This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101056732

multi-disciplinary digital - enablers for NEXT-generation AIRcraft design and operations

D2.2 – Instant CFD Demonstrator

Document authors	Francesco Montomoli
Document contributors	All WP2 members

Achievements:

Several methodologies as Neural Network (NN) architectures have been developed, adapted and tested in task 2.1, such as lambda-DNN, U-Net, l-net and autoencoders to describe flows past geometries, with different accuracy, speed up and applicability

As deliverable the demonstrator highlights the potential of ML methods to be be used in multidisciplinary design optimization (WP3/W4) and digital twin development (WP6).

Keywords
Machine Learning, instant CFD, DNN

Funded by the European Union. Views and opinions expressed are however those of the author(s) only
and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Information Table

PROJECT INFORMATION	
PROJECT ID	101056732
PROJECT FULL TITLE	NEXTAIR - multi-disciplinary digital - enablers for NEXT-generation AIRcraft design and operations
PROJECT ACRONYM	NEXTAIR
START DATE OF THE PROJECT	01/09/2022
DURATION	36 Months
CALL IDENTIFIER	HORIZON-CL5-2021-D5-01
PROJECT WEBSITE	Home (nextair-project.eu)

DELIVERABLE INFORMATION	
DELIVERABLE No AND TITLE	D2.2
TYPE OF DELIVERABLE[1]	Demonstrator
DISSEMINATION LEVEL[2]	Public
BENEFICIARY NUMBER AND NAME	14, ICL – Imperial College London, UK
AUTHORS	Anirudh Rao, Francesco Montomoli
CONTRIBUTORS	+ all WP2 members
WORK PACKAGE No	2
WORK PACKAGE LEADER	ICL
COORDINATOR VALIDATION DATE	7/11/2024

ACRONYM	MEANING
CNN	Convolutional Neural Network
DNN	Deep Neural Network

1 Use one of the following codes: R=Document, report (excluding the periodic and final reports)
DEM=Demonstrator, pilot, prototype, plan designs
DEC=Websites, patents filing,press & media actions, videos, etc.
OTHER=Software, technical diagram, etc.
ORDP : Open Research Data Pilot.

2 Use one of the following codes: PU=Public, fully open, e.g. web
CO=Confidential, restricted under conditions set out in Model Grant Agreement
CI=Classified, information as referred to in Commission Decision 2001/844/EC.

1. Introduction

The following activities have been carried out during WP2

T2.1 – Instant computational fluid dynamics with enhanced accuracy [ICL, OPT, NTUA, DAV, M1-M24]

ST2.1.1 - Instant flow-field estimation & dimension reduction [ICL, OPT, NTUA, M1-M24]

- **ICL** developed a CNN for real time CFD simulations (on real geometries within service degradation), reducing computing time by 2 order of magnitudes.
- **OPT** developed autoencoder, applied to gyroids for geometrical reconstruction and CNN for field estimation.
- **NTUA** proposed the λ-DNN architecture and used it to support MDO problems by replacing one or more disciplines, leading thus to hybrid algorithms.

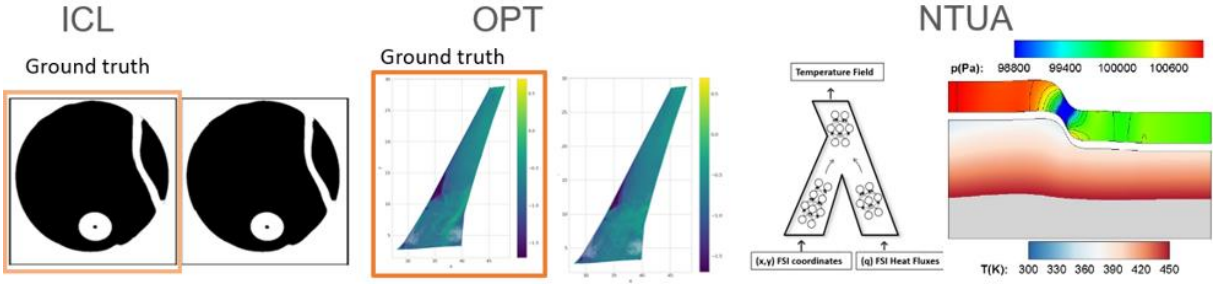

Figure 1: ST2.1.1 results at a glance.

ST2.1.2 - Enhanced accuracy via data fusion [NTUA, DAV, M1-M24]

- **NTUA** developed DNN surrogates for turbulence and transition closure to be used in shape optimizations carried out by evolutionary algorithms. The role of the DNN was to provide the turbulent viscosity field at each iteration of the solver of the mean flow equations by replacing the turbulence and transition models.
- **DAV** Applied data fusion to intakes.

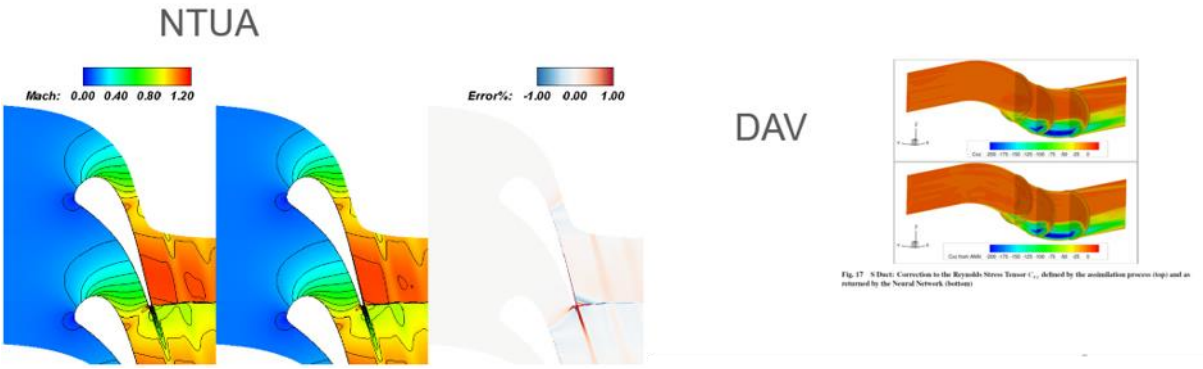

Figure 2: ST2.1.2 results at a glance.

2. λ-DNN for fluid flow-field and MDO & DNNs for turbulence closure (NTUA)

2.1. λ-DNN for fluid flow-field and MDO

The work performed in NTUA was based on the so-called λ-DNN architecture, after its two-branch shape that looks like the Greek letter λ, Figure 3-left. It consists of two separate input branches that allow for separate processing of input data with different physical meaning.

The λ-DNN architecture was applied, among others, in a Conjugate Heat Transfer (CHT) case, namely a high-temperature solid domain cooled by a low-temperature fluid flowing in an adjacent duct. The λ-DNN was trained on a database consisting of 30 geometries evaluated on the exact CHT tool. The left branch of the λ-DNN consisted of one single layer with 512 neurons and processed the coordinates of the solid domain, while the right branch also consisted of one single layer with 512 neurons and processed the heat fluxes at the fluid solid interface. These were merged into a single layer of 512 neurons that led to the output, i.e. the temperature field over the solid domain. The temperature fields as resulted from simulations with the exact CHT solver and the hybrid CFD/λ-DNN solver, as well as their relative error, are presented in Figure 3.

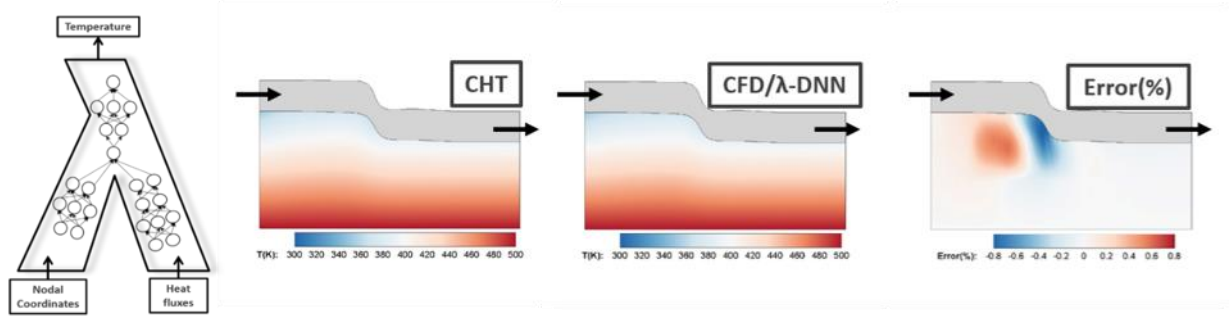

Figure 3: From left to right: The λ-DNN architecture, named after its shape that looks like the Greek letter λ. Each circle/layer comprises a number of neurons/layers. Temperature field at the solid domain as computed by the exact CHT solver and the hybrid CFD/λ-DNN solver); relative error between the two fields. This is a new geometry not included in the database. Axes not in scale.

