gao, H., Sun, L., and Wang, J.-X. Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Physics of Fluids*, 33(7):073603, 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Grattarola, D., Zambon, D., Bianchi, F. M., and Alippi, C. Understanding pooling in graph neural networks. *arXiv preprint arXiv:2110.05292*, 2021.

Guo, X., Li, W., and Iorio, F. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 481–490, 2016.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Huang, L. and Hoefler, T. Compressing multidimensional weather and climate data into neural networks. *arXiv preprint arXiv:2210.12538*, 2022.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., and Carreira, J. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, pp. 4651–4664. PMLR, 2021.

Kalaydjian, A., Balykov, A., Semiz, A., and Chan-Hon-Tong, A. Packed-ensemble surrogate models for fluid flow estimation arround airfoil geometries. *arXiv preprint arXiv:2312.13403*, 2023.

Kamenik, J., Voutchkov, I., Toal, D. J., Keane, A. J., Högner, L., Meyer, M., and Bates, R. Robust turbine blade optimization in the face of real geometric variations. *Journal of Propulsion and Power*, 34(6):1479–1493, 2018.

Kang, D., Dhar, D., and Chan, A. Incorporating side information by adaptive convolution. *Advances in Neural Information Processing Systems*, 30, 2017.

Kashefi, A., Rempe, D., and Guibas, L. J. A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. *Physics of Fluids*, 33(2):027104, 2021.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Klicpera, J., Qian, C., and Günnemann, S. Locality-based mini batching for graph neural networks. 2021.

Klocek, S., Maziarka, Ł., Wołczyk, M., Tabor, J., Nowak, J., and Śmieja, M. Hypernetwork functional image representation. In *International Conference on Artificial Neural Networks*, pp. 496–510. Springer, 2019.

Knyazev, B., Drozdzal, M., Taylor, G. W., and Romero Soriano, A. Parameter prediction for unseen deep architectures. *Advances in Neural Information Processing Systems*, 34, 2021.

Langtry, R. B. and Menter, F. R. Correlation-based transition modeling for unstructured parallelized computational fluid dynamics codes. *AIAA journal*, 47(12):2894–2906, 2009.

Lee, J., Lee, I., and Kang, J. Self-attention graph pooling. In *International conference on machine learning*, pp. 3734–3743. PMLR, 2019.

Littwin, G. and Wolf, L. Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1824–1833, 2019.

Lorraine, J. and Duvenaud, D. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*, 2018.

Makkuva, A., Oh, S., Kannan, S., and Viswanath, P. Learning in gated neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 3338–3348. PMLR, 2020.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4460–4470, 2019.

Meyer, L., Pottier, L., Ribes, A., and Raffin, B. Deep surrogate for direct time fluid dynamics. *arXiv preprint arXiv:2112.10296*, 2021.

Meyer, M., Becker, B., Poloni, C., and Gambitta, M. Aerodynamic design of a compressor rotor using an optimization-under-uncertainty approach. In *AIAA Scitech 2019 Forum*, pp. 1211, 2019.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pp. 405–421. Springer, 2020.

Naour, E. L., Serrano, L., Migus, L., Yin, Y., Agoua, G., Baskiotis, N., Guigue, V., et al. Time series continuous modeling for imputation and forecasting with implicit neural representations. *arXiv preprint arXiv:2306.05880*, 2023.

Obiols-Sales, O., Vishnu, A., Malaya, N., and Chandramowliswharan, A. Cfdnet: A deep learning-based accelerator for fluid simulations. In *Proceedings of the 34th ACM international conference on supercomputing*, pp. 1–12, 2020.

Ortiz, J. J. G., Guttag, J., and Dalca, A. V. Magnitude invariant parametrizations improve hypernetwork learning. In *The Twelfth International Conference on Learning Representations*, 2023.

