# Towards the efficient calculation of quantity of interest from steady Euler equations II: a CNNs-based automatic implementation.

Jingfeng Wang[1], Guanghui Hu[1,2,3,*]

[1] *Department of Mathematics, Faculty of Science and Technology, University of Macau, Macao S.A.R., China*
[2] *Zhuhai UM Science and Technology Research Institute, Zhuhai, Guangdong, China*
[3] *Guangdong-Hong Kong-Macao Joint Laboratory for Data-Driven Fluid Mechanics and Engineering Applications, University of Macau, China*

**Abstract.** In [J. Wang, G. Hu, arxiv: 2302.14262], a dual-consistent dual-weighted residual-based *h*-adaptive method has been proposed based on a Newton-GMG framework, toward the accurate calculation of a given quantity of interest from Euler equations. The performance of such a numerical method is satisfactory, i.e., the stable convergence of the quantity of interest can be observed. In this paper, we will focus on the efficiency issue to further develop this method. Three approaches are studied for addressing the efficiency issue, i.e., i). using convolutional neural networks as a solver for dual equations, ii). designing an automatic adjustment strategy for the tolerance in the *h*-adaptive process to conduct the local refinement and/or coarsening of mesh grids, and iii). introducing OpenMP, a shared memory parallelization technique, to accelerate the module such as the solution reconstruction in the method. The feasibility of each approach and numerical issues are discussed in depth, and significant acceleration from those approaches in simulations can be observed clearly from a number of numerical experiments. In convolutional neural networks, it is worth mentioning that the dual consistency plays an important role in guaranteeing the efficiency of the whole method and that unstructured meshes are employed in all simulations.

**Key words**: DWR-based *h*-adaptivity; Dual consistency; Convolutional neural networks; Automatic adjustment of tolerance; Newton-GMG method for Euler equations

## 1 Introduction

In [1], a dual-consistent dual-weighted residual(DWR)-based adaptive mesh method has been constructed, from which the smooth convergence of quantity of interest can be observed. This technique is particularly advantageous in applications such as the optimal

*Corresponding author. *Email addresses:* garyhu@um.edu.mo

design of vehicle shapes where the quantity of interest, e.g., the lift-to-drag ratio, needs an accurate evaluation.

It is noted that the efficiency of a numerical method is a key feature towards the practical applications, as in a typical problem of optimal design of the vehicle shape, governing partial differential equations (PDEs) need to be solved numerous times, depending on factors such as the number of design parameters, mesh resolution, optimization method [2]. In our previous work, different modules like reconstruction [3–5], NURBS enhancement [6] and DWR-based refinement [7, 8] are integrated. As a consequence, in this paper, efforts are devoted to further improving the efficiency of the numerical method proposed in [1], to make the method and related library AFVM4CFD a competitive one in the market.

For such a purpose, three approaches will be studied in depth in this paper, including i). using convolutional neural networks (CNNs) to produce numerical solutions of the dual problem, ii). designing an automatic adjustment strategy for the tolerance in the *h*-adaptive process, and iii). introducing OpenMP to parallelize modules such as the solution reconstruction and dual solver in the method.

It is noted that Galerkin orthogonality between numerical solutions and related error is the reason why Hartmann [9] suggested solving dual equations in a larger space. In [1], the refinement of mesh grids is employed for such a purpose. However, this approach is not a good choice from the efficiency point of view [10]. Although there have been other choices such as raising the order of approximate polynomials [11], high order interpolation of numerical solutions of dual equations [12], it should be pointed out that, i). solving dual equations using a classical solver involved in all aforementioned approaches, in which the time-consuming modules such as the construction of the finite-dimensional space, solving the system of linear equations, etc., need to be implemented, and ii). numerical solutions of dual equations are adopted as a weight for the mesh adaptation in the current framework. Out of the motivation to develop a high-quality mesh, the trained neural network model can fulfill this requirement while saving time significantly.

CNNs, based on the above two observations, become an ideal choice for the solver of dual equations, due to their ability to fastly generate acceptable numerical solutions of dual equations. In the recent past, neural networks have been successfully applied to implement the DWR-based h-adaptation method in computational fluid dynamics(CFD). For instance, in [13, 14], an encoder-decoder algorithm is developed to predict the indicators generated from dual equations, and in [15], primal solutions and dual solutions are both obtained from deep neural networks. Similarly, a data-driven goal-oriented mesh adaptation approach is developed to generate the error indicators in [16]. The recent breakthrough in convolutional neural networks makes training in high-dimensional problems possible to handle numerical computations in CFD. Besides, libraries build in Tensorflow [17], Pytorch [18], and cuDNN provide powerful tools to design specific models that cater to the demands of CFD issues, where the GPUs module can be adopted for further improvement in the training process. Another reason to choose CNNs in this

work is their generalization ability, i.e., quality numerical solutions of dual equations with different attack angles, Mach numbers, etc. can be generated effectively. Furthermore, the structures of the datasets in our framework are well-suited for the CNNs, where the channel collects similar information of specific elements from its reconstruction patch while height and width represent the value of different variables with corresponding mathematical properties and dimensionality.

