The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-9 June 2022, Oslo, Norway

CORRECTING THE DISCRETIZATION ERROR OF COARSE GRID CFD SIMULATIONS WITH MACHINE LEARNING ECCOMAS CONGRESS 2022

A. Kiener[1], S. Langer[2] and P. Bekemeyer[3]

[1,2] German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, $C^2A^2S^2E$
Lilienthalplatz 7, 38108 Braunschweig, Germany

[1] anna.kiener@dlr.de, [2] stefan.langer@dlr.de, [3] philipp.bekemeyer@dlr.de

Key words: Discretization Error, Coarse Grid, Computational Fluid Dynamics, Random Forest, Neural Network, Graph Neural Network

Abstract. Computational fluid dynamics is a cornerstone for the modern aerospace industry, providing important insights on aerodynamic analysis while reducing the need of expensive experiments and tests. Nevertheless, simulations of complex geometries are often performed on a discrete spatial domain too coarse to capture all relevant physical phenomena for the sake of lowering the computational cost. A consistent spatial discretization on so-called grids approximates the analytical solution of the partial differential equation with increasing number of discrete points. Such a grid refinement study is an expensive method to assess the general grid quality. This work shows that machine learning as a post-processing tool is capable of improving coarse grid simulations, even if not converged. The results show that three machine learning models varying in their complexity, namely the random forest, the neural network, and the graph neural network, are capable of finding patterns in coarse grid simulations. These patterns are used to predict the discretization error to approximate the field variables of interest of the corresponding fine grid simulation mapped onto the coarse grid. Initial training and testing is performed on the RAE2822 airfoil leading to corrected flow fields, improved surface integrals and coefficients, even when shocks are present. Additional tests are performed on the RAE5212 airfoil, showing the generalization limits of the trained models. The proposed method promises to reduce computational expenses while increasing the accuracy of the coarse grid results which works locally, e.g. it corrects the error for each cell individually and is therefore not restricted by the number of grid points. The presented results obtained by the machine learning models during post-processing are a promising baseline for more integrated developments, where the models will interact in a dynamic fashion with the flow solver to further improve coarse grid simulations.

1 INTRODUCTION

Aerodynamic analysis based on computational fluid dynamics (CFD) has become a cornerstone for the aerospace industry, whether it be for design or optimization. High-fidelity simu-

lations provide insight into highly complex physical phenomena without the need for expensive wind tunnel experiments or flight tests. Nevertheless, these methods are far from perfect.

One of the crucial aspects when performing such simulations is the need for a spatial discretization on a computational grid to approximate solutions of the governing equations of interest. The design of such grids has in general a significant influence on the accuracy of the approximate solutions. Meshes with a high grid resolution are necessary to accurately capture phenomena such as the effects of turbulence at high Reynolds numbers, where the number of degrees of freedom has to increase close to the boundary layer to correctly approximate steep gradients.

Such a fine-grained resolution significantly increases computational cost as well as simulation time and ultimately becomes a bottleneck of many high-fidelity simulations. It is best practice to conduct grid convergence studies to quantify the error induced by the spatial discretization allowing to find an optimal trade-off between cost and accuracy. However, obtaining such spatial consistent results for full aircraft configurations at flight Reynolds numbers is often regarded infeasible given an industrial setting, where it is not uncommon to have grids with several hundred million points which are still too coarse [1]. Under these conditions, it is often impossible to conduct rigorous grid convergence studies to estimate the error. In the past, several approaches have been proposed to circumvent this particular shortcoming, with adaptive grid refinement being a prominent one [2]. Although promising, especially for resolving discontinuities, the implementation of a complex adaptive mesh refinement algorithm can be daunting [3] and the needed computational time to solve the equations of interest still rises non-linearly with the total number of cells in the given grid. Lately, due to an increasing availability of hardware and respective algorithms, Machine Learning (ML) based methods have become an attractive alternative to be discovered to leverage CFD simulations [4, 5, 6, 7]. The field of ML promises to alleviate the computational cost of simulations by taking advantage of the abundance of data available from testing and simulation.

Previous research has focused on the use of Convolutional Neural Networks (CNN) to reconstruct high quality CFD data based on simulation data of poor quality. The main reason for using CNNs has been their success in the field of computer vision for the so-called super-resolution problem, where their implementation showed the ability to map the input of a low-resolution image to a highly resolved output [8]. The main advantage of CNNs is their use of filters reducing the dimensionality of the data. These filters are directly applicable to structured data, encountered in uniform data grids such as images. Fukami et al. [9] used direct numerical simulation (DNS) of turbulent flow-fields as high-resolution data and created the low-resolution equivalent via downsampling. Latter was used in a supervised setting as input to a model based on a CNN to reconstruct the DNS data. Jiang et al. [10] propose a convolutional decoder as a variation of the U-Net architecture with subsequent use of a multi-layer perceptron. Latter allows to sample the super-resolved solution at any location. Kochkov et al. [11] showed substantial speed-ups for DNS and LES simulations while applying varying CNN models to two-dimensional turbulent flow fields in a tightly coupled manner with a CFD solver. Um et al. [12] also follow a coupled approach, where their differentiable physics solver interacts with varying ML models based on fully convolutional architectures. Instead of super-resolving, they focus on learning a correction function which is applied to the coarse grid data with the aim to reduce the numerical error arising from the discretization in time and space of the PDEs to be solved. Hanna et al. [13]

used a random forest and a dense neural network model to predict the error due to the spatial discretization. Their focus was the quantification of the discretization error after the coarse simulations were performed without any coupling of the ML model and the solver.

The work of this paper will follow-up on the idea of using ML methods as a post-processing tool. The main goal is to identify how the error induced by the discrete spatial domain is quantifiable and subsequently if this error exhibits patterns which can be learned. Classical CNN models are avoided, since unstructured grids are common to model complex geometries in flow simulations, which hinders their straight-forward use on CFD data [7, 14, 15]. Instead, the use of graph convolutional neural networks is proposed, along two less complex models which are a random forest and a deep neural network. For all CFD computations, the Navier-Stokes equations in conjunction with a one-equation turbulence model are solved for different airfoils at varying Mach numbers and angles of attack.

