

B Geometry sampling

The modal representation is a useful tool to represent the geometry of the airfoil Li et al. [2019]. Taking a set of airfoil shapes, the modal analysis can extract the low dimensional modal representation that is efficient in capture the geometries. A single airfoil can be represented by n surface points characterized by coordinates as (x_i, y_i) , $i = 1, 2, \dots, n$. The y coordinates for the m airfoils can then be assembled into a matrix as

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{21} & \cdots & y_{m1} \\ y_{12} & y_{22} & \cdots & y_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1n} & y_{2n} & \cdots & y_{mn} \end{bmatrix}. \quad (1)$$

Performing singular value decomposition (SVD) in this matrix results in

$$\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \quad (2)$$

where $\mathbf{U} \in \mathbb{R}^{n \times n}$ contains the orthonormal spatial modes (airfoil shape modes) as columns, $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ is a diagonal matrix of singular values, $\mathbf{V} \in \mathbb{R}^{m \times m}$ contains the parametric coefficients in its columns.

To reconstruct or generate a new airfoil shape using the leading r modes ($r \ll n$), we define the reduced basis matrix as

$$\tilde{\mathbf{U}} \in \mathbb{R}^{n \times r}, \quad (3)$$

where $\tilde{\mathbf{U}}$ contains the first r columns of \mathbf{U} . Then, any new airfoil shape $\mathbf{y} \in \mathbb{R}^n$ can be expressed as

$$\mathbf{y} = \tilde{\mathbf{U}}\tilde{\mathbf{y}}, \quad (4)$$

where $\tilde{\mathbf{y}} \in \mathbb{R}^r$ is the vector of modal coefficients. It is noted that the x -values remain fixed and are termed “ x -slices”.

The choice of the modal-coefficients is key in generation of the airfoil shape database. Thus, instead of representing each airfoil with the (x, y) coordinate values of its profile, we use the reduced few modal coefficients that were used to generate that airfoil in the input data to the neural network. For more details on choice of modal coefficients and how the sampling is done exactly for large number of airfoils, we direct the readers to Li et al. [2019, 2020]. In this manner, the 4,800 NLF airfoils and 30,000 FT airfoils were sampled. The code to sampling these airfoils and list of modal coefficients used for the generation of the dataset are detailed in the GitHub repository publicly available in the CC-BY-SA license⁴.

C Numerical method

In this section, we discuss the numerical methods used in this paper. In the first subsection, we discuss the governing equations for the fully-turbulent and transition models used in the dataset generation for the FT and NLF airfoils. Then, we discuss the grid generation methods and input parameters used for the structured grids used in the simulations.

C.1 Governing equations

We use ADflow as our CFD solver [Mader et al., 2020]. ADflow is a tool that has been extensively tested for a class of benchmark cases in the past. It solves the compressible Euler, laminar Navier–Stokes transitional and fully turbulent RANS equations using structured multi-block and overset meshes. ADflow has been used in aerodynamic, aero structural, and aero propulsive design optimization of aircraft configurations. Furthermore, we used ADflow to perform design optimization of hydrofoils and wind turbines.

The two equations that are considered in this research are the fully turbulent RANS equation with a SA turbulence model and the transitional RANS equation SA model coupled with the e^n method [Shi

⁴GitHub Repository - <https://github.com/rohitroxp7/UniFoil>

et al., 2020]. The governing equation for compressible–RANS in conservation form is defined as follows

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{f}_c(\mathbf{q}) = \nabla \cdot \mathbf{f}_v(\mathbf{q}, \nabla \mathbf{q}) + \mathbf{s}(\mathbf{q}), \quad (5)$$

where $\mathbf{f}_c(\mathbf{q})$ is the inviscid flux, $\mathbf{f}_v(\mathbf{q}, \nabla \mathbf{q})$ is the viscous flux, and $\mathbf{s}(\mathbf{q})$ is the turbulence source term, \mathbf{q} contains the mean flow state variable and the turbulence variable, $\tilde{\nu}$ (turbulent SA eddy viscosity). The vector of conserved variables \mathbf{q} includes the mean density, mean momentum components, mean total energy, and the working variable of the SA model in order as

$$\mathbf{q} = \begin{bmatrix} \bar{\rho} \\ \bar{\rho} \bar{u} \\ \bar{\rho} \bar{v} \\ \bar{\rho} \bar{E} \\ \tilde{\nu} \end{bmatrix}, \quad (6)$$

where $\bar{\square}$ denotes time–averaging, ρ is density, u, v are the x and y velocities, E is energy and $\tilde{\nu}$ is the Spalart–Allmaras working variable used to compute the eddy viscosity as $\nu_t = \bar{\rho} \tilde{\nu} f_{v1}$, with $f_{v1} = \chi^3 / (\chi^3 + c_{v1}^3)$, $\chi = \tilde{\nu} / \nu$ and c_{v1} is an SA model constant.