The CFD/λ-DNN solver was 24% faster in terms of wall-clock time compared to the exact CHT solver, this also includes the cost of forming the database and train the network. The CFD/λ-DNN solver was then used in a multi-objective optimization aiming for min. total pressure losses between the inlet and outlet of the fluid domain (F_1) and min. overheat at the solid domain (F_2), Figure 4. The pressure field on the fluid domain and the temperature field on the solid domain are presented for the two edges of the front of non-dominated solutions, grey area representing the overheated region. Consequently, it can be concluded that the use of the CFD/λ-DNN solver in a multi-objective optimization can offer immediacy and accuracy.

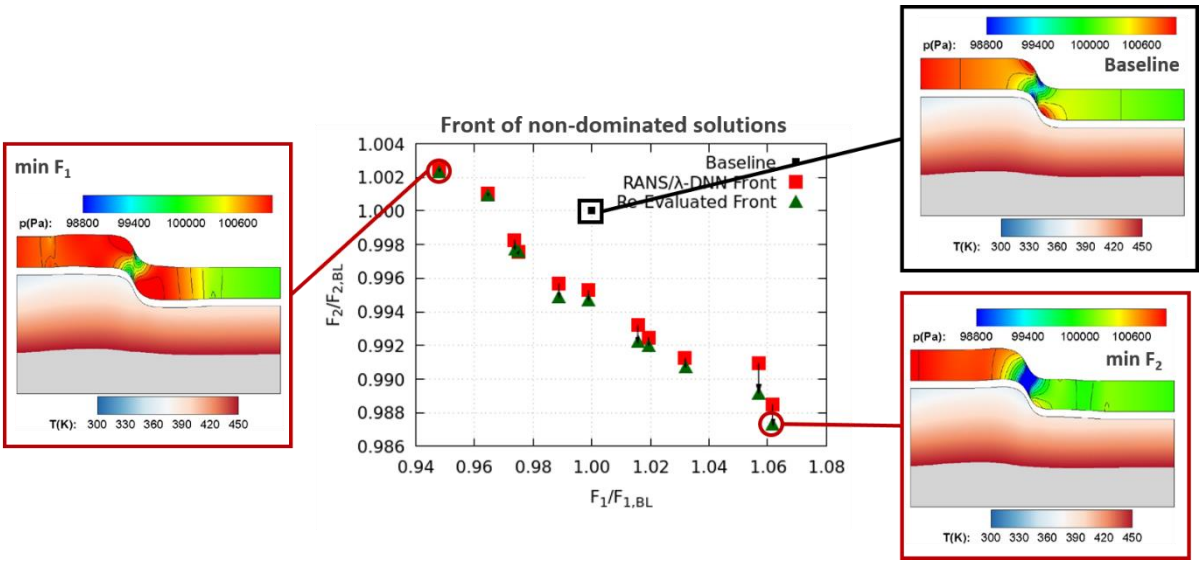

Figure 4 Front of non-dominated solutions from the Metamodel Assisted Evolutionary Algorithm computed by CFD/λ−DNN (filled red squares). The front members are re-evaluated on the CHT solver resulting in a new non-dominated front (green triangles). The computed "optimal" front members remain non-dominated. Pressure and temperature fields on the fluid and solid domain, respectively, for the baseline geometry and the geometries that correspond to the two edges of the front. Axes not in scale.

2.2. DNNs for turbulence closure

The work performed by NTUA regarding the enhanced accuracy via data fusion aimed to develop and assess DNNs for turbulence closure. The key idea was to use DNNs in order to replicate the numerical solution of the turbulence and transition model PDEs (1-4 equation in total) by providing the turbulent viscosity field at each iteration of the mean flow equations. This is illustrated in Figure 5. The goal was to decrease the computational cost of the CFD analysis ensuring satisfactory prediction accuracy.

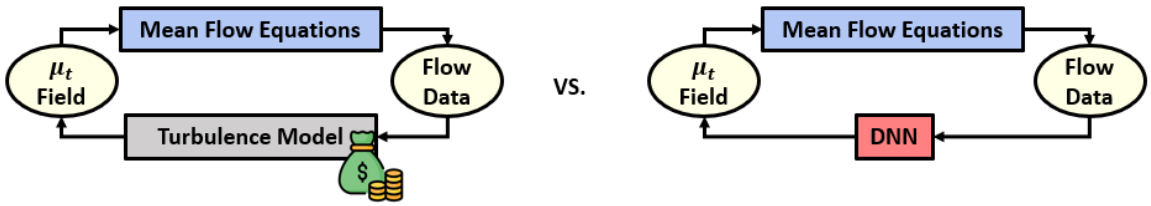

Figure 5: An iterative solver of the RANS equation using the turbulence/transition models (left) and the hybrid DNN version (right). It corresponds to a single pseud-time iteration of the RANS-TM or RANS-DNN solver.

The RANS-DNN solver was used, among others, for the shape optimization of a highly loaded transonic turbine blade. The DNN was trained on a database of 100 blade shapes and consisted of 5 layers with 512, 512, 4096, 4096 and 256 neurons each. The RANS-DNN solver was 38% faster compared to the RANS-TM one and was used in a constrained optimization aiming to reduce the mass-averaged total pressure losses of the blade. The accuracy of the solver and the convergence history of the optimization are shown in Figure 6. It can be seen that for the same computational budget of 250 time units (wall-clock time), the RANS-DNN solver (red line) reaches a solution of better quality compared to the exact RANS-TM solver (black line). A different point of view shows that in order to obtain a solution of the same quality as that of the RANS-DNN run, the RANS-TM model needs 250 more evaluations, i.e. cost increases by 50%, dotted ellipsis. Needless to say that the cost of forming the database and train the network is included in the overall cost.

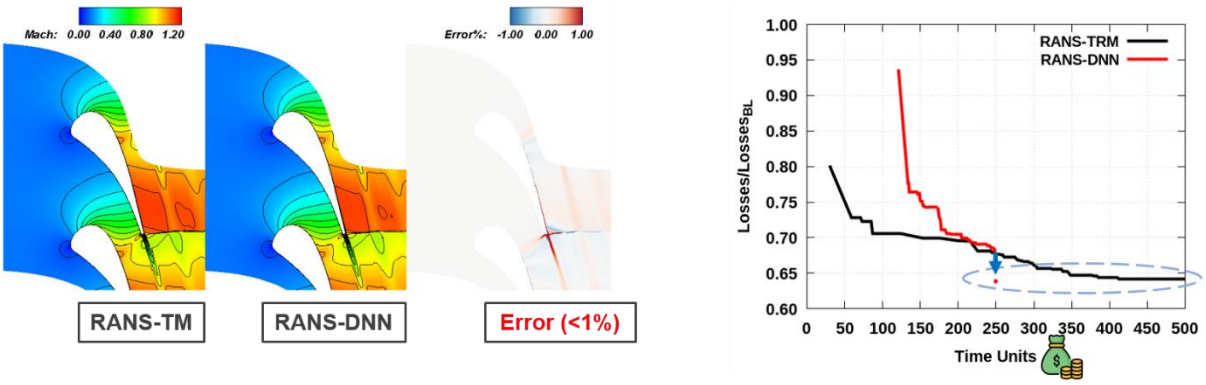

Figure 6: Flow field using the RANS-TM and RANS-DNN solver (along with the corresponding error) and their use in an Evolutionary Algorithm-based shape optimization.

3. Enhanced Turbulence modelling through data-fusion (DAV)

In order to mitigate the shortcomings of RANS simulations for turbulence modelling in a separated channel flow with adverse pressure gradient, an artificial neural network is used to compute a correction to the turbulent stress of a RANS model in the momentum equations. Test cases include the flow over a 2D periodic hill (with varying Reynolds number and hill geometry) and the flow in a 3D diffusing S duct. The training and validation data were obtained from comparison between predictions from the RANS model and publicly available direct numerical simulation (DNS) and wall resolved large eddy simulation (WRLES) results for the 2D flow and detached eddy simulation (DES) computations for the 3D flow. The model-based approach was selected to derive the correction terms required for the learning step. Different neural network (NN) structures were implemented and tested. Results show a good ability of the NN enhanced model to replicate the reference solutions. The possibility for geometry or Reynolds number extrapolation was also explored for the 2D case, with the assessment of the neural network's performance on cases on which it was not trained. Significant improvement in the prediction of the extent of the separation region was obtained. Results presented demonstrate a promising step towards the goal of including neural networks in industrial turbulence models to enhance accuracy and improve local design space exploration. Reference results are presented in Figure 7,Figure 8, Figure 9,Figure 10 and Figure 11.