2 THEORETICAL BACKGROUND

2.1 Governing Equations

The Reynolds Averaged Navier-Stokes (RANS) equations are on a given domain Ω and its closed surface $\partial\Omega$

$$\frac{d}{dt}\int_{\Omega} W d\Omega + \oint_{\partial\Omega} (F_c - F_v) dS = 0 \tag{1}$$

with the conservative variables being $W = (\rho, \rho u, \rho v, \rho w, \rho E)^T$. F_c and F_v are the vector of convective and viscous fluxes, respectively. In this work, turbulence is modelled with the negative Spalart-Allmaras one-equation model [16]. The integral form of the governing equation for the turbulent transport variable $\tilde{\nu}$ with $Q_{\tilde{\nu}}$ denoting the source term is

$$\frac{d}{dt}\int_{\Omega} \tilde{\nu} d\Omega + \oint_{\partial\Omega} (F_{c,\tilde{\nu}} - F_{v,\tilde{\nu}}) dS = \int_{\Omega} Q_{\tilde{\nu}} d\Omega \quad . \tag{2}$$

2.2 Theory on Machine Learning

2.2.1 Training and Prediction Phase

Given a set of training instances $\{(x_i, y_i)\}_{i=1}^{N}$, a supervised learning algorithm for a regression task searches a mapping function $f : X \to Y$ between the provided feature vector x_i and the continuous output y_i. In this study, the output y_i and a training instance i correspond to the discretization error $\varepsilon_{\phi,i}$ and one grid point, respectively.

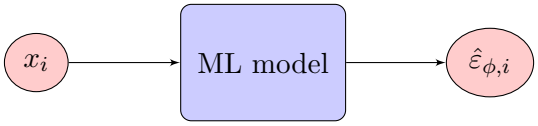


Figure 1: Prediction phase

After the training, the ML model is able to predict for previously unseen values of features x_i an approximation $\hat{\varepsilon}_{\phi,i}$ of the true output variable $\varepsilon_{\phi,i}$, such that $f(x_i) = \hat{\varepsilon}_{\phi,i} \approx \varepsilon_{\phi,i}$, as given in figure 1. Here, ϕ denotes one of the field variables of interest to be corrected, e.g.

velocity in x- and z-direction, pressure, density and the turbulence model variable, such that $\phi = \{u, w, p, \rho, \tilde{\nu}\}$. To learn this function, three models with varying complexity are used and compared, namely a random forest, a neural network and a graph neural network.

Random Forest One of the three models employed in this work is the Random Forest (RF) algorithm developed by Breiman [17]. It belongs to the class of ensemble algorithms. Ensemble models seek to improve the predictive performance by combining several weak learners. For a RF regression model, the results of many decision trees for a regression task are averaged to arrive at the final prediction. Let the prediction of a single decision tree be $f_i(x_i) = \hat{y}_{DT,i}$, then the RF prediction comprising N_{DT} decision trees is

$$\hat{y} = \frac{1}{N_{DT}} \sum_{i=1}^{N_{DT}} \hat{y}_{DT,i} \tag{3}$$

Neural Network Secondly, a fully connected deep Neural Network (NN) is implemented. A NN consists of multiple layers of so-called neurons. Each layer can be described as a regression function $f_l(x) = \hat{y}_l$, where x is the input to each layer l and $\hat{y}$ its respective output. Each regression layer can be written as

$$f_l(x) = g_l(\mathbf{W}_l x + b_l) \tag{4}$$

where g_l denotes the activation function introducing a non-linearity to the equation. The weight matrix $\mathbf{W}_l$ and the bias vector b_l are the parameters to be optimized during the training phase. This is done with the backpropagation algorithm, which uses the chain-rule to compute gradients of the chosen loss function after each training epoch and accordingly updates weights and biases in order to minimize the loss function.

Graph Neural Network As final model, the Graph Neural Network (GNN) is presented. This model has been proposed as an extension of NNs for graph structured or non-Euclidean data [18]. Its advantage lies in its applicability to graphs $\mathcal{G}$ of arbitrary structure and its permutational invariance to the number of input nodes $\mathcal{V}$ and edges $\mathcal{E}$. GNNs take into account information of neighbouring data points as well as the connectivity structure between all data points, thus maximizing the usable information content of a given data set. The function to describe a layer is written as

$$\mathbf{H}_{l+1} = f(\mathbf{H}_l, \mathbf{A}) \tag{5}$$

where the input features correspond to the first layer $\mathbf{H}_{l=0}$ and $\mathbf{A}$ denotes the adjacency matrix of the graph, which is needed to describe the structure of a graph. Different layer implementations differ in their definition of the function f on the right hand side of equation 5. For this work the node-level problem is solved, e.g. a regression task at each node $\mathcal{V}$ of the given graph $\mathcal{G}$. Furthermore, the graph convolutional operator proposed by Kipf et al. [19] is employed. By introducing two simplifications to a spectral graph convolutional framework, their work promises improvement in training times and prediction accuracy. Firstly, the identity matrix $\mathbf{I}$ is added to $\mathbf{A}$, such that $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. This allows to not only account for features of neighbouring nodes but also of the given node itself. Secondly, to avoid a change of scale of the features, a symmetric

normalization of $\tilde{\mathbf{A}}$ is introduced as $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}$. Here, $\tilde{\mathbf{D}}$ is the diagonal node degree matrix of $\tilde{\mathbf{A}}$. Thus, their proposed propagation rule for each layer of an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given as

$$f(\mathbf{H}_l, \mathbf{A}) = g_l(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}\mathbf{H}_l\mathbf{W}_l) \tag{6}$$

where g_l denotes a non-linear activation function such as for the NN layer in equation 4.