The turbulence source term is defined as follows

$$\mathbf{s}(\mathbf{q}) = \begin{bmatrix} 0 \\ \mathbf{0} \\ 0 \\ C_{b1}(1 - f_{t2})\bar{S}\tilde{\nu} + \frac{C_{b2}}{\sigma}|\nabla \tilde{\nu}|^2 - C_{w1}f_w \left(\frac{\tilde{\nu}}{d}\right)^2 \end{bmatrix}, \quad (7)$$

where C_{b1} , C_{b2} , σ , f_w , C_{w1} , d are SA model constants and \bar{S} is defined as the magnitude of the mean vorticity tensor as

$$\bar{S} = \sqrt{2\Omega_{ij}\Omega_{ij}}, \quad \Omega_{ij} = \frac{1}{2} \left(\frac{\partial \bar{u}_i}{\partial x_j} - \frac{\partial \bar{u}_j}{\partial x_i} \right) \quad (8)$$

where S is the magnitude of the mean vorticity tensor, Ω_{ij} is the mean rotation rate tensor, and \bar{u}_i is the mean velocity component in the i -th direction.

For more details, we refer the reader to the NASA website for more information ([NASA Langley Research Center]).

The transitional solver implements the e^n transition model using the linear stability theory. An intermittency function $g \in [0, 1]$ is introduced to control the turbulence onset, which modified the source term of the turbulence equation as follows:

$$g \left(C_{b1}(1 - f_{t2})\bar{S}\tilde{\nu} + \frac{C_{b2}}{\sigma}|\nabla \tilde{\nu}|^2 - C_{w1}f_w \left(\frac{\tilde{\nu}}{d}\right)^2 \right). \quad (9)$$

Here, if $g = 1$ the original SA equation is recovered and we are in the fully turbulent mode. If $g = 0$, the turbulence source term is completely turned off and we are in the fully laminar regime. When g takes the value between 0 and 1, we are in the transitional region. The determination of g is based on the e^n method taking the information of the base flow.

Using the base flow information, once the transition location is found, the turbulence term will affect the main flow terms and thus resulting in a new equilibrium. A nonlinear block Gauss–Seidel method is used to handle the coupling between the RANS and the e^n equations. More details on the transition modeling can be found in Shi et al. [2020].

C.2 Grid generation

We use pyHyp Secco et al. [2021] to generate the volume grid, which uses a hyperbolic volume grid marching scheme to extrude structured surface grids into volume grids. Thus, the grid generation was performed such that the domain is an O-type grid with the airfoil body–fitted to the center of the domain with one cell thickness in the z -direction as shown in Figure 3.

The one-cell-thickness necessity arises from the fact that ADflow does not support pure two-dimensional simulations and therefore one needs to set one-cell thickness in the infinite span direction to assume two–dimensional (2D) nature of the flow. This is essentially a pseudo–2D simulation.

Table 4: Fine grid generation parameters

Description	Value
Number of points along airfoil surface till TE	293
Number of TE points	11
Number of points in marching direction	85
First layer thickness	$10^{-6} c$
Distance from the airfoil surface to march the hyperbolic grid	$100c$

To maintain the same shape for the input data format, we ensured to maintain the same input parameters for the grid generation. The input parameters are summarized in Table 4. In the table, “TE” stands for trailing edge and “ c ” stands for chord, which was set to a value of 1m in all simulations.

D Output file contents

As introduced in Section 3, the dataset comprises three categories of simulations:

1. Fully turbulent simulations for FT airfoils,
2. Fully turbulent simulations for NLF airfoils,
3. Transition simulations for NLF airfoils.

For each of the output files, we describe the contents in the following subsections. All simulation output files follow a consistent naming convention and are provided in CGNS format. The structure and content of these files are summarized in Table 5. In the following subsections, we discuss the output folders and files content into two sub-categories based on the geometry, the first being the FT-airfoil dataset and second being the NLF airfoil dataset.

Table 5: Overview of simulation data categories and file organization.

Governing equation	Airfoil	Folder	Suffix	Freestream File
Fully Turbulent	FT	Turb_Cutout_<1-6>.zip	turb	Airfoil_Case_Data_turb.tab
Fully Turbulent	NLF	NLF_Airfoils_Fully_Turbulent.zip	lam	Airfoil_Case_Data_Trans_Lam.tab
Transition	NLF	NLF_Airfoils_Fully_Turbulent.zip	lam	Airfoil_Case_Data_Trans_Lam.tab

D.1 FT-airfoil dataset

The field data for FT airfoils is organized into multiple cutout folders named:

Turb_Cutout_<1-6>.zip.

Each simulation file follows the naming pattern:

airfoil_<airfoil_num>_G2_A_L0_case_<case_num>_000_surf_turb.cgns .

The integers <airfoil_num> and <case_num> refer to the specific airfoil geometry and the freestream condition, respectively. Detailed freestream conditions for each case are provided in the file:

Airfoil_Case_Data_turb.tab .

Each CGNS file contains the Mach number, pressure coefficient, C_p , and the velocity vector field. These files are compatible with open-source tools such as ParaView and PyVista library Sullivan and Kaszynski [2019] and commercial software such as TecPlot (commercial) for data extraction and visualization purposes.