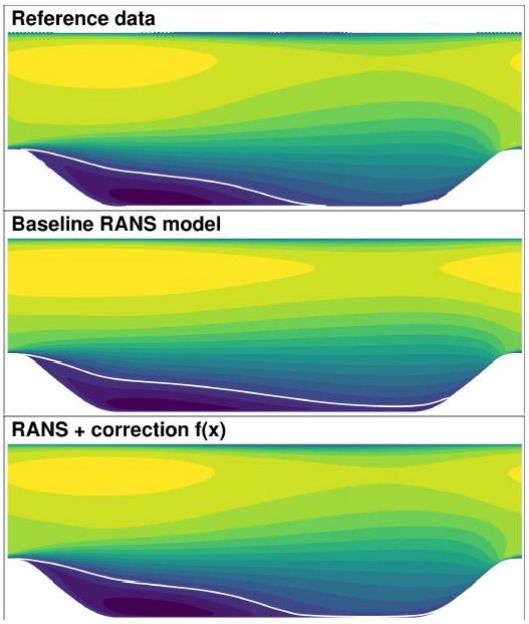

Figure 7: Mean stream-wise velocity field for the periodic-hill case. The reverse flow region is delimited by a thick white line.

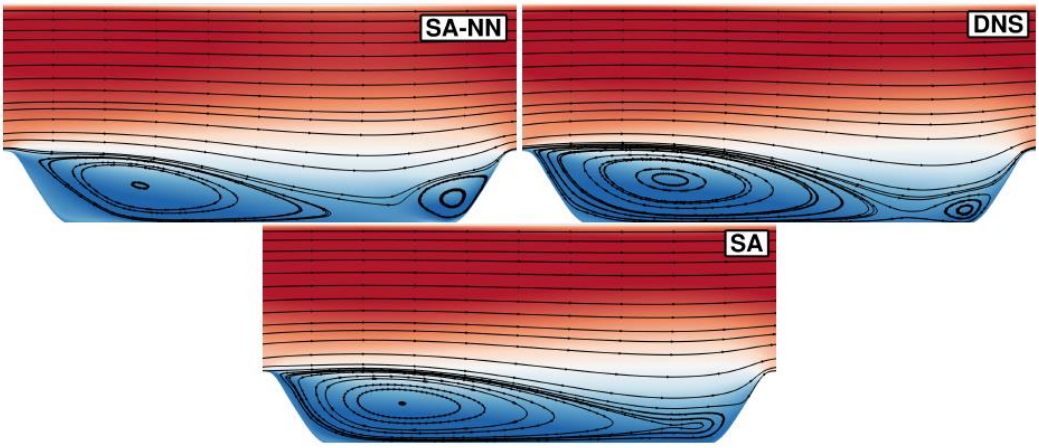

Figure 8: Geometry extrapolation, Re 5600, a=0.5 (short hill), streamlines.

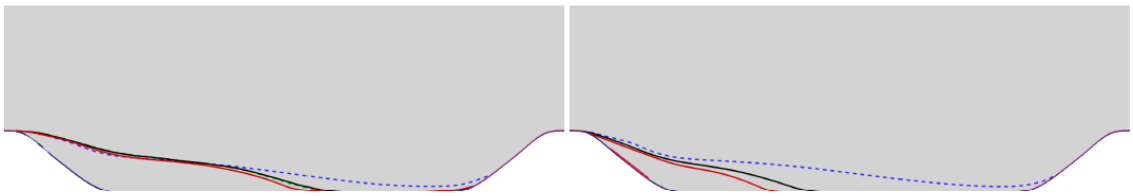

Figure 9: Reynolds extrapolation, 2D periodic-hill case, Re=2800 (left) and Re=10595 (right). Black reference, blue RANS, red NN-RANS by DAV. Reference and SA-DAV lines are almost superimposed.

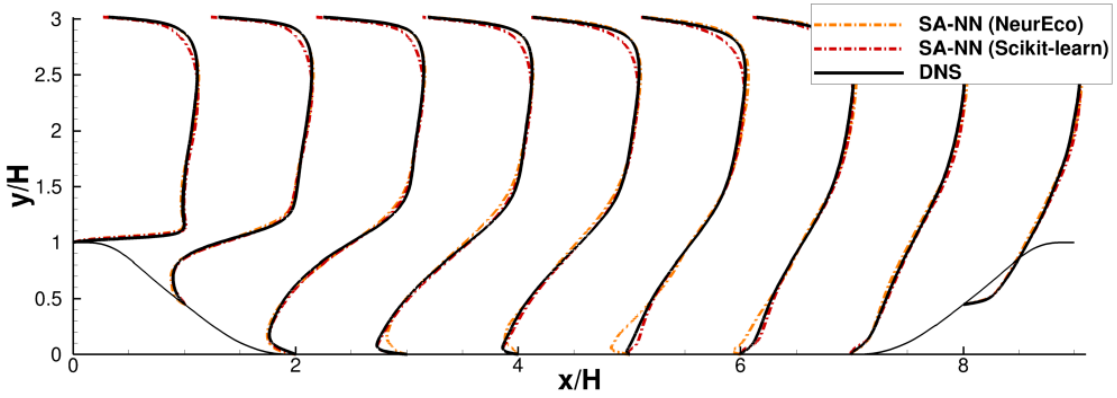

Figure 10: 2D periodic-hill case at Re=2800; detailed comparison between results from different neural networks structures.

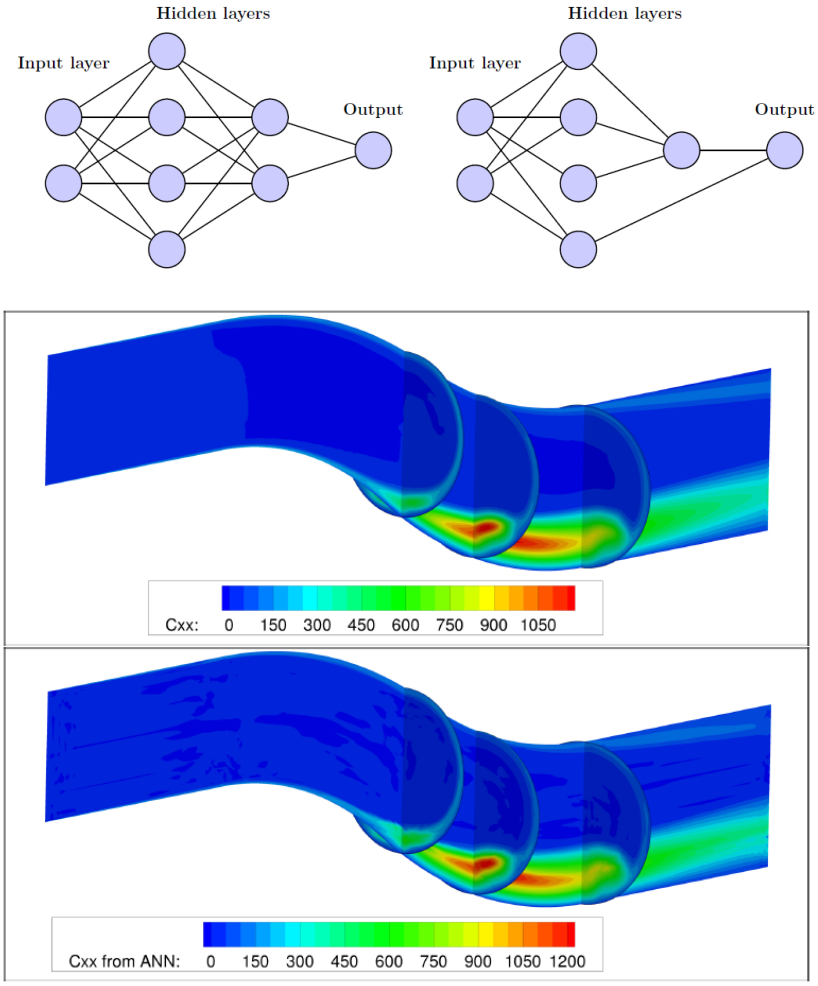

Figure 11: S Duct test case. Top: fully connected vs parcimoneous ANN. Bottom: correction to the Reynolds Stress Tensor Cxx defined by the assimilation process and as returned by the Neural Network.