To accelerate the numerical simulation, an automatic adjustment strategy for the tolerance is constructed in the *h*-adaptive process. It is known that in an *h*-adaptive mesh method after an error indicator is generated for each element, a tolerance needs to be given to guide the local refinement and/or coarsening of mesh grids so that the global error of numerical solutions can be effectively controlled. In [1], the strategy for setting up this tolerance is to use a fixed and sufficiently small one during the whole simulation. This is not a good strategy since in the first several rounds of adaptive refinement of mesh grids, unreliable error indicators would result in over-refinement and/or coarsening of mesh grids if a sufficiently small tolerance is employed. Furthermore, the nonlinearity of governing equations also implies that a dynamic adjustment of the tolerance would be a better choice. Motivated by the decreasing threshold method proposed in [19, 20], error indicators are plotted for analysis and the corresponding dynamic tolerance selection method is developed in this work. Then an automatic DWR-based h-adaptivity method is developed without manual intervention.

Another efficient technique for accelerating simulations is to use OpenMP [21] to parallelize modules both for solving primal and dual equations. It is noted that the hardware used in this work for numerical experiments is a workstation with multiple CPU cores, which makes OpenMP a perfect choice for parallelization. In OpenMP, data race is a key issue that should be avoided during the parallelization of the algorithm. The good news is that in our Newton-GMG framework, there are many time-consuming modules, such as the solution reconstruction, the update of cache information for each element, as well as the multigrid method, which can be implemented without data race. Hence, in this paper, the OpenMP parallelization of the algorithm and code proposed in [1] will be discussed in detail for the acceleration.

In this paper, we will follow the aforementioned three approaches to further improve the efficiency of the numerical method proposed in [1]. Firstly, we have constructed a CNNs-based dual solver. For fear of overfitting, techniques such as batch normalization and pooling have been incorporated. Furthermore, in order to circumvent the issues of gradient vanishing and explosion, initialization and residual connection modules are integrated, as highlighted in references [22,23]. This enhanced solver efficiently predicts the dual equations on the current mesh, facilitating rapid adaptation processes. Notably, the trained model demonstrates generalization capabilities across training datasets, configurations, and multiple airfoils. Secondly, an automatic adjustment strategy for tolerance is devised in the *h*-adaptive process. This strategy revolves around plotting the distribution of mesh adaptation steps and analyzing them using the Kolmogorov-Smirnov method. Our findings suggest that the distribution of these indicators closely follows the Weibull

distribution, characterized by a KS-stat at its uppermost value. However, it should be noted that the parameters might differ for various adaptation steps. In some configurations, the distribution even aligns more closely with the gamma distribution. Owing to these variations, relying on a mathematical expression to select tolerance values is deemed impractical. Instead, we have employed a methodology where values of indicators are systematically sorted, and a fixed proportion is chosen to derive the tolerance for different adaptation steps. Finally, three modules in our algorithm, i.e., solution reconstruction, update cache information for each element and CNNs form dual solver are parallelized with OpenMP. The scalability of these modules is illustrated in experiments with the different number of threads. Using the standard metric of speedup, typical in parallel computing evaluations, our results indicate a substantial enhancement in the performance of the framework.

It is found from the numerical experiments that the CNNs dual solver saves time in an order of magnitude compared with the traditional solver. More specifically, the time complexity is $O(n)$. Besides, the dynamic tolerance strategy accelerates the calculation of the target functional and benefits for automatic selection without manual intervention. With such a strategy, a stable growth rate of mesh size can be expected. In [1], the importance of dual consistency within the DWR-based h-adaptation is discussed, such property extends to the construction of training datasets, which has been further elaborated upon in this research. The trained model has generalization capabilities that configurations beyond the trained category can still be simulated precisely. It is worth noting that our CNNs solver can generate reliable dual solutions on unstructured mesh. With the introduced parallel module, the time spent on primal solver and dual solver has been saved a lot.