Pan, S., Brunton, S. L., and Kutz, J. N. Neural implicit flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data. *arXiv preprint arXiv:2204.03216*, 2022.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Popov, G. M., Baturin, O., Goriachkin, E., Kolmakova, D. A., Volkov, A., and Egorov, I. Optimization algorithm for axial multistage compressor workflow. In *AIAA Propulsion and Energy 2020 Forum*, pp. 3683, 2020.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

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

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Skorokhodov, I., Ignatyev, S., and Elhoseiny, M. Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10753–10764, 2021.

Spurek, P., Winczowski, S., Tabor, J., Zamorski, M., Zięba, M., and Trzciński, T. Hypernetwork approach to generating point clouds. *arXiv preprint arXiv:2003.00802*, 2020.

Stanley, K. O. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

Thuerey, N., Weißenow, K., Prantl, L., and Hu, X. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

Toal, D. J. Some considerations regarding the use of multi-fidelity kriging in the construction of surrogate models. *Structural and Multidisciplinary Optimization*, 51(6): 1223–1245, 2015.

Ukai, K., Matsubara, T., and Uehara, K. Hypernetwork-based implicit posterior estimation and model averaging of cnn. In *Asian Conference on Machine Learning*, pp. 176–191. PMLR, 2018.

Vinuesa, R. and Brunton, S. L. The potential of machine learning to enhance computational fluid dynamics. *arXiv preprint arXiv:2110.02085*, 2021.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

White, C., Ushizima, D., and Farhat, C. Fast neural network predictions from constrained aerodynamics datasets. In *AIAA Scitech 2020 Forum*, pp. 0364, 2020.

Wilcox, D. C. Reassessment of the scale-determining equation for advanced turbulence models. *AIAA journal*, 26 (11):1299–1310, 1988.

Xiao, X., Zhou, Y., Wang, H., and Yang, X. A novel cnn-based poisson solver for fluid simulation. *IEEE transactions on visualization and computer graphics*, 26(3): 1454–1465, 2018.

Zhang, Y., van Rozendaal, T., Brehmer, J., Nagel, M., and Cohen, T. Implicit neural video compression. *arXiv preprint arXiv:2112.11312*, 2021.

Zhao, H., Gallo, O., Frosio, I., and Kautz, J. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57, 2016.

A. Compressor stator and rotor

Table 3. Summary of hyper-net training on the compressor dataset. A dataset is a set of simulations.

DATASET	$\mathcal{D}^{h-net}_{\text{COMP-STATOR}}$	$\mathcal{D}^{h-net}_{\text{COMP-ROTOR}}$
DIM. DESIGN SPACE	13	6
#SURFACE TRIANGLES/K	61	42
#SURFACE NODES/K	31	21
#TRAINING SAMPLES	135	128
#TEST SAMPLES	264	316
#EPOCHS	400	400
TRAINING LOSS	1.9E-3 $\pm$ 5.7E-5	3.3E-3 $\pm$ 7.8E-5
VALIDATION LOSS	3.7E-3 $\pm$ 1.3E-3	4.2E-3 $\pm$ 3.5E-4
TEST LOSS	3.8E-3 $\pm$ 1.5E-3	4.3E-3 $\pm$ 1.3E-4
TRAINING TIME/HOURS	23	18
PEAK MEMORY/GB	14	12