4. Instant flow predictions for conjugate heat transfer in microchannles using Deep geometric learning (OPT)

4.1. Introduction

Technical activities carried out by OPT in ST2.1.1 focused on the definition and implementation of a ML model for instant flow predictions in multiscale Conjugate Heat Transfer (CHT) problems. The final goal is to deploy the techniques developed in ST2.1.1 for the creation of a Digital Twin of the Micro-Channel Heat eXchangers (MCHX) covered in WP6.

At the beginning of ST2.1.1, an explorative activity was conducted in order to identify the most promising ML architectures for solving multiscale CHT problems. The initial investigation focused on the following three main aspects:

1. Applicability to the multiscale CHT problem,
2. Accuracy of CFD predictions,
3. Model performances in terms of costs and computational resources required by the training.

Two families of ML models were considered:

- Physically Informed Neural Networks (PINNs)
- Deep Convolutional Neural Networks based on U-Net architecture.

In PINNs, the neural network is trained to “infer” the equations governing a physical phenomenon from high-fidelity data (i.e., simulations, experimental data, etc.). Although these models can produce accurate predictions for relatively simple problems, the exploratory activity highlighted the following main issues:

- low accuracy in case of complex domains and/or topological changes,
- challenges in the handling of complex boundary conditions.

In MCHX, the geometry of micro-channels is characterized by a significant geometric complexity. Moreover, accurate handling of boundary conditions is essential for robust and accurate simulations of the heat exchanger. For these reasons, the second class of models (deep convolutional networks based on the U-Net architecture) were selected as the model of choice.

U-Net architectures are characterized by an encoder-decoder structure, and are commonly employed in *image-to-image translation* problems. In an image-to-image translation task, the ML model is trained to “translate” an input image into an output image. In the context of CFD predictions, the input image is represented by a 3D image of the input geometry (rasterized geometry), while the output image is represented by a 3D image with multiple channels, each corresponding to a fluidynamic field of interest (velocity, pressure, temperature, etc.).

During the initial exploratory activity, some critical issues emerged also for this second class of architectures. Since these models are “agnostic” to the physics of the problem at hand, U-Net architectures are usually data-hungry. In order to obtain a model with good predictive capabilities, a significant amount of data is required during the training. This is obviously impractical due to the high computational cost of CHT simulations. Moreover, the quality of the results is limited by the resolution of the input image, resulting in geometric variations smaller than the pixel size being ignored by the model. In order to overcome the above problems, the ML model was developed using a *divide and conquer strategy*. Instead of training a single model to learn the mapping from the space of the input geometries to the

space of CFD solutions (both highly dimensional), two different models were designed and trained separately:

- A deep geometric learning model, which is responsible of generating an encoding of the lattice geometry,
- A convolutional decoder, which produces instant flow predictions starting from an encoding of the (local) lattice structure.

The task of the encoder is to “condense” the input geometry into a limited number of parameters (thus achieving dimensionality reduction). The encoding of the input geometry is then used as the input to the U-Net decoder. The latter’s task is to provide a prediction of the fluid dynamic fields starting from the encoding of the input geometry, i.e. the lattice. In order to achieve a sub-pixel accuracy, the input geometry is given in terms of Signed Distance Function (SDF) instead of using a “staircase approximation”, thus removing any limitations related to the resolution of the input image.

4.2. Deep geometric learning

Deep-geometric learning is based on the idea of synthetically describing a heterogeneous family of geometries through a limited number of parameters (latent code), which efficiently encode shape and topology. Most of the architectures employed for deep geometric learning are based on an encoder-decoder structure, where the encoder provides the latent code corresponding to the input geometry. However, since the latent code is not interpretable, a decoder is required during the training in order to quantify the encoding error. Such error is then back-propagated to update both the encoder and the decoder. Once the model is trained, the encoder can be used to encode new (unseen) shapes, while the decoder can be used to generate new topologies by sampling the latent space learnt during the training phase. These generative capabilities fit perfectly in a geometric optimization context. However, the main drawbacks of the encoding-decoding pipeline are typically related to robustness and slow convergence of the training.

To overcome these difficulties, an “auto-decoder” strategy was adopted, which requires only to train a decoder. In fact, the ML architecture consists of a convolutional decoder which takes the coordinates of a point in the physical space and the latent code of the shape as inputs. The decoder returns a prediction of the SDF value at the input point. This particular choice is motivated by the fact that working with point clouds removes the need of input data provided in a structured form (e.g. data attached to a computational mesh). During the training both the decoder and the latent code are optimized simultaneously, thus avoiding the use (and training) of an encoder. In practice, the training of the “auto-decoder” is performed by solving two optimization tasks simulatenously. On one side, the decoder is trained assuming that the input latent code is correct (this corresponds to a standard supervised learning task). At the same time, the latent code is optimized in such a way that it can be explained in the best way possible given the decoder at the current stage of the training. After the training, inference is done by simply optimizing the latent code of the new shape given the trained decoder.

In order to validate the geometric decoder, several test cases were conducted, including geometries not strictly related to the test cases in WP6.

4.3. Deep geometric encoding of gyroids

The first test was performed in order to assess the capability of the auto-decoder to encode the gyroid shapes. The traning dataset consists of the same lattice geometries used in the optimization of the MCHX as described in the deliverable report D6.3.

Validation was performed on a set of 18 lattice geometries, which were never presented to the model during the training stage. For each validation geometry, the encoding (latent code)

of the SDF was firstly computed by latent code optimization. Next, the geometry was decoded to obtain a prediction of the SDF value on a cloud of points randomly sampled within the domain of interest (see *Figure 12*). The prediction of the SDF was then compared with the ground-truth value, i.e. the value of the true SDF computed at the same point locations. The error histogram on some of the validation geometries is reported in *Figure 13*. The average L_2 error between the predicted SDF and the ground-truth value (relative to the characteristic size of the lattice cell) is less than 0.001%, which corresponds to an error (in physical units) of about 5-6 μm for a lattice cell of size of 1 cm using a latent space of size 16. This error could be further reduced by increasing the size of the latent space, i.e., the number of parameters used to encode the lattice cell geometry. However, since the validation error is significantly below the typical deviations from the nominal geometry of an Additive Manufacturing process, the accuracy reached by the present model was considered sufficient for the purposes of this project.

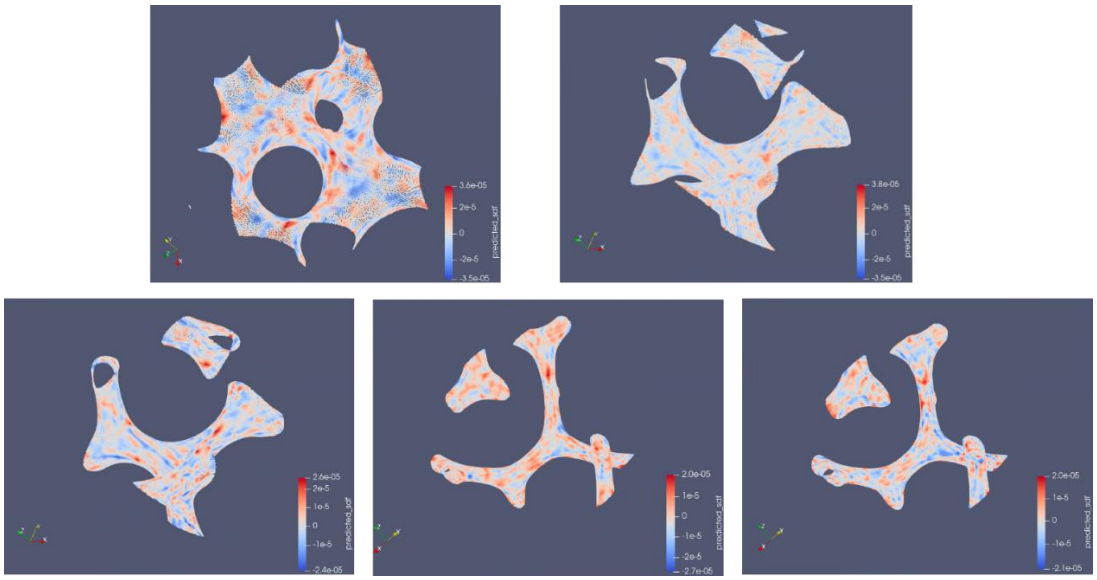

Figure 12: Examples of gyroid geometries encoded by the deep-geometric learning model developed in ST2.1.1. Colormap represents the error from the ground-truth SDF. SDF is normalized by the cell size.