The rest of this paper is organized as follows. In Section 2, we present the fundamental notations and provide a concise introduction to the Newton-GMG solver. In Section 3, the architecture of our convolutional neural networks is introduced and the training performance is discussed. We emphasized the importance of dual consistency in the training process. In Section 4, two acceleration strategies are discussed. We organize the dynamic tolerance selection method and analyze the effects of this algorithm. Then parallel computing is adopted. In Section 5, we present the details of the numerical experiments, which show the reliability of our CNNs-based dual solver.

# 2 A Brief Introduction to Dual-Consistent DWR-based h-adaptive Newton-GMG

## 2.1 Basic notations

Let $\Omega$ be a domain in $\mathbb{R}^2$ with boundary $\Gamma$. The inviscid two-dimensional steady Euler equations derived from the conservation laws can be written as: Find $\mathbf{u} : \Omega \to \mathbb{R}^4$

such that

$$\nabla \cdot \mathcal{F}(\mathbf{u}) = 0, \quad \text{in } \Omega, \tag{1}$$

subject to a certain set of boundary conditions. Here $\mathbf{u}$ and $\mathcal{F}(\mathbf{u})$ are the conservative variables and fluxes, which are given by

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ E \end{bmatrix}, \qquad \text{and } \mathcal{F}(\mathbf{u}) = \begin{bmatrix} \rho u_x & \rho u_y \\ \rho u_x^2 + p & \rho u_x u_y \\ \rho u_x u_y & \rho u_y^2 + p \\ u_x(E+p) & u_y(E+p) \end{bmatrix}, \tag{2}$$

where $(u_x, u_y)^T, \rho, p, E$ denote the velocity, density, pressure, and total energy, respectively. We use the equation of state to close the system, which is

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u_x^2 + u_y^2). \tag{3}$$

Here $\gamma = 1.4$ is the ratio of the specific heat of the perfect gas.

## 2.2 Finite volume discretization

In [8], Equations (1) are solved with finite volume method. The methodology will be briefly introduced as follows. To derive a discretization scheme, a shape regular subdivision $\mathcal{T}$ turns $\Omega$ into different control volumes. $K_i$ is used to define the $i$-th element in this subdivision. $e_{i,j}$ denotes the common edge of $K_i$ and $K_j$, i.e., $e_{i,j} = \partial K_i \cap \partial K_j$. The unit outer normal vector on the edge $e_{i,j}$ with respect to $K_i$ is represented as $n_{i,j}$. Consequently, with the divergence theorem, the Euler equations in this discretized domain can be reformulated as

$$\mathcal{A}(\mathbf{u}) = \int_\Omega \nabla \cdot \mathcal{F}(\mathbf{u}) dx = \sum_i \int_{K_i} \nabla \cdot \mathcal{F}(\mathbf{u}) dx = \sum_i \sum_j \oint_{e_{i,j} \in \partial K_i} \mathcal{F}(\mathbf{u}) \cdot n_{i,j} ds = 0. \tag{4}$$

With the numerical flux $\mathcal{H}(\cdot,\cdot,\cdot)$ introduced in this equation, a fully discretized system can be obtained as

$$\sum_i \sum_j \oint_{e_{i,j} \in \partial K_i} \mathcal{H}(\mathbf{u}_i, \mathbf{u}_j, n_{i,j}) ds = 0. \tag{5}$$

## 2.3 The Newton-GMG solver

To solve the Equation (5), the Newton method is employed for the linearization and a linear multigrid method proposed in [24] is utilized for solving the system. At first, the Equation (5) is expanded by the Taylor series. By neglecting the higher-order terms, the system becomes

$$\begin{aligned}\alpha\left\|\sum_i\sum_j\int_{e_{i,j}\in\partial\mathcal{K}_i}\mathcal{H}(\mathbf{u}_i^{(n)},\mathbf{u}_j^{(n)},n_{i,j})ds\right\|_{L_1}\Delta\mathbf{u}_i^{(n)}&+\sum_i\sum_j\int_{e_{i,j}\in\partial\mathcal{K}_i}\Delta\mathbf{u}_i^{(n)}\frac{\partial\mathcal{H}(\mathbf{u}_i^{(n)},\mathbf{u}_j^{(n)},n_{i,j})}{\partial\mathbf{u}_i^{(n)}}ds\\&+\sum_i\sum_j\int_{e_{i,j}\in\partial\mathcal{K}_i}\Delta\mathbf{u}_j^{(n)}\frac{\partial\mathcal{H}(\mathbf{u}_i^{(n)},\mathbf{u}_j^{(n)},n_{i,j})}{\partial\mathbf{u}_j^{(n)}}ds\\&=-\sum_i\sum_j\int_{e_{i,j}\in\partial\mathcal{K}_i}\mathcal{H}(\mathbf{u}_i^{(n)},\mathbf{u}_j^{(n)},n_{i,j})ds,\end{aligned}\tag{6}$$