2.2.2 Validation Metric

Various validation metrics exist to assess the quality of a model. In this work the coefficient of determination R^2 is used (equation 7). $\overline{y}$ denotes the mean of the set of output variables $\{y_i\}_{i=1}^N$. A model predicting the exact output y will reach a coefficient of determination $R^2 = 1$, whereas $R^2 = 0$ corresponds to a model predicting the average $\overline{y}$ for any given instance.

$$R^2(\hat{y}_i) = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \overline{y})^2} \tag{7}$$

2.2.3 Hyperparameter Optimization

The hyperparameters h of a given ML model can be optimized with respect to its performance on a validation data set evaluated for an objective function $J(h)$. In this work, the objective is to maximize the chosen validation metric, e.g. the coefficient of determination R^2

$$h_{opt} = \arg\max_{h \in \mathcal{H}} J(h) = \arg\max_{h \in \mathcal{H}} R^2(h) \tag{8}$$

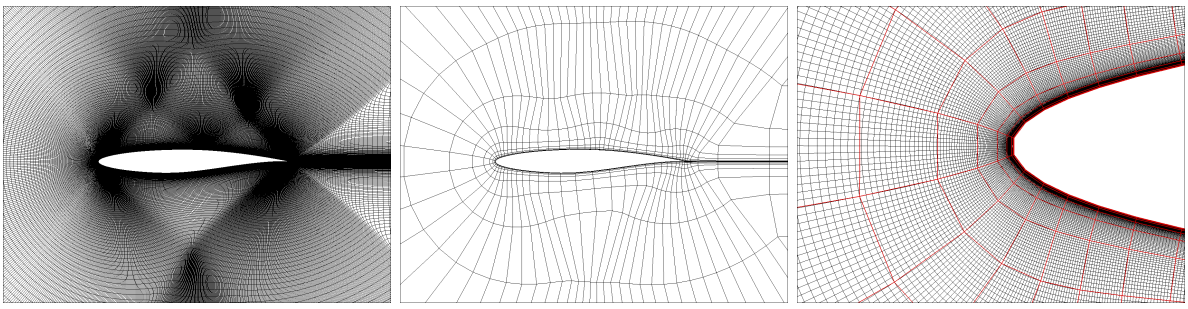
Beside tedious manual hyperparameter tuning, common strategies for hyperparameter optimization are grid and random search, in which all possible combinations or a random set of combinations are explored for a given set of hyperparameters. A more effective strategy is Bayesian optimization, where a probabilistic surrogate model explores only the promising solution space based on priors. The subsequent hyperparameters are chosen in an informed manner, by taking into account already explored trials. In this work a sampler using a Tree-structured Parzen Estimator algorithm [20] is used.

For the RF model, the number of decision tree estimators, the minimum samples at the leaf nodes, the maximum number of features for a split and the bootstrapping parameter are tuned. The hyperparameters to be tuned for the NN are the number of hidden layers, the number of neurons for each layer, the batch size, activation function, optimizer and the learning rate with and without exponential decrease. The best optimizer and activation function found during the NN tuning were then employed for the GNN. Only the number of stacked convolutional layers and their feature dimensionality as well as the learning rate and its exponential decay rate are tuned for the GNN.

3 APPLICATION CASE

3.1 RAE2822 Simulation

This paper presents results for the turbulent flow over the two-dimensional transonic airfoil RAE2822. The DLR-CFD simulation software TAU [21] is used, which employs a vertex-centered scheme. The high-resolution (HR) and low-resolution (LR) solutions are obtained on structured C-type grids with $1280 \times 256 = 327'680$ and $80 \times 16 = 1'280$ cells, respectively (2a and 2b). Hence, the resolution factor between both grids is 256. The two grids are nested grids, since the coarse grid is obtained via agglomeration of the HR grid, as can be seen in figure 2c. A detailed study on this grid sequence was conducted by Langer [22].



(a) HR grid 1280×256 (b) LR grid 80×16 (c) Detail of the grids

Figure 2: Coarse and fine grid

Steady-state simulations are performed for the Mach numbers M={0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7} and various angles of attack, summing up to a total of 144 simulations (see appendix 5.1 for further details). For high angles of attack not all simulations converged on the LR grid, e.g. their residuals showed an oscillating behaviour after a reduction of only two to four orders of magnitude. The Courant-Friedrichs-Lewy number is constant across all coarse simulations to ensure consistency between these non-converged LR solutions. Moreover, for non-converged flow simulations the solutions were obtained by averaging the results over the last 3000 iterations. For all other simulations on the coarse and fine grid, the convergence criterion was reached, meaning that a reduction of the density residual of at least ten orders of magnitude has been obtained.

3.2 Mapping Operator

The HR solution is not directly used as ground truth for the training. Instead, the variables of interest of the HR solution ϕ_{HR} are mapped onto the coarse grid

$$\phi_{mapped} = \mathcal{T}\phi_{HR} \tag{9}$$

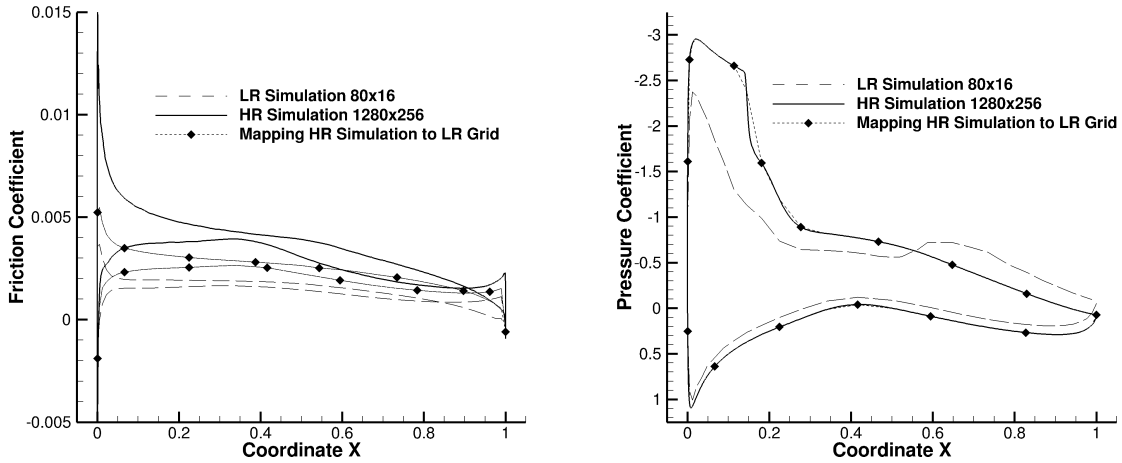
where $\mathcal{T}$ denotes a mapping function. This mapping allows to quantify the discretization error ε for each variable ϕ at each vertex i of the coarse grid as

$$\varepsilon_{\phi,i} = \phi_{mapped,i} - \phi_{LR,i} \tag{10}$$

where ϕ_{LR} denotes the variables of interest of the LR solution. This mapping poses the following limitations to the proposed procedure: a) The choice of the mapping function $\mathcal{T}$ is non-trivial, especially around curvatures when the vertices of the LR and HR grids are not overlapping if non-nested grids are used and b) by reducing the number of data points it naturally decreases the accuracy of the HR solution. Since for the investigated test case the LR vertices are coinciding with their corresponding HR vertices, the nearest neighbour interpolation method is a natural choice as a mapping function $\mathcal{T}$. Thus, the first mentioned limitation can be avoided.