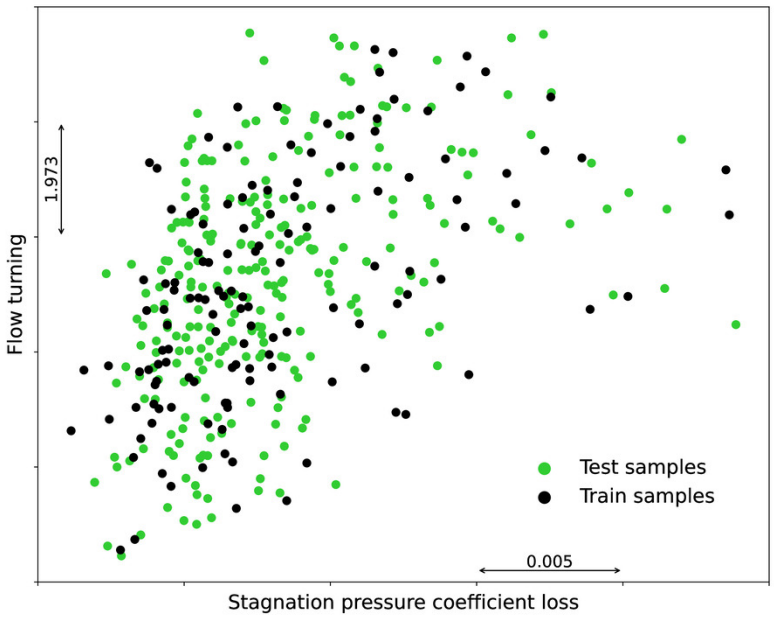

Figure 15. Aerodynamic characteristics of the samples randomly generated from the blade `comp-stator`.

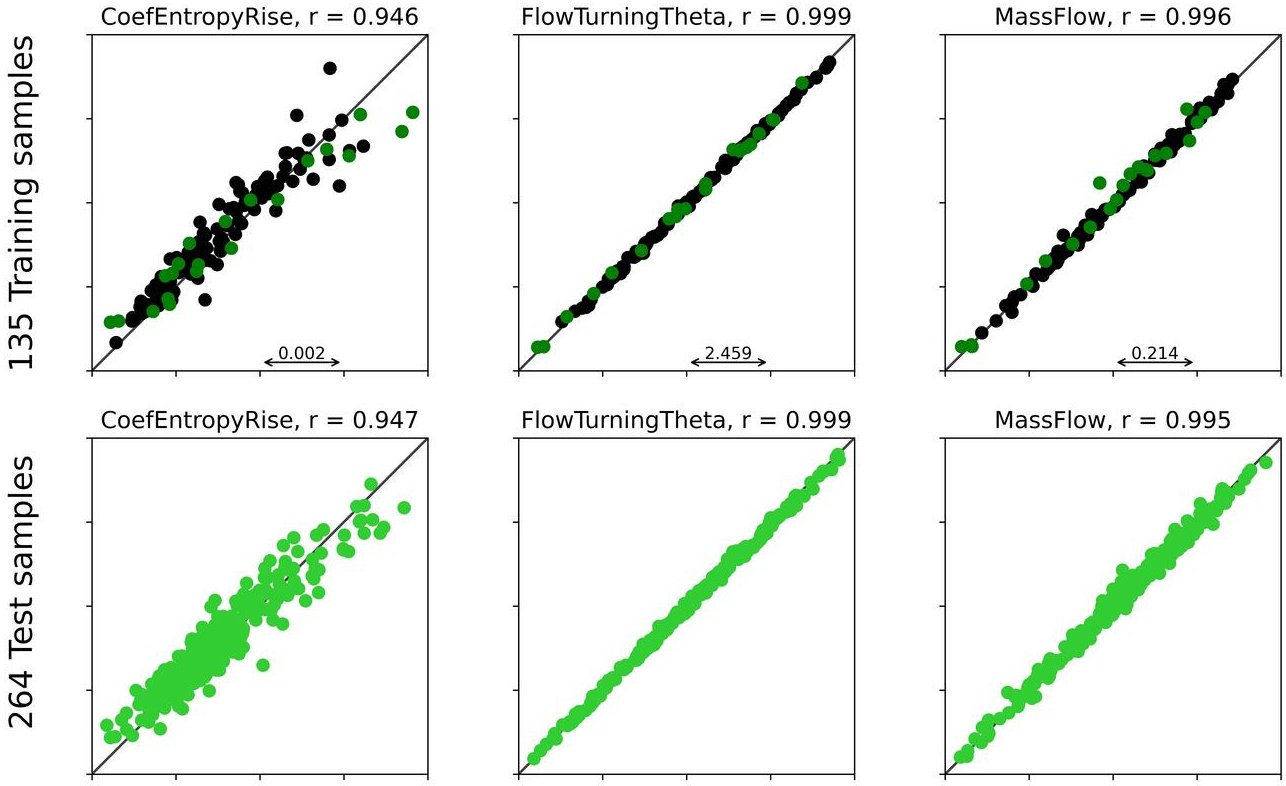

Figure 16. Correlation plots for the samples randomly generated from the blade `comp-stator`. r designates the Pearson product-moment correlation coefficient. In the top row of each figure, training samples are in black whereas validation samples are in green.