A visualization of the field data present in the fully-turbulent dataset is shown in Figure 4. In the figure, we show the Mach number, velocity magnitude and pressure coefficient values. It also

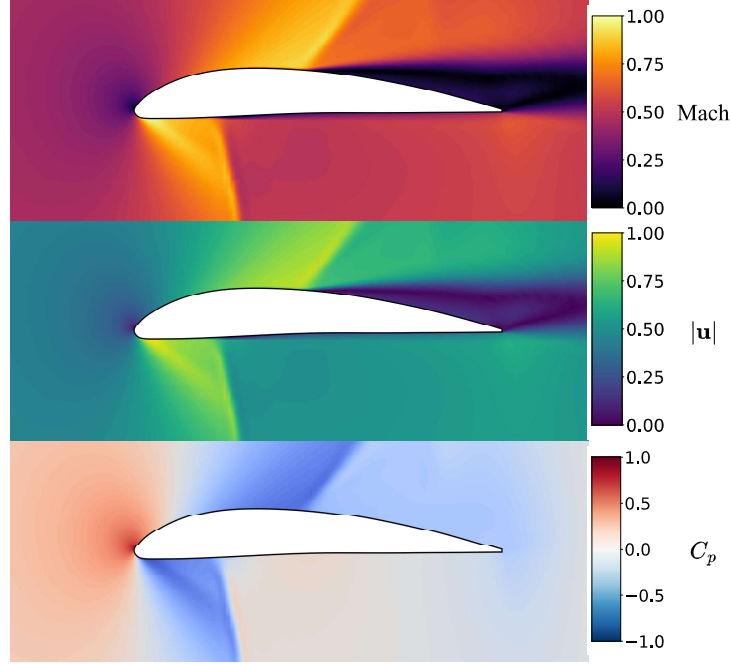


Figure 4: Plots of the data present in the fully turbulent simulations. $|\mathbf{u}|$ is the velocity magnitude and C_P is the coefficient of pressure.

showcases shocks on both the suction and pressure sides, with flow separation aft the shock foot. The velocity magnitude $|\mathbf{u}|$ is computed using the formula

$$|\mathbf{u}| = \sqrt{u^2 + v^2}, \quad (10)$$

where u is the x -velocity magnitude and v is the y -velocity magnitude. Both these values including the velocity magnitude pre-computed through ADflow are available in the CGNS files.

D.2 NLF–airfoil dataset

Two datasets are available for the NLF airfoils, the fully-turbulent dataset and the transition dataset. For both the datasets, the airfoils were run for the same freestream conditions.

The field data for NLF airfoils fully turbulent simulations is organized into a single folder named:

`NLF_Airfoils_Fully_Turbulent.zip`,

with each simulation file inside this folder that follows the name:

`airfoil_<airfoil_num>_G2_A_LO_case_<case_num>_000_surf_lam.cgns`.

As aforementioned, the integers `<airfoil_num>` and `<case_num>` refer to the specific airfoil geometry and the freestream condition, respectively. Detailed freestream conditions for each case are provided in the file:

`Airfoil_Case_Data_Trans_lam.tab`,

for both the fully-turbulent and transition datasets of the NLF–airfoils. Each CGNS file, once again, contains the Mach number, pressure coefficient, C_p , and the velocity vector field, as shown in Figure 4 for the FT-airfoils.

The transition simulation dataset however, follows a slightly different folder structure. Within each archive named `Transi_Cutout_<1-4>.zip`, the data is organized into subfolders following the format:

airfoil_<airfoil_num>_G2_A_LO_case_<case_num> .

Each of these subfolders contains the following files:

- CGNS files containing flow variables including the pressure coefficient, C_p , density, Mach number, and velocity field.
- A slice file in tabular format containing velocity and pressure field data extracted at the transition location along both the suction and pressure surfaces of the airfoil. These values represent a one-dimensional “slice” through the computational domain.

Finally, the archive Transi_sup_data_cutout_<1-4>.zip mirrors this folder structure and includes eight supplementary files per case. These provide extended data on the transition characteristics, enabling deeper analysis of the laminar-to-turbulent transition region. A full description of these supplementary files and their contents is available in the accompanying README file on the GitHub repository⁵.

A visualization of the velocity magnitude field and the field of ratio of turbulent eddy viscosity to the molecular viscosity is shown in Figure 5. The transition onset points are marked on the airfoil surface using the red-colored arrow on the suction side of the airfoil. The transition to turbulence was only observed for the suction side in this case. The location for transition point was obtained from the “transiLoc.dat” file.

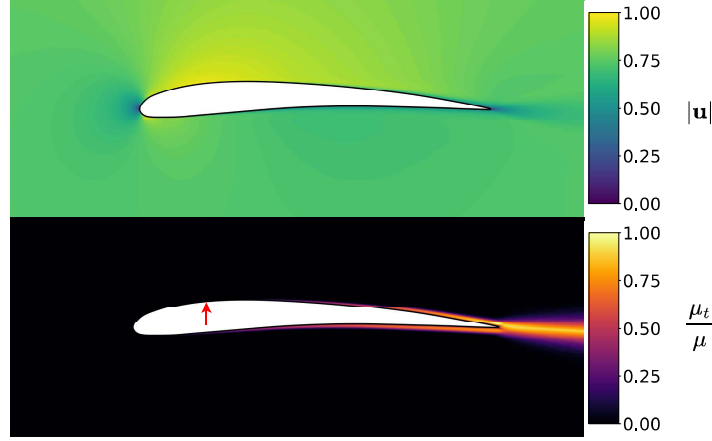


Figure 5: Velocity magnitude field and eddy viscosity ratio field in one of the transition flow simulations of the dataset. The laminar–turbulent transition point for the suction–side is located by the red arrow in the lower figure.