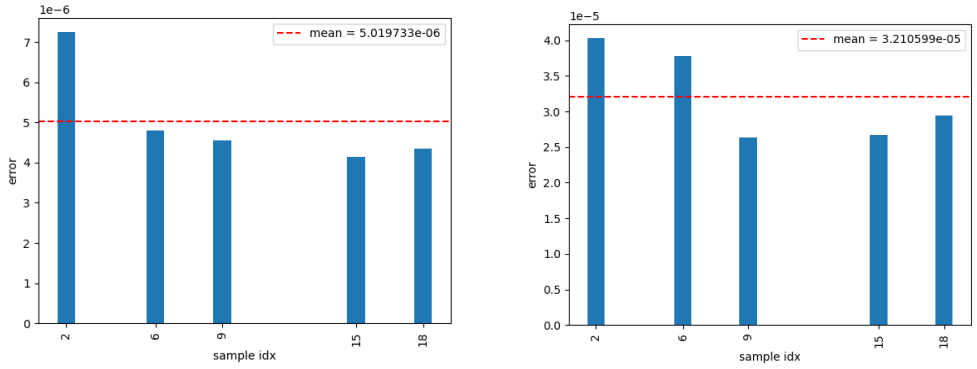

Figure 13: L_2 (left) and L_{inf} (right) error between the predicted SDF and the ground truth value on the validation geometries shown in Figure 12. In all cases, the relative error after the encoding-decoding pipeline is less than 0.001% which corresponds to approximately 5-6 μm for a lattice cell with a characteristic dimension of 1 cm.

-

4.4. Deep geometric encoding of complex geometries.

In order to test the accuracy of the geometric encoding on complex geometries, a second test was performed on a realistic geometry of a car (DRIVAER).

The training dataset was generated starting from 8 varations of the base geometry, each corresponding to a deformation typically encountered in the early design stages of a autovehicle (shown in *Figure 14*).

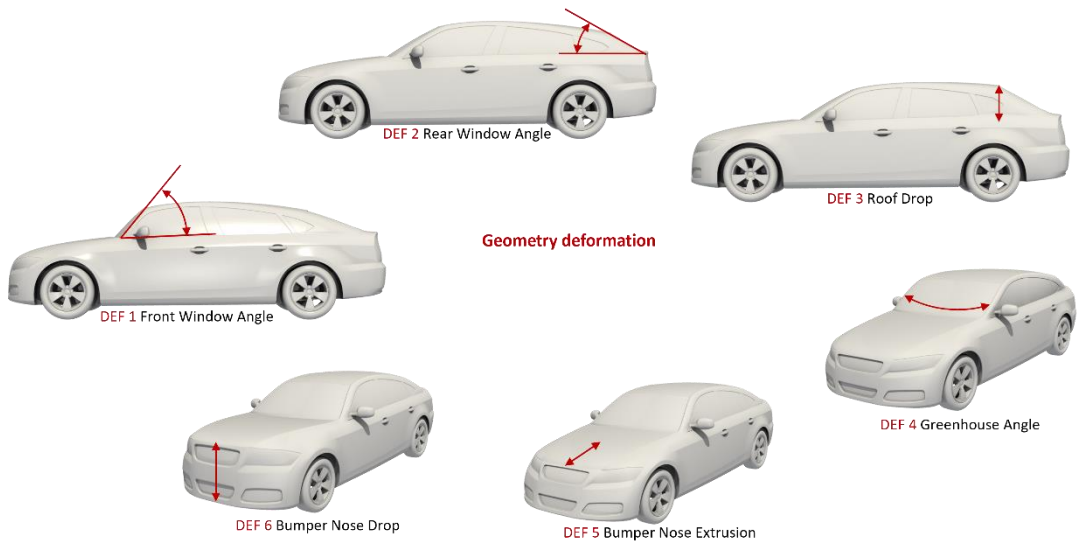

Figure 14: Variations of the base DRIVAER car used to generate the training dataset for the deep geometric encoding of auto-vehicles.

The base deformations shown in *Figure 14* were combined together to generarate a dataset with several variations of the base DRIVAER car. The dataset was used to to train the auto-decoder using a 80/20 training-validation splitting and a latent space size of 16. In all test cases, the accuracy of the encoding is comparable with the previous test case. The relative error between the predicted SDF and the ground-truth is approximately 0.07%, which corresponds to an absolute error (physical units) of 4mm for a car length of approximately 4 m. As can be noted from *Figure 15*, small scale details (such as side mirrors) are reconstructed correctly after the encoding. To our knowledge, such details tend to be blurried out by in more classical techniques based on encoder-decoder approach.

Finally, wall times required for decoder training and latent code optimization are reported in Table 1. Test were conducted on a A10g GPU. Traning times for the deep geometry encoder are in the order of minutes for the tranining of the decoder and order of seconds for latent code optimization. The last row represents the time required to generate the mesh of the reconstructed geometries.

	CPU	NVIDA A10g (24GB VRAM)
Decoder training	~14 h	~17’
Latent code optimization	~550’’ (~9’)	~3’’
Mesh reconstruction	~3475’’ (~58’)	~34’’

Table 1. Wall-clock time required to train decoder (first row) and for latent code optimization (second row) for the DRIVAER test case. For comparison, last row shows the time required for the mesh reconstruction given the prediction of the SDF generated by the auto-decoder.

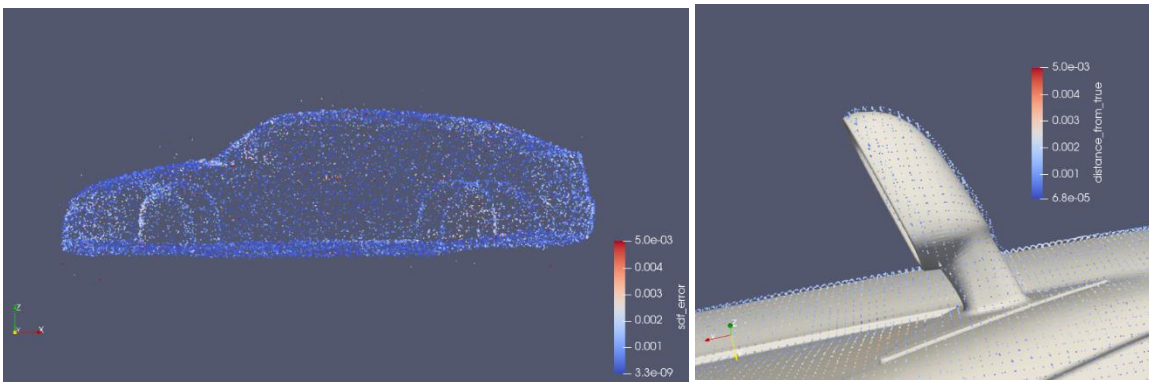

Figure 15: Example of a car geometry reconstructed after the encoding. Small scale details are also reconstructed correctly using the auto-decoder developed in ST2.1.1.

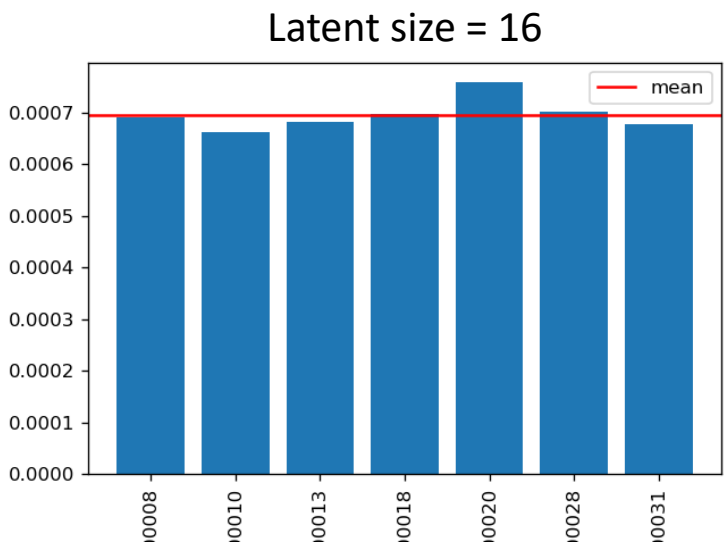

Figure 16: L_2 error (normalized by the car length) between the SDF predicted by the decoder and the ground-truth SDF. The error is below 0.001% corresponding to an encoding error of 4mm for a car length of 4m.

4.5. Deep geometric encoding of noisy gyroids.

In the last test case, the auto-decoder was trained to encode the shape of gyroid cells "polluted" by high frequency perturbations (see *Figure 17*). The goal is to assess the encoding performances in case of input data affected by noise. In practice, high frequency perturbations are introduced by the Additive Manufacturing process and are responsible for significant deviations from the nominal geometry. Also in this case, the geometric encoder, was able to encode the shape with an accuracy below 0.001%.