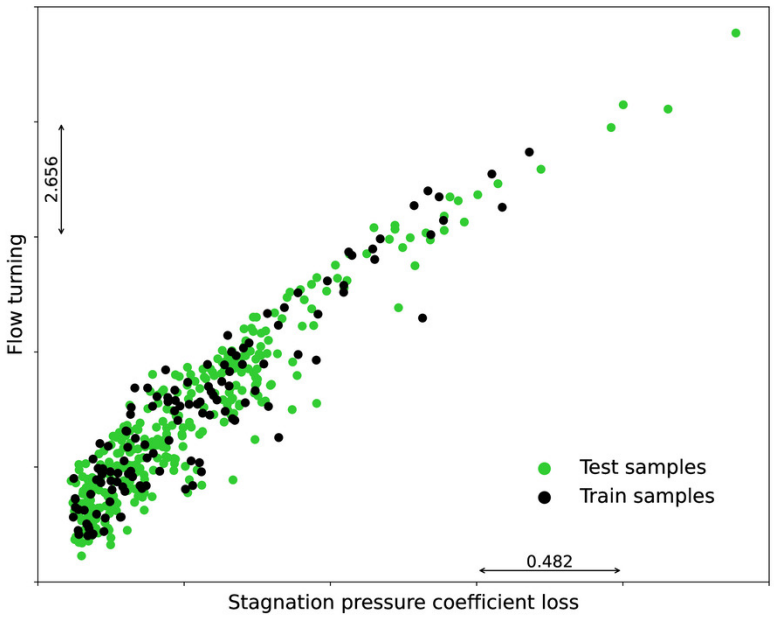

Figure 17. Aerodynamic characteristics of the samples randomly generated from the blade `comp-rotor`.