The e^N method predicts the laminar separation and Tollmien–Schlichting (TS) waves induced transition mechanism. The TS waves amplification is characterized by a factor called N_{TS} factor. Once it reaches a threshold of 7, the unstable wave is amplified in such a magnitude that the turbulence onset is triggered, which is when the flow transitions to fully–turbulent.

In the e^N transition model, the reduced TS wave frequency is given by

$$f = \frac{\omega}{Re_\delta}, \quad (11)$$

where ω is the dimensionless angular frequency and Re_δ is the displacement–thickness based Reynolds number. If α_i is the local dimensionless spatial amplification rate for the TS wave, then the N factor is given by

$$N = \int_{x_0}^x \alpha_i(x, f) dx, \quad (12)$$

where x is the spatial location in the computational domain and x_0 is the starting point of calculation. Thus, the N_{TS} factor is defined as the maximum amplification among all waves

$$N_{TS} = \max_f (N(f)). \quad (13)$$

⁵GitHub Repository - <https://github.com/rohitroxp7/UniFoil>

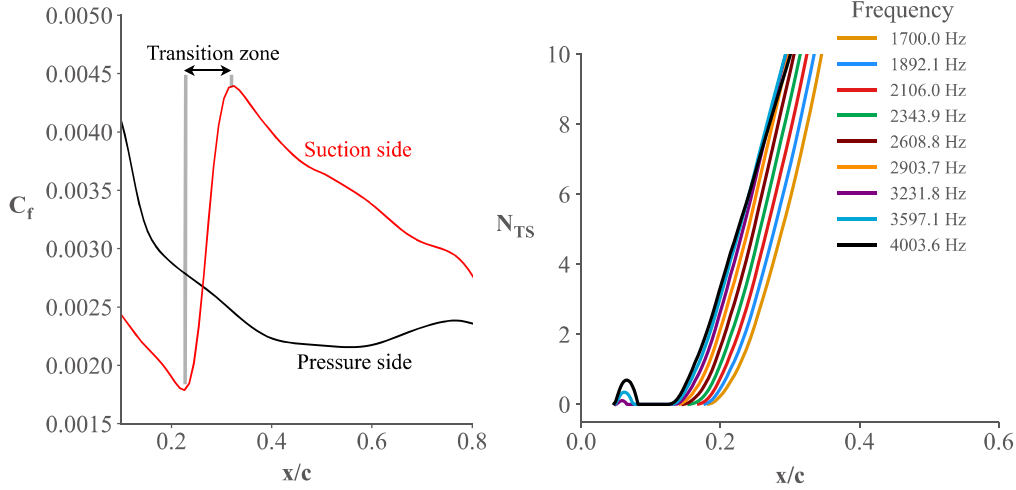


Figure 6: Skin-friction coefficient C_f and the N_{TS} factor vs x/c where c is the chord. The N_{TS} factor is plotted for their different frequencies. The region before the transition zone is laminar flow region and region after the transition zone is the fully-turbulent flow region.

Out of the eight files, the key files are “`nfactor_ts.dat`” and “`transiLoc.dat`”. The former has the TS wave factor N_{TS} and the latter has the transition point location, if detected, on the suction and pressure surfaces. Details on accessing and visualizing these files can be found in the GitHub repository⁶. Details on the transition characteristics and the factors can be found in the paper by Shi et al. [2020].

Figures of the skin-friction coefficient C_f and the N_{TS} factor plotted against x/c from the `nfactor_ts.dat` file is shown in Figure 6, for the airfoil shown in Figure 5. The N_{TS} factor for TS waves is computed as shown in Equation (13). When transition occurs, the skin-friction coefficient starts to abruptly increase. The transition region highlighted in Figure 6 can be seen for the suction-side of the airfoil where the skin friction coefficient suddenly rises to a value of roughly 0.0043 before it starts to dip. For the N_{TS} factor, values begin to increase around x/c of 0.2, in-line with the observation from the skin-friction coefficient plot.

E Data preprocessing

To ensure a smooth handling of input data, maintaining the same input shape to the neural network and to preserve the grid resolution without loss of data that typically occurs when interpolating onto a uniform grid, the data from the O-grid was reshaped into a neat 2D array. The process of reshaping is highlighted in Figure 7. We start at the TE and move clockwise along the airfoil surface to loop till we reach back to the TE. This forms the first bottom-most layer of the 2D array. We then move to the next layer and repeat till all the layers are completed.

Since the same tool (pyHyp) and grid generation inputs from Table 4 were used, we maintained the same number of “O-grid rings” and ordering of points to reshape all input data in a similar manner. Figure 7 shows this concept in the top row and shows the implementation of the reshaping algorithm for the velocity field in the bottom row.