Figures 3 to 4 show different comparisons of computed coefficients of the LR and HR simulation solution as well as the HR solution field variables mapped to the coarse grid. Looking at the friction coefficient in figure 3a and the viscous drag coefficient in figure 4c resulting from the mapping, it can be observed that the coarse boundary layer cells are not resolved enough to capture the same velocity gradient as the HR solution. Hence, the friction coefficient and the viscous drag coefficient are in general lower than for the HR solution. Nonetheless, it is assumed for this study that the coarse grid has enough degrees of freedom to capture the relevant physical phenomena of the HR solution after the mapping. Therefore, the mapped solution is taken as ground truth for the ML model training and evaluation. After the prediction of the discretization error $\hat{\varepsilon}_{\phi,i}$ by the ML model, this value is used to correct the flow field of the LR solution computed on the coarse grid as:

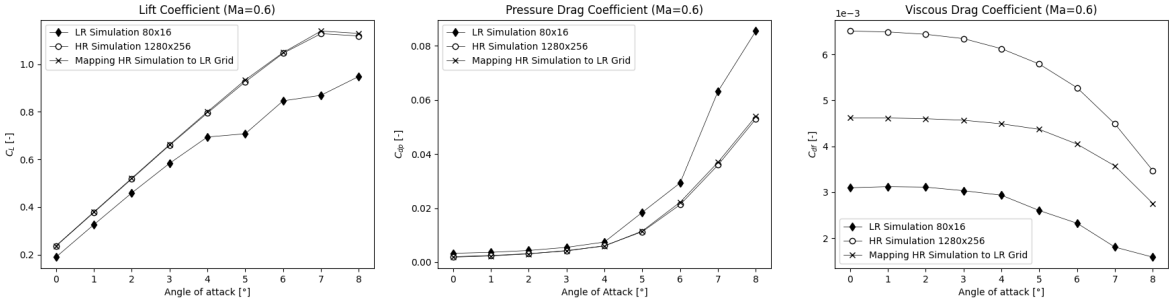
$$\phi_{LR,i} + \hat{\varepsilon}_{\phi,i} = \hat{\phi}_{mapped,i} \approx \mathcal{T}\phi_{HR,i} \tag{11}$$



(a) Friction coefficient (M=0.5, AoA=3.0)

(b) Pressure coefficient (M=0.6, AoA=7.0)

Figure 3: Comparison of surface coefficients



(a) C_L (b) C_{dp} (c) C_{df}

Figure 4: Coefficients of LR, HR and mapped solution (M=0.6)

3.3 Feature Selection and Data Pre-Processing

Since it is assumed that the discretization error correlates to steep gradients within a flow solution, the first and second derivatives of the variables of interest in each dimension are a natural choice as input features for the ML model [13]. The gradients are computed using a weighted least squares approach with an inverse distance weight as described in [23]. The local cell Reynolds number $Re_i = \frac{|U_i| d_{w,i}}{\nu_i}$ is another feature using the wall distance d_w as reference length. There is an additional need to quantify the state of convergence of the LR data, since a set of non-converged simulations is part of the LR database. Hence, the residuals of the density, the velocity, the energy and the turbulent variable are chosen as features. As a global quantification of the uncertainty induced by the non-converged simulations, the final density residual of the simulation is an additional feature. As data pre-processing, the Yeo-Johnson power transformation [24] is applied to each feature as well as to each output variable to stabilize variances and therefore make the data more Gaussian-like. Additionally, the data sets are standardized to zero mean and unit variance.

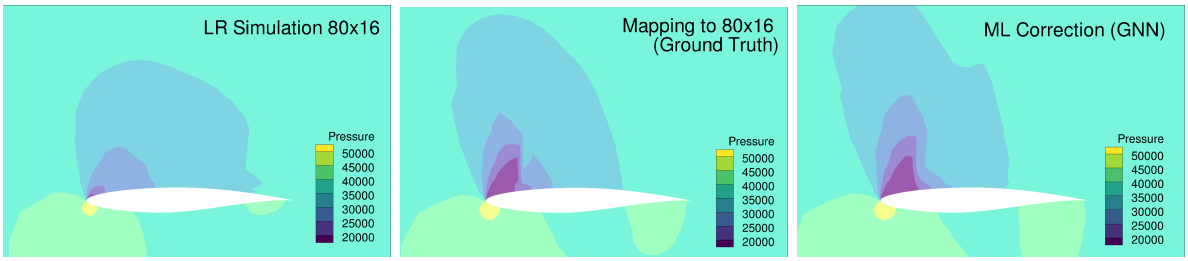
3.4 Training and Validation Strategy

The simulations at Mach 0.6 were set aside to serve as a test set, since they are of special interest exhibiting a shock at higher angles of attack. For all ML models a 8-fold cross validation strategy was employed, where one set of the remaining simulations at a certain Mach number were used as validation set. Thus in total, there are 135 simulations set aside for training and validation and 9 simulations for testing. For the implementation of the RF the Scikit-learn library [25] and for the NN model the SMARTy library [26] with the PyTorch backend [27] were used. The pytorch geometric library [28] was employed for the GNN model. For the hyperparameter tuning, the optuna toolbox is used [29] with a TPE sampler and a median pruning strategy. Details on the final hyperparameters can be found in the appendix 5.2.