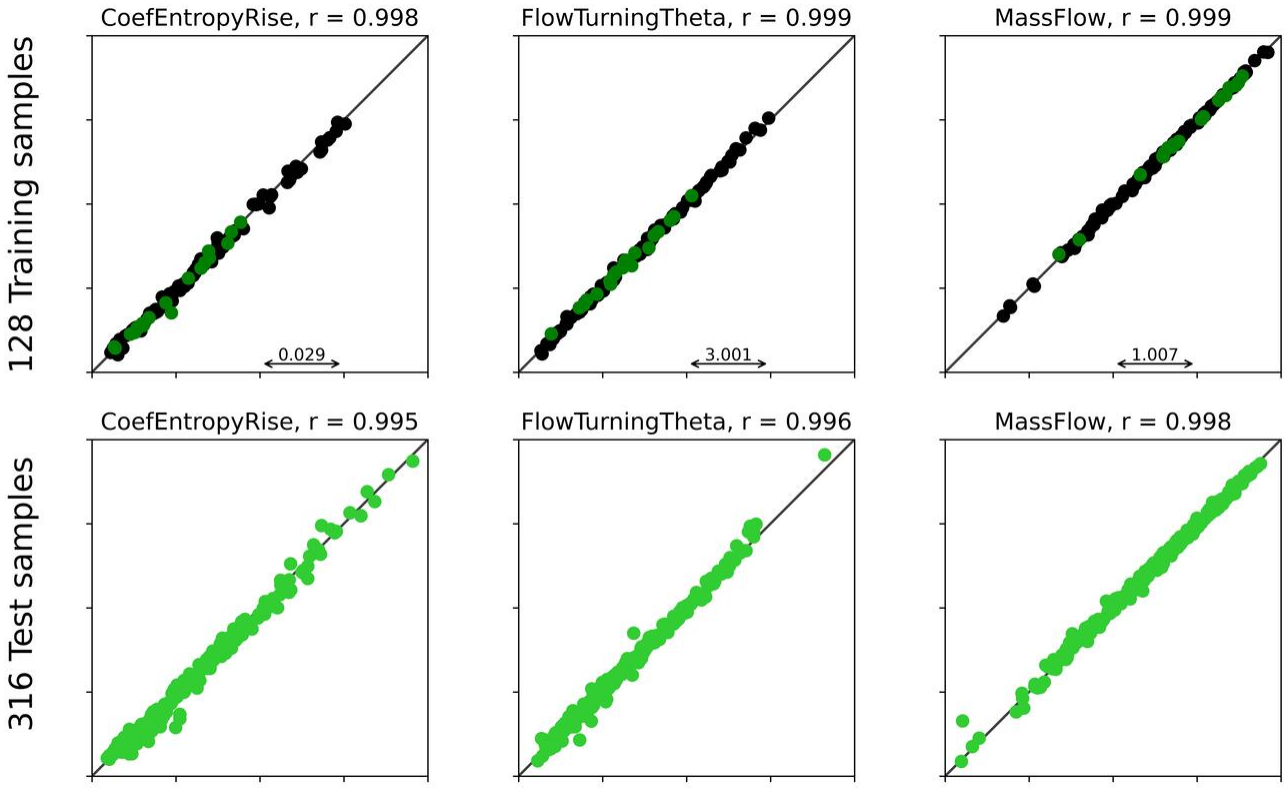

Figure 18. Correlation plots for the samples randomly generated from the blade `comp-rotor`. r designates the Pearson product-moment correlation coefficient. In the top row of each figure, training samples are in black whereas validation samples are in green.

B. Turbine stator and rotor

Table 4. Summary of hyper-net training on the turbine dataset. A dataset is a set of simulations.

DATASET	$\mathcal{D}^{h-net}_{\text{TURB-STATOR}}$	$\mathcal{D}^{h-net}_{\text{TURB-ROTOR}}$
DIM. DESIGN SPACE	17	13
#SURFACE TRIANGLES/K	42	42
#SURFACE NODES/K	21	21
#TRAINING SAMPLES	132	129
#TEST SAMPLES	497	459
#EPOCHS	400	400
TRAINING LOSS	2.4E-3 $\pm$ 3.2E-5	2.7E-3 $\pm$ 3.1E-5
VALIDATION LOSS	3.3E-3 $\pm$ 2.8E-4	3.7E-3 $\pm$ 2.3E-4
TEST LOSS	3.3E-3 $\pm$ 8.4E-5	3.7E-3 $\pm$ 1.3E-4
TRAINING TIME/HOURS	18	18
PEAK MEMORY/GB	18	18

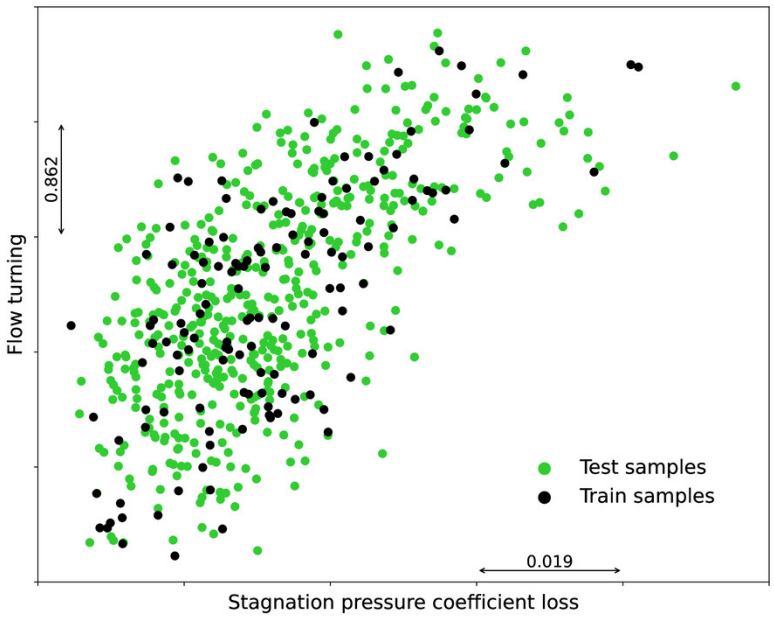

Figure 19. Aerodynamic characteristics of the samples randomly generated from the blade `turb-stator`.