F Preliminary results for Machine Learning

A neural network was trained using the *UniFoil* dataset. In this section, we discuss results from the training of this neural network. First, we discuss the details on network architecture. This includes details such as neuron activations, number of layers and the neural network setup. Then, we discuss the results from the neural network by comparing the reconstructed values against the ground-truth data.

⁶GitHub Repository - <https://github.com/rohitroxp7/UniFoil>

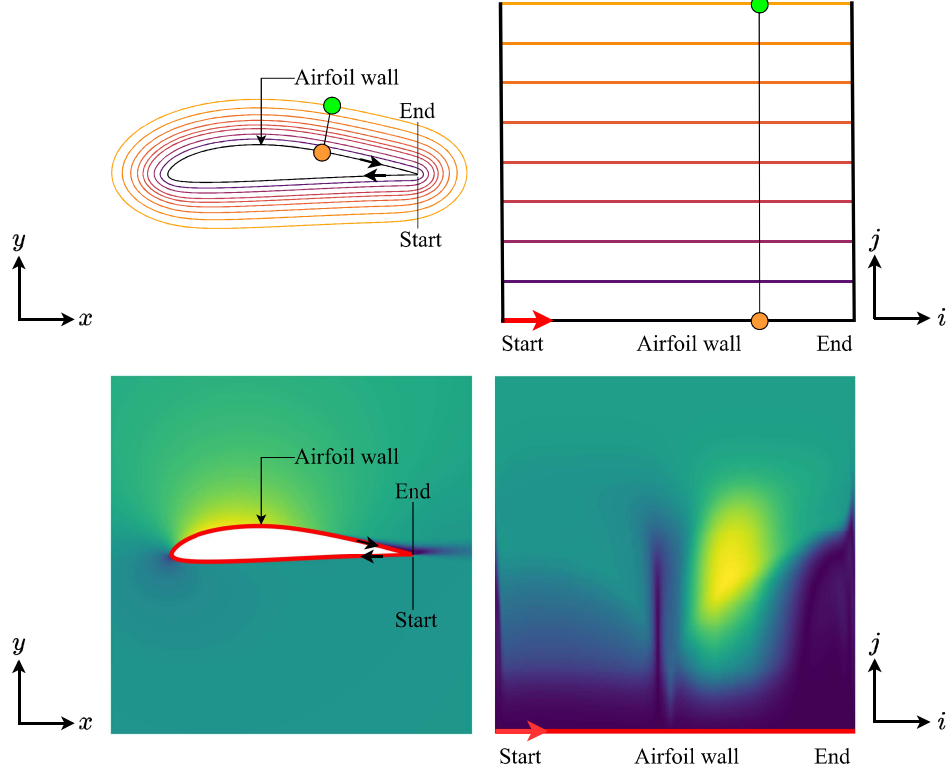


Figure 7: The input data is transformed from an O-grid mesh to a uniform rectangular grid by reordering the points in an array-like structure, without need to preserve their original spatial arrangement. The first row presents the concept and second row presents the application to the velocity-field data. The first column represents the data in the Cartesian coordinate system while the second column represents the data simply arranged in a 2D space using the node numbering, following the order of path guided by arrows in the figures.

F.1 Neural network architecture

The neural network trained on the UniFoil dataset is an encoder-decoder model with a latent space deep neural network (DNN). This model is trained on the pressure-coefficient data extracted from each airfoil simulation. The model is split into two main components with different layer designs used to down-sample and up-sample the input space.

The encoder consists of two smaller encoding blocks made of a convolutional layer and a max pooling layer. The decoder is the opposite and consists of two blocks made of a convolutional layer and a up-sampling layer. Once the model is trained, the encoder can be used to convert all of the data into the model’s latent space. These latent space values are then saved and used to train the DNN. The train-test split used for this model was 80% / 20%. The model was trained for 100 epochs. Model batch size was 32. All neurons’ activations were set to ReLU.

An overview of the architecture of the neural network is shown in Figure 8. A summary of the training setup is shown in Table 6. In the table the “Param #” field represents the quantity of variables that can be modified during the training procedure. Densely connected layers such as the ones prevalent in the DNN have a large number of these parameters as all 6134 nodes are connected to every single node below.

The model was trained by first preparing the encoder-decoder model and training it on the reshaped pressure coefficient data. Once this model is trained, the encoder is used to convert all of the pressure coefficient data into latent space values which are saved separately. These latent space values are then combined with matching airfoil feature data in order to train the deep neural network. When the deep neural network is done training, the model can be used to predict an airfoils pressure coefficient field

by first converting the feature data into the latent space and then passing this latent space data into the decoder. This results in a pressure coefficient field in the square grid format which must finally be reshaped into the O-Grid using the geometry data from the airfoil.

It is possible to simplify this method by cutting out the encoding step entirely and instead training a hybrid deep neural network and convolutional model. This can have a similar architecture but doesn't store the latent space information or a encoder allowing it to operate more efficiently. This approach should be tested in the future with different layer architectures to see if it produces more accurate results.