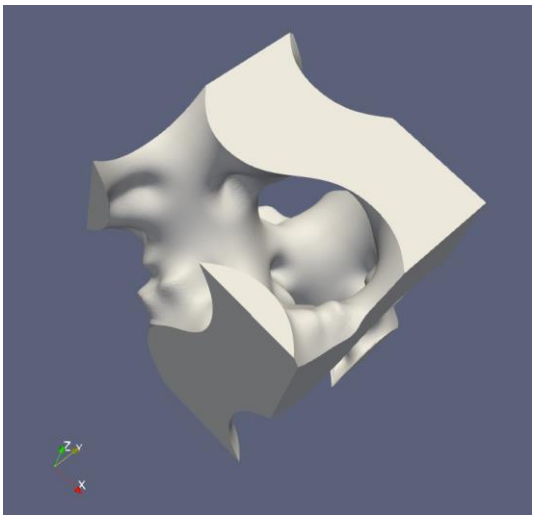

Figure 17: Example of a "bumpy gyroid" in the training dataset. A high frequency disturbance is added to the base gyroid shape to simulate artifacts and deviations from the nominal geometries introduced by the Additive Manufacturing process.

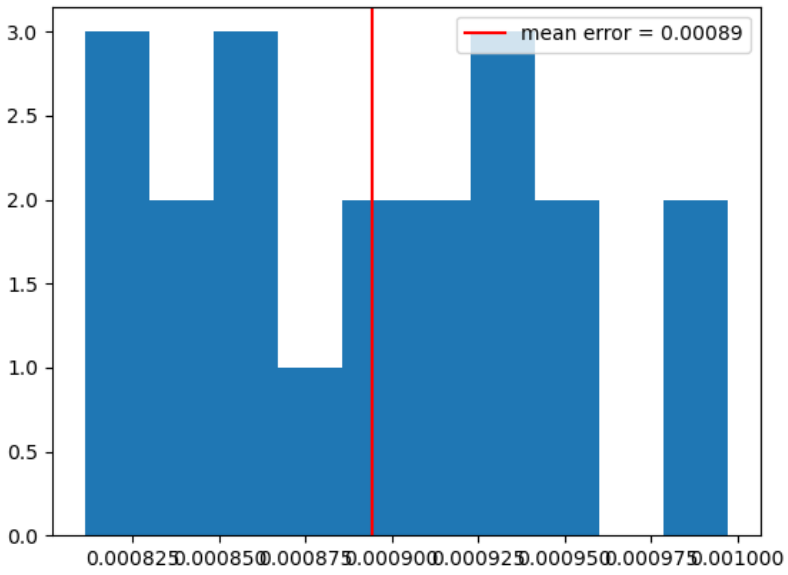

Figure 18: Mean L_2 error (normalized by the cell size) between the SDF predicted by the auto-decoder and the ground-truth SDF. The error is again below 0.001% despite the high-frequency disturbance added to the input geometry.

4.6. Instant flow predictions of CFD fields in gyroids

With the geometric auto-decoder trained, the ML model for instant CFD predictions consists only of the decoder part of the U-Net architecture. Given the lattice geometry, the SDF on z-normal slices is firstly computed and encoded using the geometric auto-decoder. Next, the encoding of the SDF slice is fed to the CFD decoder alongside with the pressure gradient acting on the lattice cell and the index of the slice. Finally, the prediction on each slide are reitreived and packed together to recover the 3D CFD field.

The current version of the model uses a formulation of the UNet architecture applied to 2D image-to-image translation problems. For this reason, the input geometry is processed in slices. Although there is a version of the UNet architecture capable of processing 3D images, the numerical results obtained during the model tuning show slightly better prediction accuracy for the 2D architecture.

The resulting workflow is schematically shown in *Figure 19*.

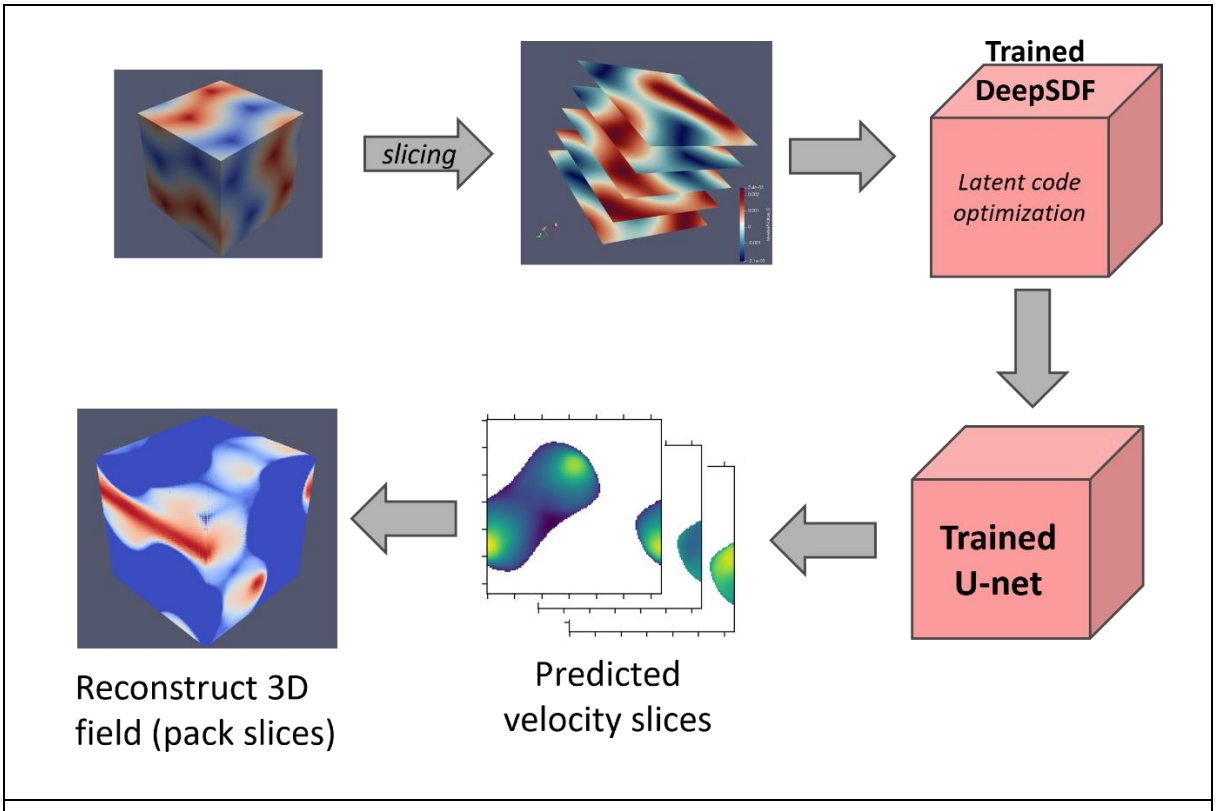

Figure 19: Schematic depiction of the instant flow prediction workflow. The SDF of the input geometry (local lattice cell) is firstly encoded "by-slice" using the auto-decoder approach described in the previous sections. The latent code, alongside the pressure gradient acting on the cell is fed to the U-net decoder to provide predictions of CFD fields on each slice of the domain. Slices are finally packed together to obtain the 3D fields of interest.

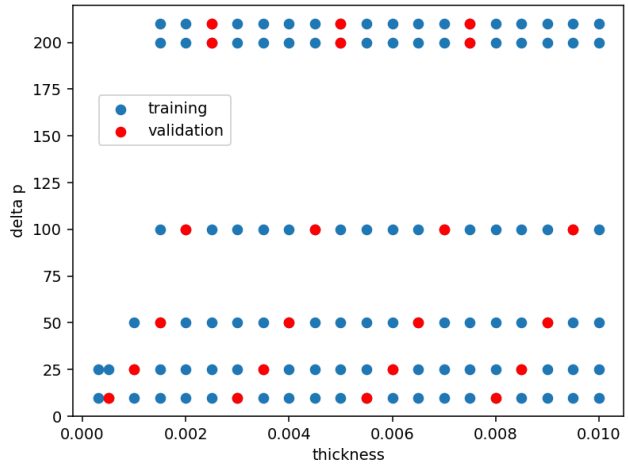

Figure 20: Training and validation datasets of high fidelity CFD simulations used to train the ML model for instant flow predictions. Each point in the dataset represents a high-fidelity CFD simulation performed on a gyroid cell with varying geometry pressure drop acting on the cell.