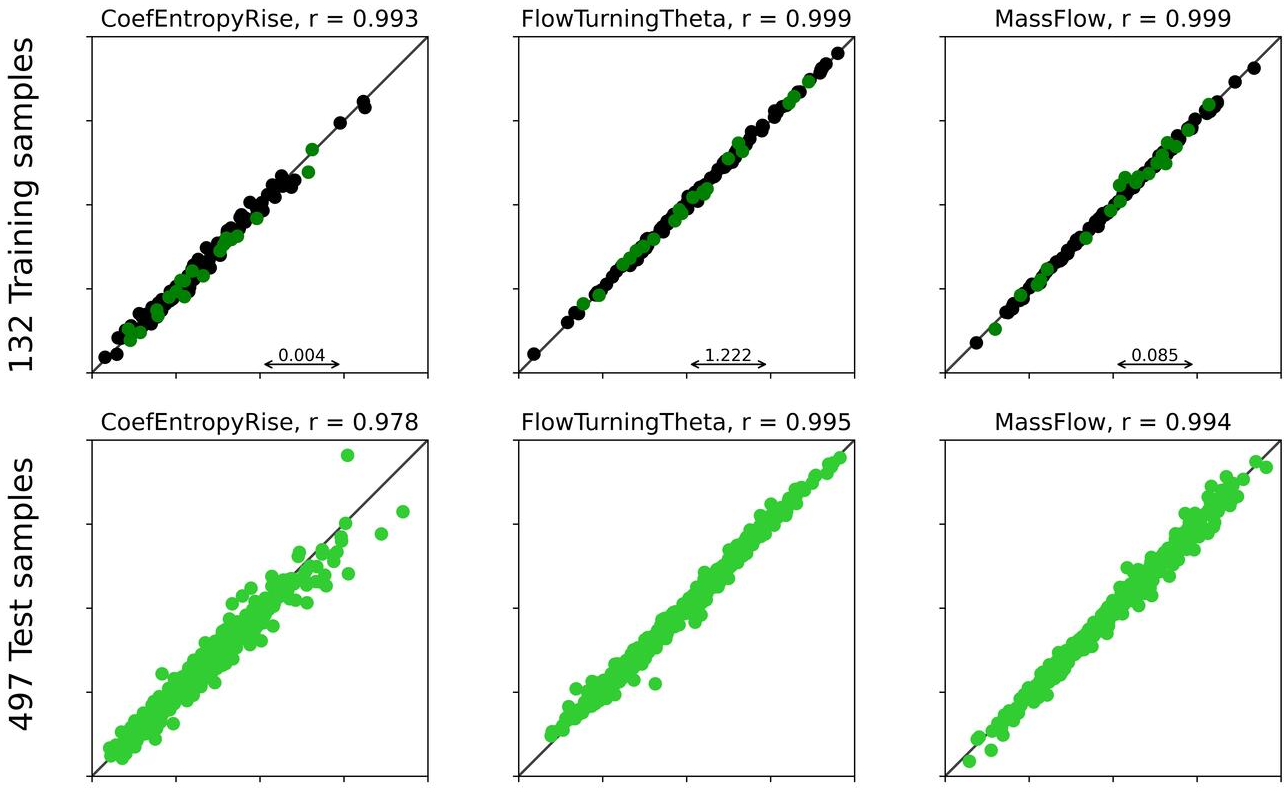

Figure 20. Correlation plots for the samples randomly generated from the blade `turb-stator`. r designates the Pearson product-moment correlation coefficient. In the top row of each figure, training samples are in black whereas validation samples are in green.