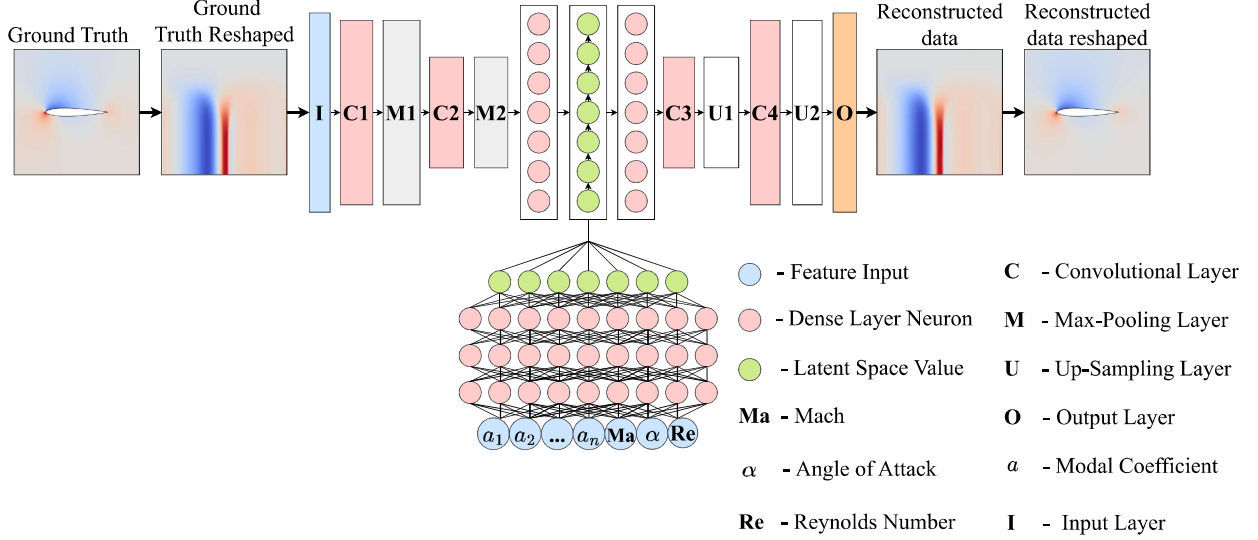


Figure 8: An overview of the entire training pipeline. The modal coefficients a_i are obtained from Equation (4).

F.2 Training

The neural network was trained using the FT-airfoils dataset. 400,000 simulations from the dataset were used for training. During the training process, the convolutional network and DNN errors were plotted against the number of epochs. We use the following metrics for the loss function

$$\begin{aligned} \text{loss}_{\text{encoder-decoder}} &= \frac{1}{n_{\text{sample}}} \sum_i \frac{\|C_{p,\text{real}}^{(i)} - C_{p,\text{pred}}^{(i)}\|}{\|C_{p,\text{real}}^{(i)}\|}, \\ \text{loss}_{\text{DNN}} &= \frac{1}{n_{\text{sample}}} \sum_i \frac{\|C_{p,\text{real}}^{(i)} - C_{p,\text{pred}}^{(i)}\|}{\|C_{p,\text{real}}^{(i)}\|}, \end{aligned} \quad (14)$$

where $C_{p,\text{real}}$ is the pressure coefficient from the ground-truth data and $C_{p,\text{pred}}$ is the pressure coefficient from the predicted data. A plot of the training error as shown in Equation (14) is shown in Figure 9. It is observed that the error steadily decreased for both networks and flattened out after 30 epochs for the DNN while still showing a slight but very slow decreasing trend for the convolutional network layer.

F.3 Prediction

Once the training was complete, a plot for the predicted values of pressure coefficient $C_{p,\text{pred}}$ against the ground truth values of $C_{p,\text{real}}$ was made for all the dataset simulations and their reconstructed field values. This plot is shown in Figure 10. In an ideal case, the network would predict with no loss of accuracy. In such a scenario, we would have a one-one mapping of the predictions and ground truth data, yielding a straight line, $C_{p,\text{real}} = C_{p,\text{pred}}$. This is overlaid onto the plot as well for an ideal-case

Table 6: Neural Network Architecture Summary

Component	Layer Type	Output Shape	Param #
Encoder	Input	(32, 84, 292, 1)	0
	MaxPooling2D	(32, 42, 146, 1)	0
	Conv2D	(32, 42, 146, 32)	320
	MaxPooling2D	(32, 21, 73, 32)	0
	Conv2D	(32, 21, 73, 4)	1,156
	Flatten	(32, 6132)	0
	Dense	(32, 6132)	37,607,556
	Output	(32, 6132)	0
Decoder	Input	(32, 6132)	0
	Dense	(32, 6132)	37,607,556
	Reshape	(32, 21, 73, 4)	0
	Conv2DTranspose	(32, 21, 73, 32)	1,184
	UpSampling2D	(32, 42, 146, 32)	0
	Conv2DTranspose	(32, 42, 146, 8)	2,312
	UpSampling2D	(32, 84, 292, 8)	0
	Conv2DTranspose	(32, 84, 292, 1)	73
	Output	(32, 84, 292, 1)	0
Deep Neural Network	Input	(32, 17)	0
	Dense	(32, 6134)	122,680
	Dense	(32, 6134)	37,632,090
	Dense	(32, 6134)	37,632,090
	Dense	(32, 6134)	37,632,090
	Dense	(32, 6134)	37,632,090
	Dense	(32, 6134)	37,632,090
	Output	(32, 6134)	0