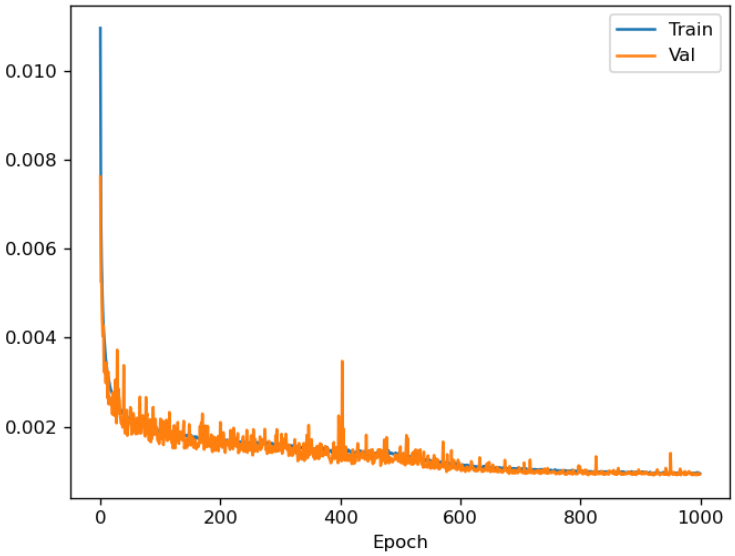

Figure 21: Training and validation losses for the training of the U-Net model for instant CFD prediction in gyroid channels.

The ML for CFD predictions was trained using the same dataset of CFD simulations presented in the deliverable report, D6.3. After the training, validation was performed on a set of geometries and operating ranges never presernted to the model during the training phase (red point in *Figure 20*). For each validation geometry, the prediction of the fluid dynamic fields generated by the model was compared with the result of high-fidelity simulations computed in OpenFoam. *Figure 22*, shows the error histogram on the validation set, demonstrating the excellent predictive capability of the ML model. The average relative error is around 4% (relative to the average flow velocity in the micro-channel) on the set of

validation geometries and operating ranges. This error is compatible with the error obtained during the mesh dependency study which was around 6%.

A demonstrator showcasing the capabilities of the approach developed in this project will be made available on a public GitHub repository: https://github.com/optimad/DeepGeoFastCFD This demonstrator will provide users with a hands-on example of instant CFD predictions using geometric deep learning, allowing them to explore its features, performance, and potential applications. By accessing the repository, users can gain insights into the implementation details, experiment with the code, and test/train the models on their own datasets.

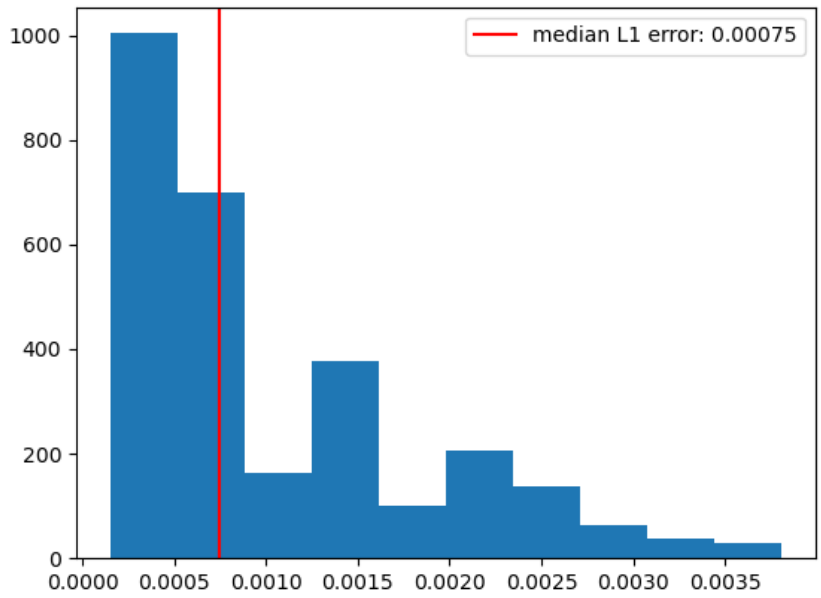

Figure 22: Histogram of error on CFD prediction on validation geometries. The median L1 error is around 7.5 10^{-4} m/s which corresponds to an average relative error of approximatively 4% (relative to the average flow velocity in the micro-channel). Note that the vast majority of the predictions are clustered around the median value, and few outliers exhibiting a prediction error around 10%.

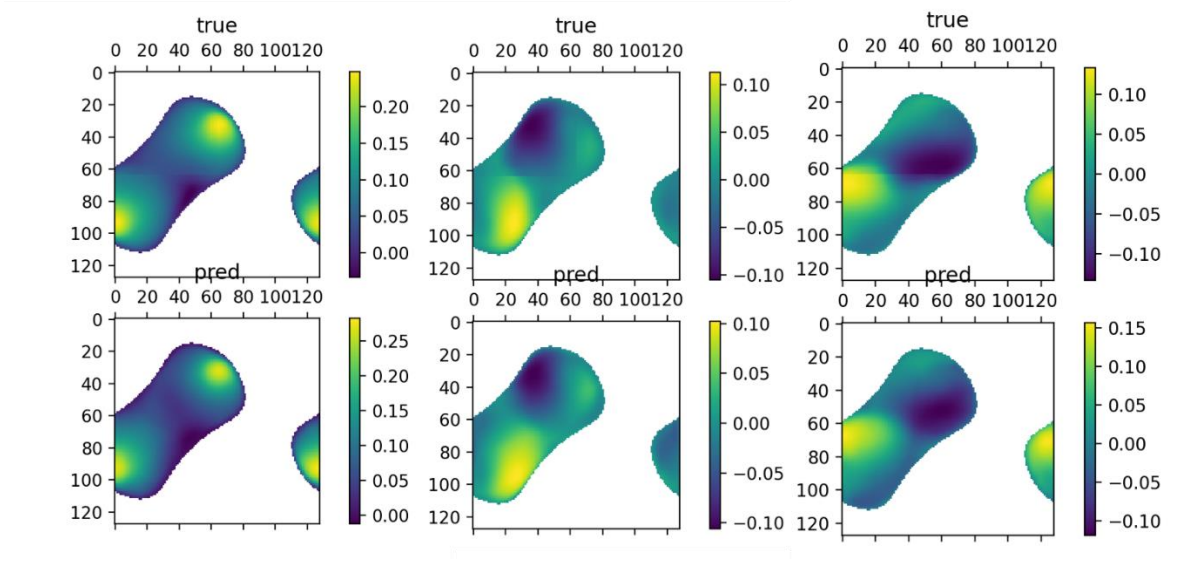

Figure 23: Example of CFD prediction of the velocity field on a slice a lattice cell. Upper row: high fidelity CFD results (ground truth). Bottom row: instant prediction of the velocity field obtained with the ML model developed in ST2.2.1.

5. Airfoil flow prediction by DNN (ICL)

In this subsection, Imperial College explored the use of a deep neural network (DNN) to predict the flow around transonic airfoils in freestream. Using an existing open-source repository (GitHub - tum-pbs/coord-trans-encoding: This is the source code for our paper "Towards high-accuracy deep learning inference of compressible turbulent flows over aerofoils") based on the work of Chen and Thuerey (2023) [1], we demonstrate the working of this code by making suitable adaptations to work on CPU clusters, together with determining the sensitivities of the hyper-parameters of the neural network. Results are presented to showcase the error of computation, besides computing the L1 norm. The pertinent results with regards to the airfoils in freestream – such as u,v velocities, skin friction coefficient and pressure coefficient can be predicted from this code. The following paragraphs briefly detail the architecture of the neural network, and we present a few results from the testing of the dataset.

The demonstrator closely follows the approach used by Chen and Thuerey from TU Munich, who have performed the CFD analysis of airfoils in the transonic regime. Utilising the UIUC database, the coordinates of the airfoils are obtained, and a structured mesh around the airfoils was constructed. The boundaries of the domain are located 20 chord lengths from the airfoils in the upstream, lateral and downstream directions. A univalent transformation of the x-y coordinates of the grid points to curvilinear coordinates was performed to obtain a square grid of 128x128. The CFD simulations were performed in CFL3D v6.7 using the Spalart-Allmaras turbulence model. The free-stream condition for a ground truth simulations was randomly selected from a range of Mach numbers $M\propto \in$ [0.55, 0.8], Reynolds numbers $Re\propto \in$ [0.5, 5] million, and angles of attack [−0.5°, 8.0°]. The number of degrees of freedom for the neural network was 128x128x5 (with the flow variables, density, u,v velocities, speed of sound and the eddy viscosity from the turbulence closure).