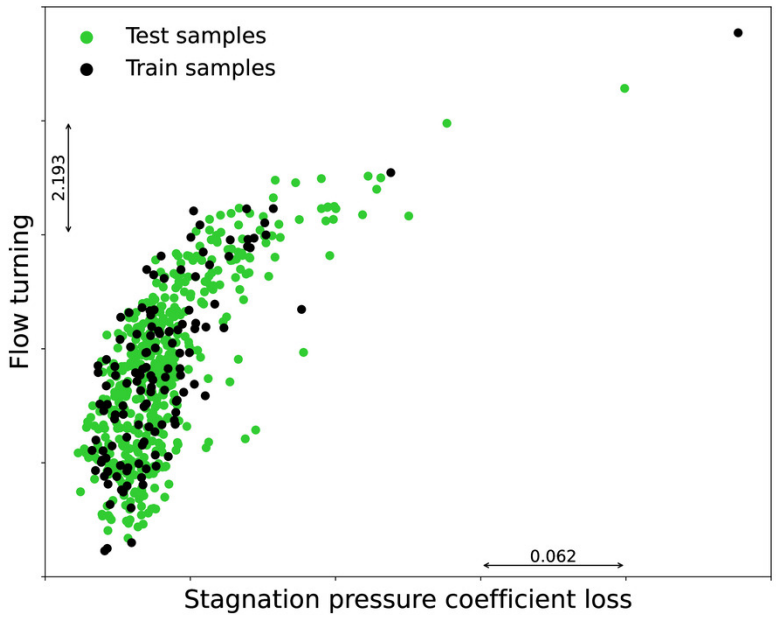

Figure 21. Aerodynamic characteristics of the samples randomly generated from the blade `turb-rotor`.