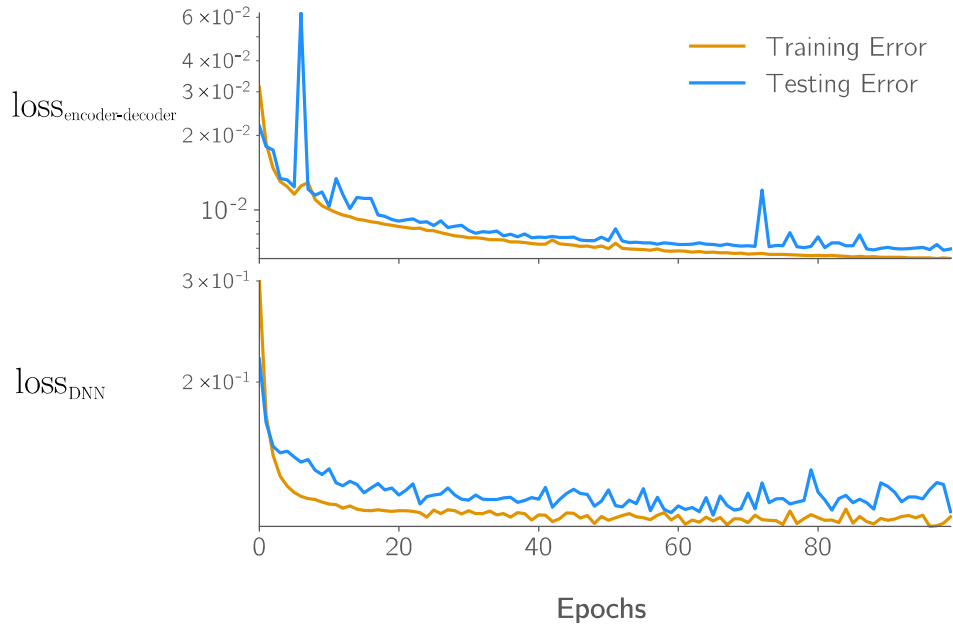


Figure 9: Training error against the number of epochs for the convolutional network and DNN.

comparison. In the figure, it is observed that the prediction values were appreciably close to the ground truth values, however the slope of the data does not match the slope of the one–one mapping. In this case predicted C_p values are greater than the ground truth near the high end of the plot while predicted C_p values near the low end of the plot are less than the ground truth. This shows that the predicted values are scaled slightly more than the ground truth data with the effect becoming more prevalent the further the values are from zero.

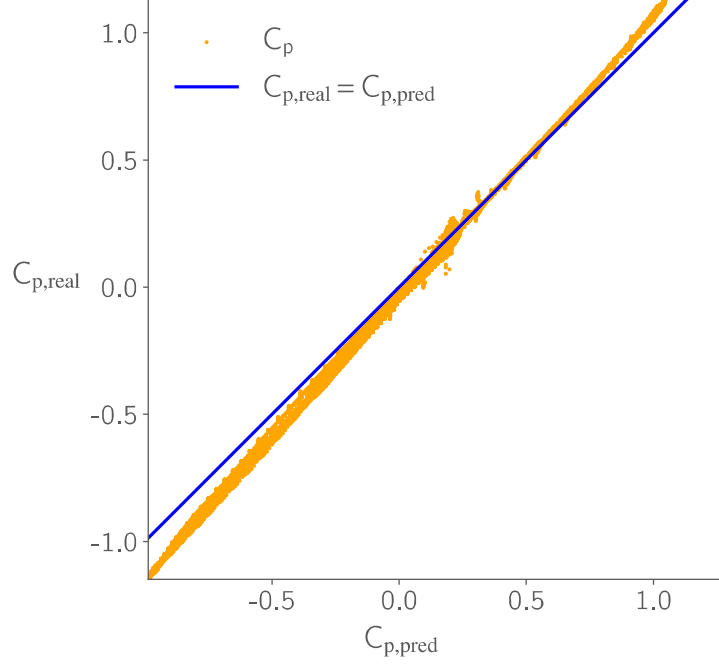


Figure 10: Predicted and ground truth values for C_p .

Next, to quantify the errors in the learning process, all of the input data was regenerated for the flow conditions they were simulated for. An L_2 error difference study between the actual data and the predicted data for each airfoil was then carried out and the number of airfoils quantity was plotted against the L_2 error for every airfoil in the 400,000 simulations and is shown in Figure 11. The relative L_2 error is defined as follows

$$\frac{\|C_{p,\text{real}}^{(i)} - C_{p,\text{pred}}^{(i)}\|}{\|C_{p,\text{real}}^{(i)}\|}, \quad (15)$$

where $C_{p,\text{pred}}$ is the pressure coefficient values from the predicted data, the $C_{p,\text{real}}$ is the pressure coefficient from the ground-truth data and i is the simulation count. This histogram plot gives an overview of the error distribution in the dataset. Roughly 35,000 airfoil simulations from the 400,000 had the highest error of approximately 10^{-1} .