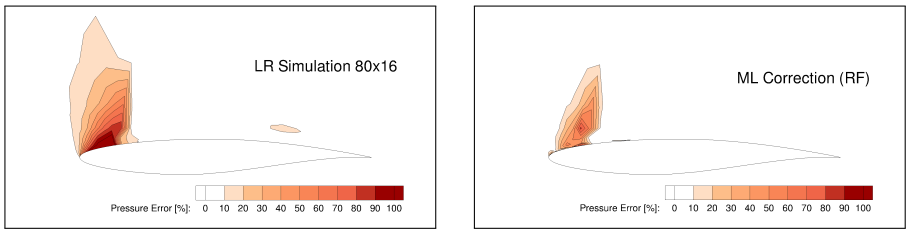
3.5 Test Case Results

This section presents results for the test set of the RAE2822 at M=0.6. Since the correction of the flow field is the main focus of the proposed method, firstly a qualitative comparison of

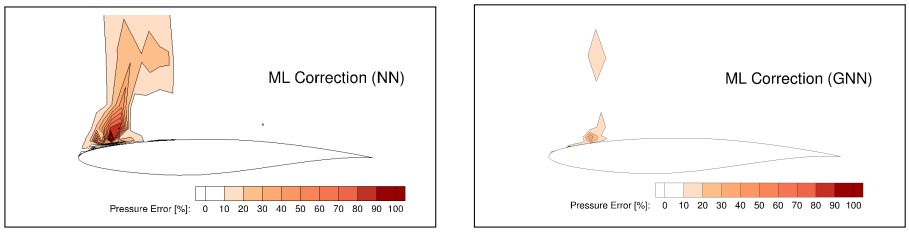
the pressure field is given in figure 5. It presents the percentage error of the pressure at Mach 0.6 at angle of attack 8.0 before and after the ML correction. Looking at the ground truth in figure 5b, it is visible that the flow field is not smoothly captured by the coarse grid after the mapping. Nevertheless, the mapping to the coarse grid is able to capture the shock resolved on the fine grid, whereas the stand-alone simulation on the coarse grid does not feature such a non-linearity (5a). This is reflected in the percentage error given in figure 5d, which shows that the error is mainly present around and upstream of the shock location. Figures 5e-5g show the correction of the coarse simulation for the ML models. All three models are able to reduce the error at the leading edge, thus predicting a shock where the coarse grid simulation failed to do so. All models succeed in reducing the error, with the performance of the GNN being the best. For the RF model the maximum error is found at 87.6 % with a still comparable distribution to the coarse grid solution. The GNN model however decreases the error to a maximum of 37.9 % and deviations from the ground truth only remain in close proximity to the shock location. Similar field correction trends have been observed for the other variables.



(a) Pressure LR solution (b) Pressure ground truth (c) GNN correction



(d) Pressure error LR simulation (e) RF correction

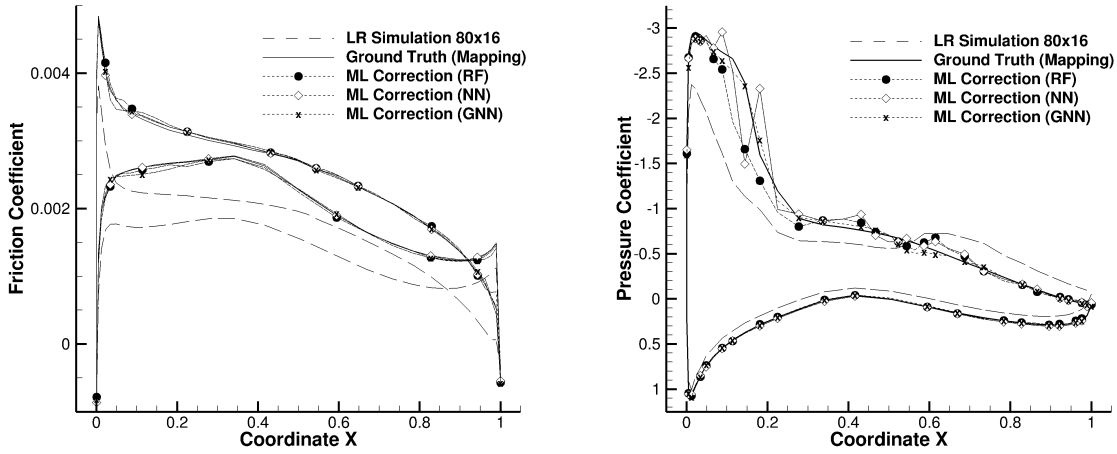


(f) NN correction (g) GNN correction

Figure 5: Pressure error before and after ML correction

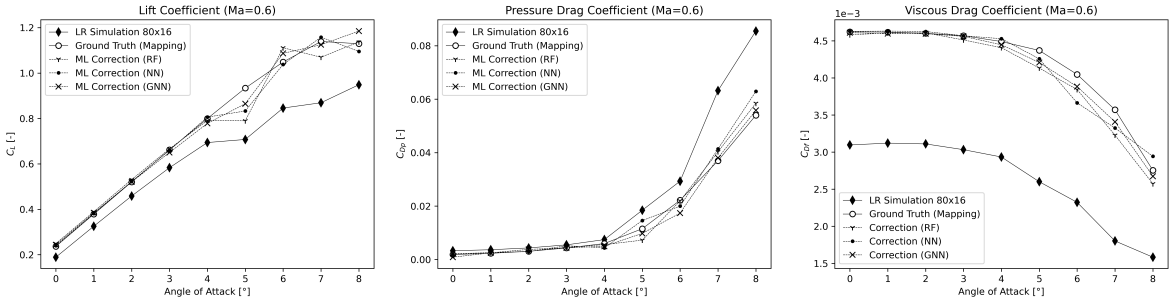
In figure 6, the friction and pressure coefficient along the airfoil surface are visible for the

coarse grid simulation to be corrected, the ground truth and the ML corrections for the same test set (M=0.6). Figure 6a presents a case with low angle of attack where the coarse grid simulation converged and all models performed well, as can be seen in the improved friction coefficient. Figure 6b shows a case at high angle of attack, where the coarse grid simulation did not converge. It is again visible that the GNN model performed best, since the other models only slightly improve the pressure along the shock location. The integrated lift, pressure drag and viscous drag coefficients are given in figure 7. All models reduce the error significantly at low angles of attack, where all simulations converged. At higher angles of attack, a reduction of the prediction accuracy occurs due to physical non-linearities and uncertainties from the non-converged data sets. Still, all models improve the coefficients showing the right trend with the GNN being the best.



(a) Friction coefficient for AoA=2.0 (b) Pressure coefficient for AoA=7.0

Figure 6: Corrected surface coefficients for M=0.6



(a) Lift coefficient (b) Pressure drag coefficient (c) Friction drag coefficient

Figure 7: Corrected coefficients for M=0.6

A final comparison on the test set RAE2822 between the models is done by computing the coefficient of determination R^2 as given in equation 7. Figure 8 contains the scores at the lowest angle of attack AoA=0.0 and the three highest values of AoA={6.0, 7.0, 8.0}. Only the RF model predicts for low angles of attack consistently accurate results for all variables. Instead, the NN and GNN model show a decreased performance for the turbulence model variable $\tilde{\nu}$ but consistent R^2 scores for the remaining variables. Looking at the high angles of attack and therefore non-converged cases, the GNN outperforms the other models for all variables.