The DNN uses the conventional convolutional neural neural network (CNN) architecture – U-Net, which involves the encoder and decoder parts connected by skip connections as seen in Figure 24. For the training, validation and testing, 9300, 400 and 20 datasets are used,

respectively. The resulting errors are computed using the L1 norm between the ground truths and the predicted flow fields.

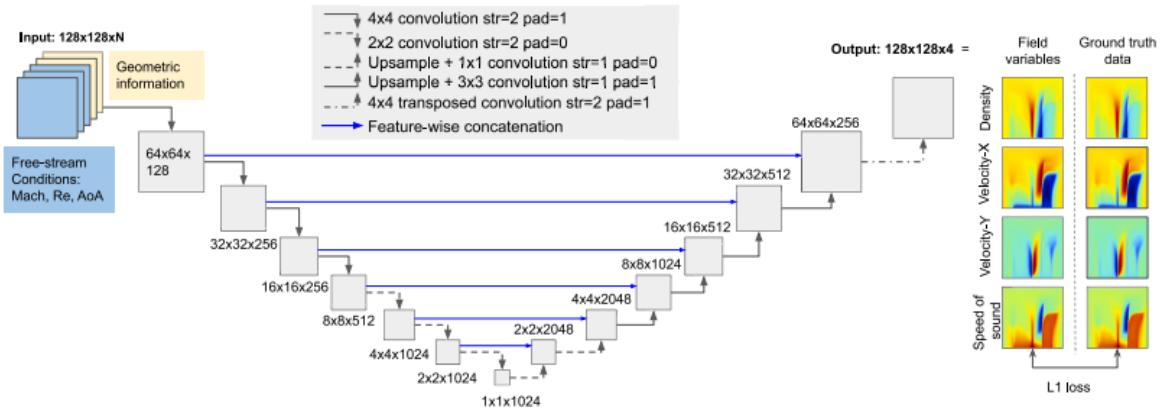

Figure 24: Architecture of the U-Net used in this demonstrator. Image obtained from Chen and Thuerey [1], which shows the convolution operations followed by downsampling until the bottleneck layer (very bottom) and followed by upsampling operations and transpose convolution operations. The skip connections in blue show the concatenation at each level.

The code was adapted to run on the ASRC Orthrus CPU cluster on a single node (112 cores) using Python3.11 and Pytorch. Modifications were made to the generation of output images, to obtain the ground truth, prediction and the errors in the flow. Additional modifications were made to the plotting of the skin friction and pressure coefficient plots as seen in Figure 25, for the open-source airfoil ah63k127.

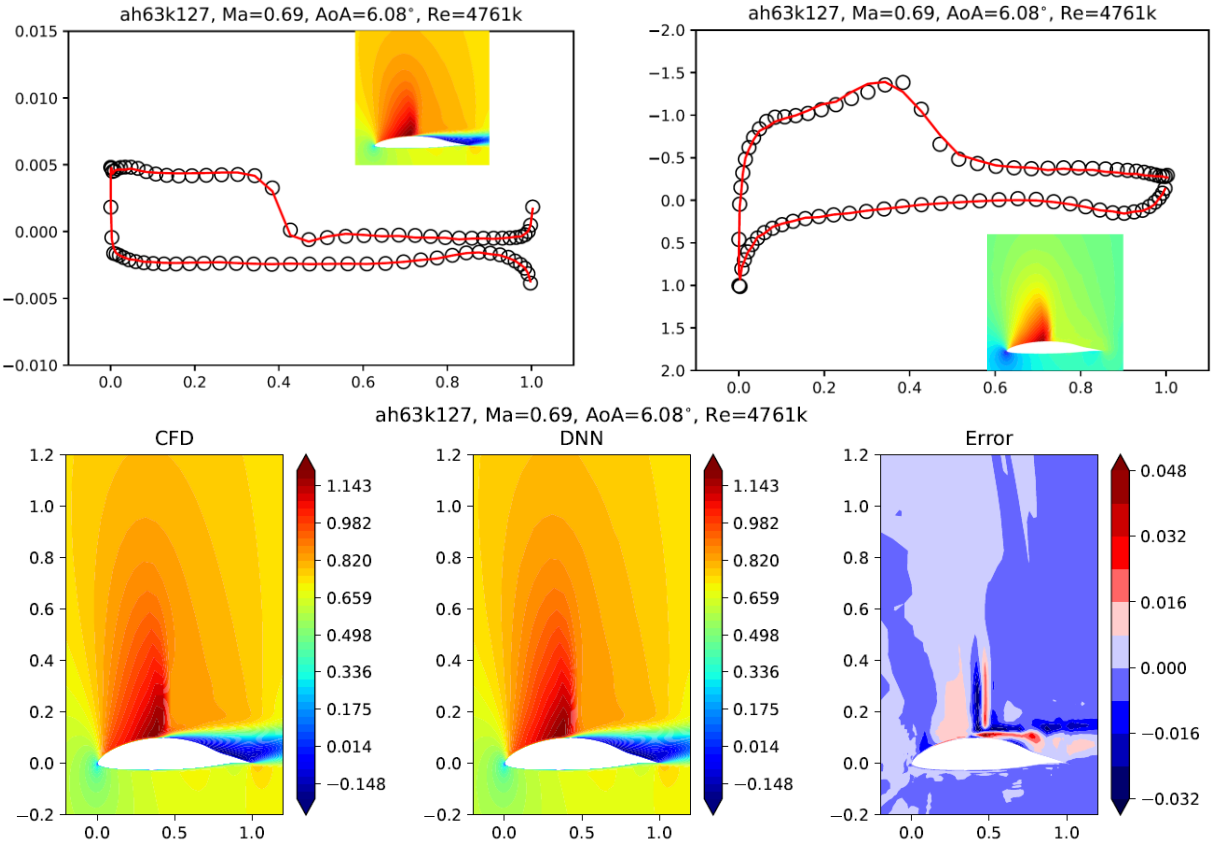

Figure 25: Top left: Skin friction coefficient, top right: Pressure coefficient. Open symbols – predictions; red lines – ground truth. Bottom. Contours of u-velocity for the CFD, DNN and the error.

The contour plot of the normalised velocities shows the relatively close match between the CFD and predictions, with the error occurring close to the vicinity of the shock and the separated regions.

The impact of the network hyperparameters on the L1 norm are investigated as seen in Figure 26. For a constant learning rate (LR = 1e-4), decreasing the batch size provides a lower error rate. A drop in the L1 norm is observed after 2000 epochs even for a larger value of the LR = 1e-3. The L1 norms for are typically lower when the batch sizes are smaller and the minimum number of epochs required for lower values of L1 norm would exceed 3000-4000 epochs which take about 2 days of training.

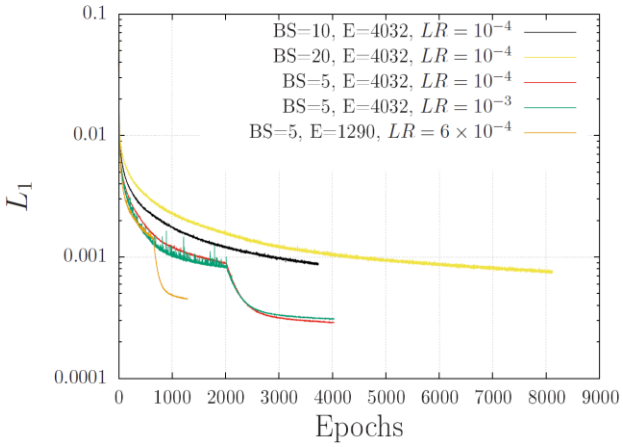

Figure 26: Variation of the L1 norm with batch size, number of epochs and learning rate. The last case with 1290 epochs and a batch size of 5 was used by the authors.

Conclusion:

1. The numerical code from the open-source repository has been modified for CPU usage on a cluster. The code has been validated for the prediction of the airfoils in the transonic regime of the flow.
2. Further modifications include the outputs from the code which now showcase the CFD solution, the DNN prediction and the error. Furthermore, the flow field of the airfoil has been shown in the plot for the comparison of the pressure coefficient, skin friction.
3. Lastly, we have investigated the sensitivity of the L1 norm metric to the changes in the hyperparameters of the network.

References:

[1] Li-Wei Chen and Nils Thuerey, *Towards high-accuracy deep learning inference of compressible flows over aerofoils*, Computers & Fluids, Volume 250, 2023, 105707, ISSN 0045-7930,https://doi.org/10.1016/j.compfluid.2022.105707.