Finally, we present a visualization of the ground truth C_p fields, reconstructed C_p fields and the L_2 error for those plots for some of the airfoils randomly picked from the dataset in Figure 12. Appreciable comparisons can be seen for the field values between the ground truth and reconstructed C_p data fields.

We can determine how accurate our model is at reproducing the airfoil data field by calculating the L_2 norm error between the ground truth and predicted output. We can see two examples of higher L_2 norm in the 3rd and 4th airfoil in Figure 12, with error of 0.131 each.

Another way to examine the prediction quality of the model is to plot the absolute error $|C_{p,\text{real}}^{(i)} - C_{p,\text{pred}}^{(i)}|$ between the ground truth and the predicted output. This allows us to visualize areas of high error such as the shock present above the 3rd and 4th airfoil shown in Figure 12. Often the model

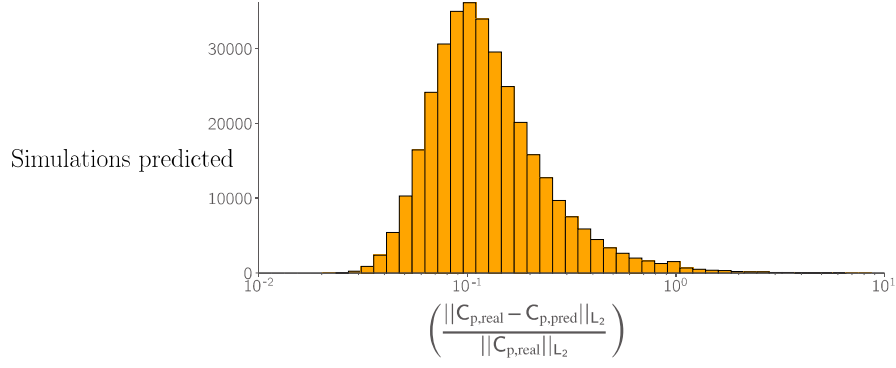


Figure 11: Number of airfoil simulations vs L_2 norm of the differences in C_p between ground truth and predictions.

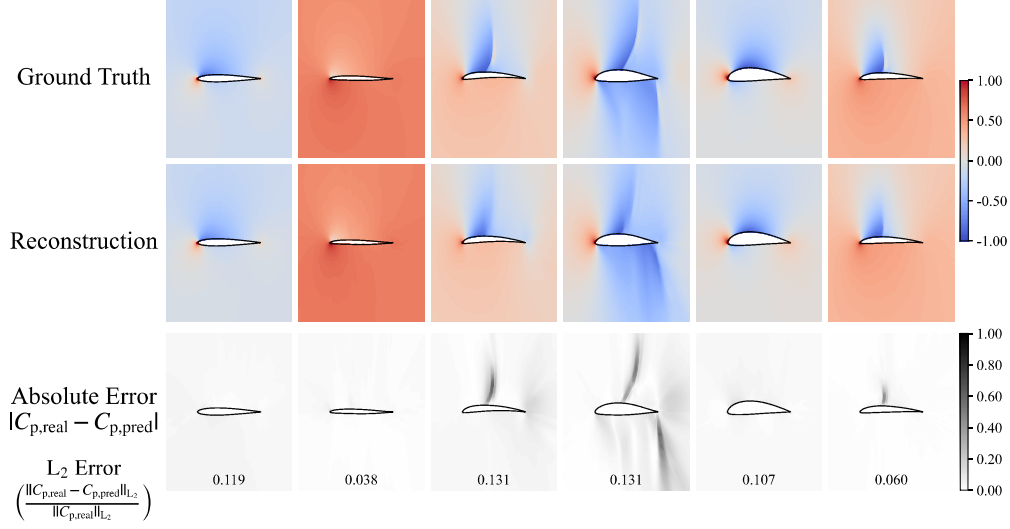


Figure 12: C_p field contours from the ground truth data (first row) and reconstructed data (second row). Absolute error field is plotted in the third row.

fails to fully represent the high pressure deltas of a shock resulting in high error present along the surface of a normal shock.

Despite this the model is capable of generating pressure coefficient data that can be used to locate the shock if not fully map it. This method also allows us to see areas of consistent error such as the field around the 1st and 5th airfoils.

Unlike the sharp discontinuities that appear near shocks this error is far more gradual and represents a failure of the model to determine what the average pressure coefficient values are far from the surface of the airfoil. It is possible that this is because the highest resolution input data makes up a small portion of the shown image. Since the unwrapping method extracts the pressure coefficient values as seen on the O-Grid, the model has a high bias to predicting values correctly closest to the airfoil surface.

As the distance to the surface of the airfoil increases, the resolution of the input data and thus the expected accuracy of the model decreases. Another possible cause of the high L_2 error in these figures is a low L_2 norm value for the real C_p . Since this term is in the denominator of the L_2 error formula, an extremely low value here will cause any small deviations in the numerator to result in high values for the error.

G Acknowledgement

The first author would like to thank Dr. Shaowu Pan and Dr. Sicheng He for the NSF–allocations on Purdue University’s ANVIL cluster which provided the computational resources for the dataset generation.