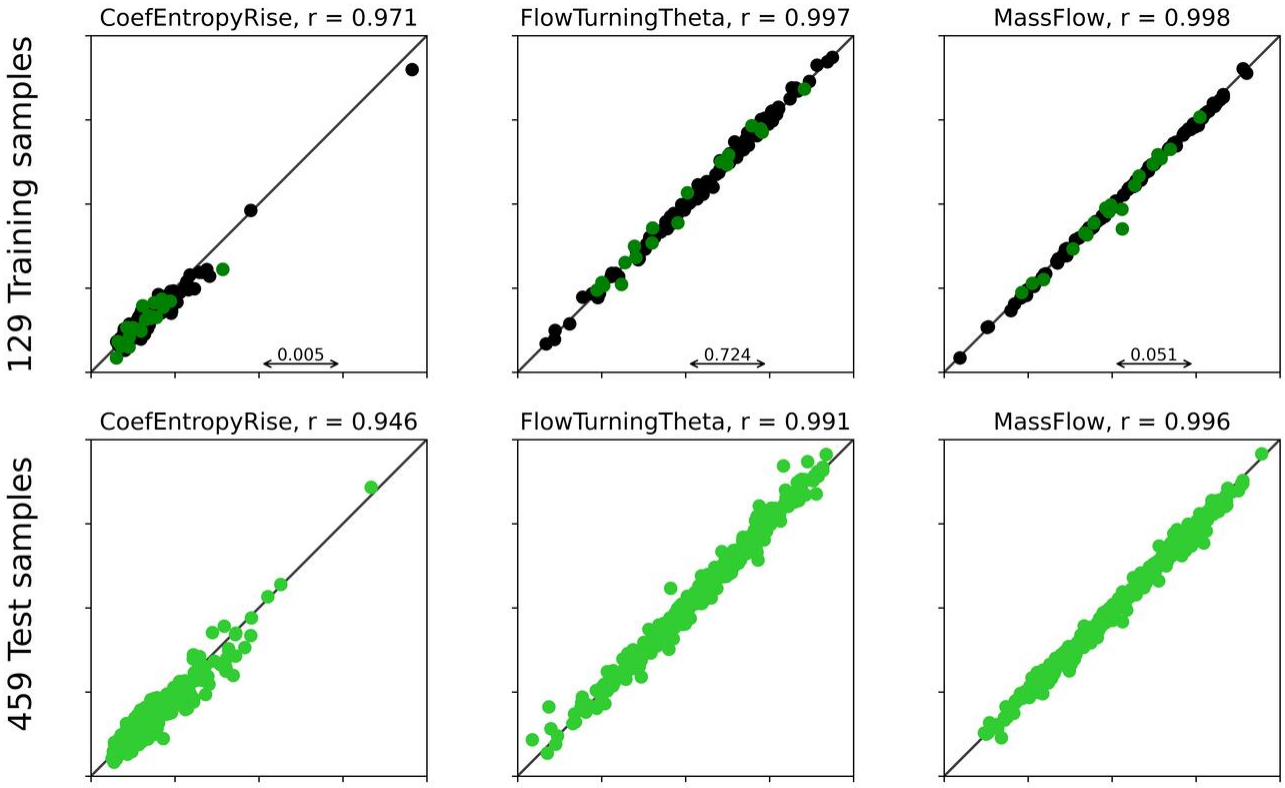

Figure 22. Correlation plots for the samples randomly generated from the blade `turb-rotor`. r designates the Pearson product-moment correlation coefficient. In the top row of each figure, training samples are in black whereas validation samples are in green.

C. Embedding Dimension

The dimension of the embedding for a given dataset has to be set greater than the intrinsic dimension of the dataset that is unknown. However, it is less than or equal to the number of degrees of freedom of the blade parameterization that is known. In our experiments it is usually less than 16 except for the dataset `turb-stator` for which it is 17. So we carried out a study to find out a reasonable minimum embedding dimension as follows: From the blade embeddings, the blades are reconstructed and the quality of the reconstruction is assessed by the Chamfer loss. To generate a point cloud with variable number of points, we sample a vector of dimension 3 from the normal distribution, append it to the blade embedding and push it through a small MLP that outputs the spatial coordinates. Figure 23 shows that an embedding dimension of 12 is sufficient for a good reconstruction.

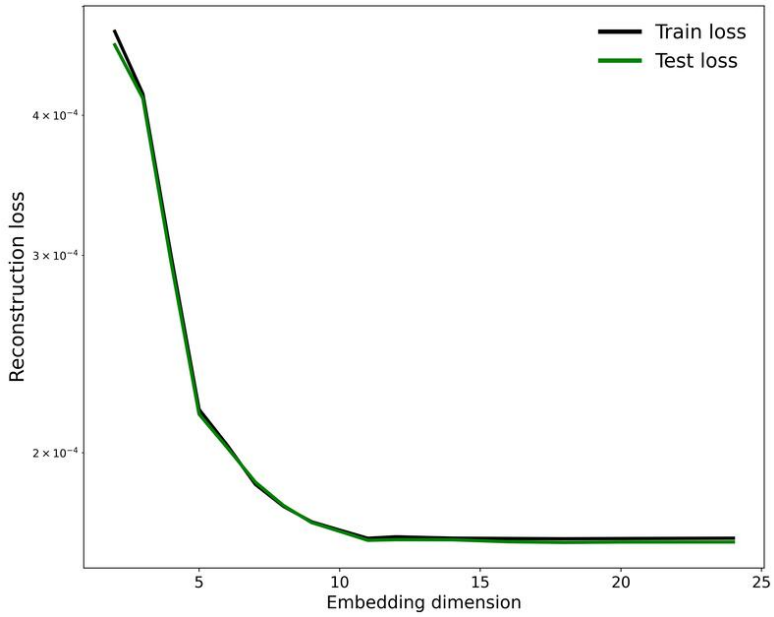

Figure 23. Reconstruction loss for the samples in `turb-stator`. The blade parameterization has 17 degrees of freedom.