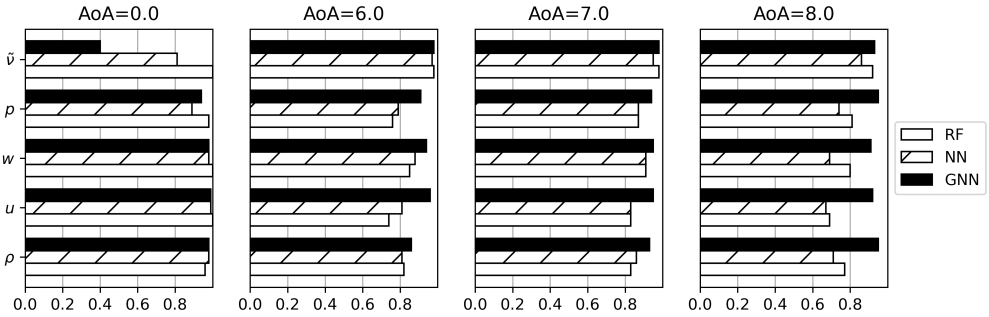


Figure 8: Coefficient of determination R^2 for test set RAE2822 M=0.6, AoA={0.0, 6.0, 7.0, 8.0}

3.6 Generalization Test Case

As a test case to assess the generalization capabilities of the trained models, additional simulations are carried out on a different transonic airfoil geometry. For this, mesh deformation has been performed based on the HR grid of the RAE2822 with a subsequent agglomeration to obtain the LR grid. The similar airfoil shape RAE5212 is chosen, as given in figure 9. Solutions were computed for M={0.35, 0.6} at angles of attack 3.0 and 7.0. The angles of attack were chosen to cover a case in the linear and in the non-linear flow regime as well as a converged and a non-converged computation.

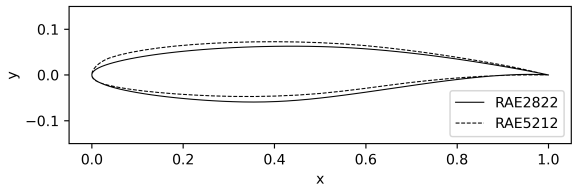
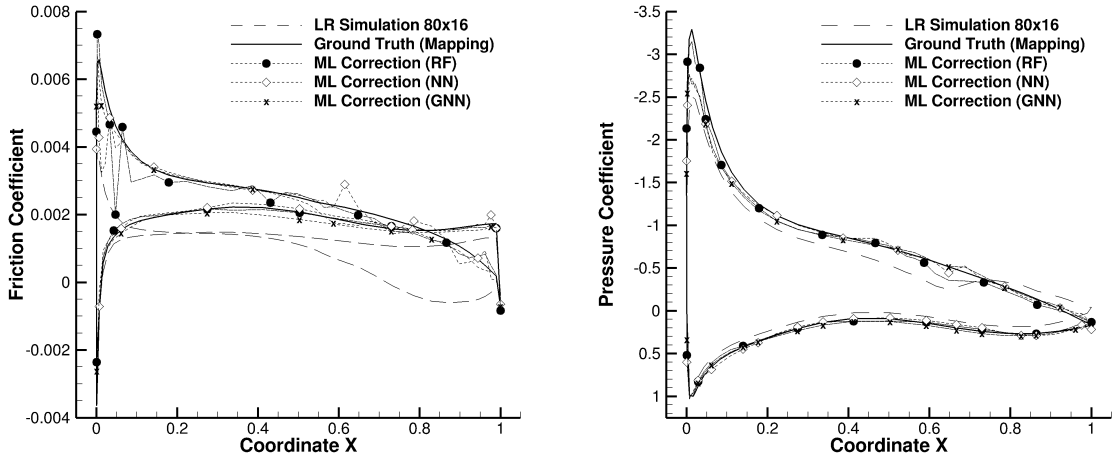


Figure 9: Airfoils for this study

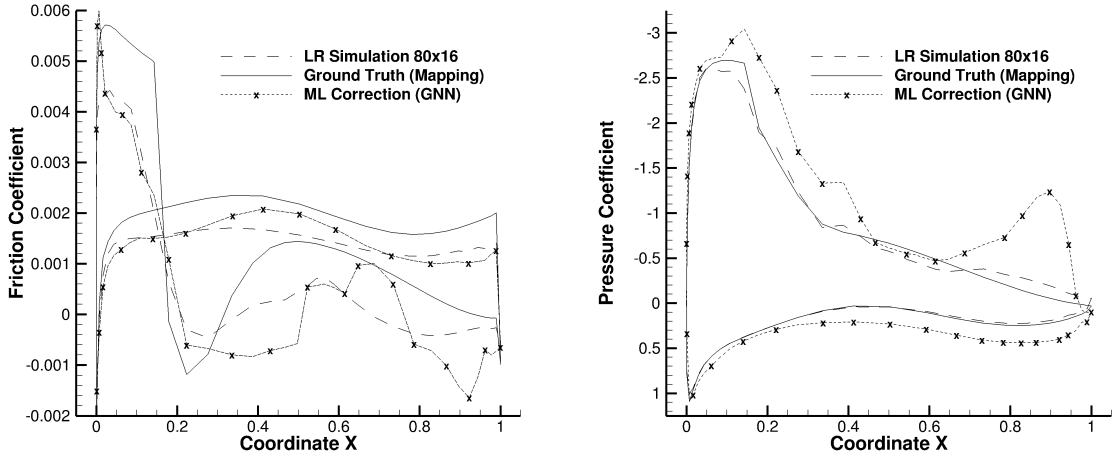
The ML models trained on the RAE2822 geometry as described above were directly employed for the prediction on the RAE5212 data sets. In general, all models performed well for the lower Mach number independent of the angle of attack. This can be seen in the derived friction and pressure coefficient in figure 10. Although all models tend to improve these coefficients, the GNN model shows the least noisy behaviour. For M=0.6 at AoA=3.0, similar improvements are observed. The limits in the predictive capability of the models are found for M=0.6 at an angle of

attack of 7.0, where all models fail to improve the coarse grid simulation and even deteriorate the LR simulation where it was comparable to the ground truth. Friction and pressure coefficients of this case for the GNN are plotted in figure 11.



(a) Friction coefficient for AoA=7.0 (b) Pressure coefficient for AoA=7.0

Figure 10: Prediction for RAE5212 for M=0.35



(a) Friction coefficient for AoA=7.0 (b) Pressure coefficient for AoA=7.0

Figure 11: Prediction for RAE5212 for M=0.6

4 CONCLUSION

Conducting CFD simulations on grids with sufficient degrees of freedom to approximate the actual solution is in practice rarely feasible. Thus, it is of interest to find methods which generate accurate results while keeping computational costs low. This work presents a procedure to quantify the discretization error between solutions of two nested grids and, subsequently, employs a ML model to correct the inaccurate coarse grid solution. Three different ML models were tested and compared: A random forest, a neural network and a graph neural network.

Good results are achieved on the test set for the RAE2822 airfoil covering different flow regimes. The RF model shows good results in the linear regime, whereas it was outperformed in the non-linear region, especially for the non-converged cases, by the GNN model. The overall better performance of the GNN model shows that it is capable of grasping more information by taking into account the connectivity to neighbouring nodes and their features. Qualitatively, the field corrections were analyzed and a quantitative analysis was done by deriving various coefficients such as surface distribution as well as global integral values. Difficulties were encountered in the non-linear regime, where additional uncertainties were introduced by using non-converged data sets. The generalization test on the different airfoil geometry RAE5212 has shown that the models obtain accurate results for a low angle of attack for both low and high Mach number. Improvements at a higher angle of attack, which leads for both Mach numbers to a non-converged simulation on the coarse grid, were only possible at low Mach numbers. The models fail to generalize on a non-converged data set for high angles of attack and high Mach numbers. On the one hand, a limitation of the method is the chosen mapping function, which naturally decreases the data quality obtained on the fine grid. On the other hand, it was shown that ML models can find patterns in coarse grid CFD simulations and therefore improve them, even if the data set contains non-converged simulations. Additionally, conducting a local cell-wise correction does not limit the correction on a static number of data points. Finally, the use of the GNN instead of classical CNNs seems to be promising for cases where simpler models such as the RF fail. Since the NN model showed a noisy behaviour on various test cases, the training procedure could be reconsidered with particular focus on avoiding overfitting effects.

For the future, the proposed method will be tested on a three dimensional data set. Moreover, developing a more integrated methodology by closely coupling an ML model with the CFD solver could enable improvements for correcting the discretization error beyond a pure post-processing approach.

REFERENCES

1. Langer, S. Investigations of a compressible second order finite volume code towards the incompressible limit. *Computers & Fluids* **149,** 119 –137. ISSN: 0045-7930. http://www.sciencedirect.com/science/article/pii/S0045793017300683 (2017).
2. Berger, M. J. & Jameson, A. Automatic adaptive grid refinement for the Euler equations. *AIAA Journal* **23,** 561–568. eprint: https://doi.org/10.2514/3.8951. https://doi.org/10.2514/3.8951 (1985).
3. Berger, M. & Colella, P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* **82,** 64–84. ISSN: 0021-9991. https://www.sciencedirect.com/science/article/pii/0021999189900351 (1989).

4. Brunton, S. L., Noack, B. R. & Koumoutsakos, P. Machine Learning for Fluid Mechanics. *Annual Review of Fluid Mechanics* **52,** 477–508. eprint: https://doi.org/10.1146/annurev-fluid-010719-060214. https://doi.org/10.1146/annurev-fluid-010719-060214 (2020).
5. Vinuesa, R. & Brunton, S. L. *The Potential of Machine Learning to Enhance Computational Fluid Dynamics* 2021. https://arxiv.org/abs/2110.02085.
6. Duraisamy, K. Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence. *Phys. Rev. Fluids* **6,** 050504. https://link.aps.org/doi/10.1103/PhysRevFluids.6.050504 (5 May 2021).
7. Fukami, K., Fukagata, K. & Taira, K. Assessment of supervised machine learning methods for fluid flows. *Theoretical and Computational Fluid Dynamics* **34,** 497–519. ISSN: 0935-4964 (2020).
8. Dong, C., Loy, C. C., He, K. & Tang, X. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **38,** 295–307 (2016).
9. Fukami, K., Fukagata, K. & Taira, K. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics* **870,** 106–120. ISSN: 0022-1120. http://arxiv.org/pdf/1811.11328v2 (2019).
10. Jiang, C. M. *et al. MeshfreeFlowNet: A Physics-Constrained Deep Continuous Space-Time Super-Resolution Framework* in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (IEEE Press, Atlanta, Georgia, 2020). ISBN: 9781728199986.
11. Kochkov, D. *et al.* Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences* **118** (May 2021).
12. Um, K., Brand, R., Fei, Y. R., Holl, P. & Thuerey, N. *Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H.) **33** (Curran Associates, Inc., 2020), 6111–6122. https://proceedings.neurips.cc/paper/2020/file/43e4e6a6f341e00671e123714de019a8-Paper.pdf.
13. Hanna, B. N., Dinh, N. T., Youngblood, R. W. & Bolotnov, I. A. Machine-learning based error prediction approach for coarse-grid Computational Fluid Dynamics (CG-CFD). *Progress in Nuclear Energy* **118,** 103140. ISSN: 01491970 (2020).
14. Gao, H., Sun, L. & Wang, J.-X. PhyGeoNet: Physics-Informed Geometry-Adaptive Convolutional Neural Networks for Solving Parameterized Steady-State PDEs on Irregular Domain. *Journal of Computational Physics* **428,** 110079. ISSN: 00219991. http://arxiv.org/pdf/2004.13145v2 (2021).
15. Morimoto, M., Fukami, K., Zhang, K., Nair, A. G. & Fukagata, K. Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low dimensionalization. *Theoretical and Computational Fluid Dynamics* **35,** 633–658. ISSN: 0935-4964 (2021).
16. Allmaras, S., Johnson, F. & Spalart, P. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. *Seventh International Conference on Computational Fluid Dynamics (ICCFD7),* 1–11 (Jan. 2012).
17. Breiman, L. Random Forests. *Machine Learning* **45,** 5–32 (Oct. 2001).

18. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE transactions on neural networks* **20,** 61–80 (2009).
19. Kipf, T. N. & Welling, M. *Semi-Supervised Classification with Graph Convolutional Networks* in *5th International Conference on Learning Representations (ICLR-17)* (2016). https://arxiv.org/abs/1609.02907.
20. Bergstra, J., Bardenet, R., Bengio, Y. & Kégl, B. *Algorithms for Hyper-Parameter Optimization* in *Advances in Neural Information Processing Systems* (eds Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F. & Weinberger, K.) **24** (Curran Associates, Inc., 2011). https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.
21. Kroll, N., Langer, S. & Schwöppe, A. *The DLR Flow Solver TAU - Status and Recent Algorithmic Developments* in *52nd Aerospace Sciences Meeting* (2014). eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2014-0080. https://arc.aiaa.org/doi/abs/10.2514/6.2014-0080.
22. Langer, S. *An initial investigation of solving RANS equations in combination with two-equation turbulence models* tech. rep. (Institut für Aerodynamik und Strömungstechnik, Sept. 2019). https://elib.dlr.de/129170/.
23. White, J. A., Nishikawa, H. & Baurle, R. A. *Weighted Least-squares Cell-Average Gradient Construction Methods For The VULCAN-CFD Second-Order Accurate Unstructured Grid Cell-Centered Finite-Volume Solver* in *AIAA Scitech 2019 Forum* (2019). eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2019-0127. https://arc.aiaa.org/doi/abs/10.2514/6.2019-0127.
24. Yeo, I.-K. & Johnson, R. A. A New Family of Power Transformations to Improve Normality or Symmetry. *Biometrika* **87,** 954–959. ISSN: 00063444. http://www.jstor.org/stable/2673623 (2022) (2000).
25. Buitinck, L. *et al. API design for machine learning software: experiences from the scikit-learn project* in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013), 108–122.
26. Bekemeyer, P. *et al. Data-driven Aerodynamic Modeling Using the DLR SMARTy Toolbox* in *Aviation Conference 2022, (Paper accepted for publication)* (2022).
27. Paszke, A. *et al.* in *Advances in Neural Information Processing Systems 32* 8024–8035 (Curran Associates, Inc., 2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
28. Fey, M. & Lenssen, J. E. *Fast Graph Representation Learning with PyTorch Geometric* in *ICLR Workshop on Representation Learning on Graphs and Manifolds* (2019).
29. Akiba, T., Sano, S., Yanase, T., Ohta, T. & 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).

5 APPENDIX

5.1 Conducted simulations

The following plot lists the Mach numbers and angles of attack, which were used to create the data sets. For testing, all simulations at Mach 0.6 were used (9 simulations), whereas the other

simulations were used for training and validation (135 simulations). The blue and red crosses indicate converged and non-converged simulations for the 80×16 grid, respectively. There are in total 71 converged and 73 non-converged data sets.

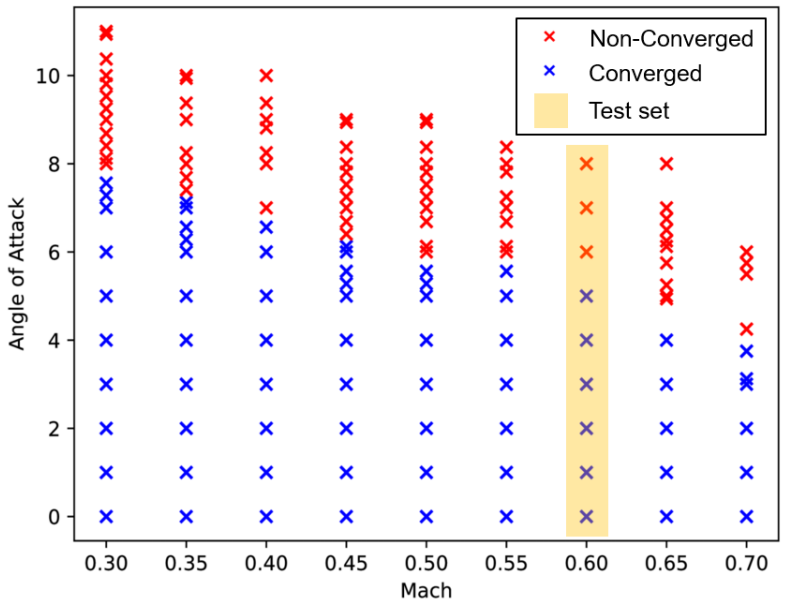


Figure 12: Data sets

5.2 Hyperparameter Tuning

5.2.1 Random Forest

The tuned hyperparameters for the RF model are the number of decision tree estimators N, the minimum samples at the leaf nodes s, the maximum features x_{max} for a split and if bootstrapping B is applied or not. For the RF implementation the Scikit-learn library [25] was used.

Table 1: RF Hyperparameter

	Search space	Final hyperparameter
N	[30-250]	126
s	[1-10]	1
x_{max}	[5-37]	15
B	[True, False]	False

5.2.2 Neural Network

For the NN architecture the number of hidden layers l, number of neurons in each hidden layer n_l, batch size b, activation function for the input and hidden layers g, optimizer o and the

learning rate lr with and without exponential decrease are chosen for the tuning process. For the implementation of the NN the SMARTy library [26] was used with the PyTorch backend [27].

Table 2: NN Hyperparameter

	Search space	Final hyperparameter
l	[6-12]	9
n_l	[50-550]	{486, 501, 481, 340, 416, 318, 472, 396, 315, 391, 282}
b	[32, 128, 256, 1024, 1280*, 2048]	1024
g	[tanh, relu]	tanh
o	[Adam, SGD]	Adam
lr	[0.00001-0.001]	0.000117 with exp. decrease

*Corresponds to the number of data points in one simulation case.

5.2.3 Graph Neural Network

For the GNN architecture the same ADAM optimizer and tanh activation function were chosen for the NN training. The learning rate lr, the exponential decay rate γ, the number of stacked convolutional layers l_c (which corresponds to an aggregation of "hops" in the neighborhood around each node) and the feature dimensionality H_l of each such layer (excluding in- and output) was optimized for. The GNN was implemented using the pytorch geometric library [28] and specifically the implementation of the graph convolutional operator "GCNConv" proposed by Kipf et al. [19] was used.

Table 3: GNN Hyperparameter

	Search space	Final hyperparameter
lr	[0.00005-0.001]	0.000797
γ	[0.95-1.0]	0.966
l_c	[8-14]	13
H_l	[50-550]	{353, 324, 416, 390, 439, 251, 363, 252, 390, 433, 546, 433, 419}