where $\partial\mathcal{H}(\cdot,\cdot,\cdot)/\partial\mathbf{u}$ denotes the Jacobian matrix of numerical flux, and $\Delta\mathbf{u}_i$ is the increment of the conservative variables in the $i-$th element. After each Newton iteration, the cell average is updated by $\mathbf{u}_i^{(n+1)}=\mathbf{u}_i^{(n)}+\Delta\mathbf{u}_i^{(n)}$. In the simulation, a regularization term is added to the system. This regularization is effective since the local residual can quantify whether the solution is close to a steady state. Based on the observation, the solution is far from a steady state at the initial stage. With the iteration processed, the solution gets updated and approaches the steady state where the regularization term gets close to zero.

To solve the Equation (6), the geometrical multigrid technique is applied. In [1], the regularization and geometrical multigrid methods are also applied to the dual equations which perform well to get a robust solver for obtaining dual equations.

## 2.4 Dual Weighted Residual Mesh Adaptation

In practical issues, engineers care about how to calculate the quantity of interest efficiency. For instance, in the field of shape optimal design, lift and drag are of main concern. Then the target functionals are considered as

$$\mathcal{J}(\mathbf{u})=\int_\Gamma p_\Gamma(\mathbf{u})\mathbf{n}\cdot\beta,\tag{7}$$

where $p_\Gamma$ is the pressure along the boundary of airfoil and $\beta$ in the above formula is given as

$$\beta=\begin{cases}(\cos\alpha,\sin\alpha)^T/C_\infty, \text{ for drag calculation,}\\(-\sin\alpha,\cos\alpha)^T/C_\infty, \text{ for lift calculation.}\end{cases}\tag{8}$$

Indeed, the quantity of interest can be calculated with high precision if the solutions from (1) are solved accurately. However, the computational cost increases exponentially with uniform refinement. Then dual-weighted residual method offers an opportunity to calculate the quantity of interest efficiency. In [12], discretized dual equations are solved from the perspective of extrapolation, which is easily implemented on a finite volume scheme. A brief introduction to this method will be given in the following section at first.

Suppose $\mathcal{T}_H$ is the partition of the computational domain into a coarse mesh, and $\mathcal{T}_h$ is the partition into fine mesh correspondingly. The primal equations (1) are denoted by a

residual form, where $\mathbf{u}_H$ denotes the numerical solutions obtained from the discretization on coarse mesh, i.e.

$$\mathcal{R}_H(\mathbf{u}_H) = 0. \tag{9}$$

Accordingly, $\mathbf{u}_h$ denotes the numerical solutions obtained from the discretization on fine mesh, i.e.

$$\mathcal{R}_h(\mathbf{u}_h) = 0. \tag{10}$$

The original motivation of this method is to estimate the target functional $\mathcal{J}(\mathbf{u})$ on a fine mesh $\mathcal{J}_h(\mathbf{u}_h)$ without solving the equations on fine mesh. A multiple-variable Taylor series expansion is applied at first, i.e.,

$$J_h(\mathbf{u}_h) = J_h(\mathbf{u}_h^H) + \left.\frac{\partial J_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H} (\mathbf{u}_h - \mathbf{u}_h^H) + \cdots, \tag{11}$$

here $\mathbf{u}_h^H$ represents the coarse solution $\mathbf{u}_H$ mapped onto the fine space $\mathcal{V}_h$ via some prolongation operator $I_h^H$,

$$\mathbf{u}_h^H = I_h^H \mathbf{u}_H. \tag{12}$$

Similarly, the equations on the fine mesh can be expanded through the Taylor expansion, i.e.,

$$\mathcal{R}_h(\mathbf{u}_h) = \mathcal{R}_h(\mathbf{u}_h^H) + \left.\frac{\partial \mathcal{R}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H} (\mathbf{u}_h - \mathbf{u}_h^H) + \cdots. \tag{13}$$

Symbolically, the Jacobian matrix $\left.\frac{\partial \mathcal{R}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}$ can be inverted to get the error representation,

$$\mathbf{u}_h - \mathbf{u}_h^H \approx -\left(\left.\frac{\partial \mathcal{R}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}\right)^{-1} \mathcal{R}_h(\mathbf{u}_h^H). \tag{14}$$

Substituting the expression (14) into (11), the quantity of interest can be denoted as

$$J_h(\mathbf{u}_h) = J_h(\mathbf{u}_h^H) - (\mathbf{z}_h)^T \mathcal{R}_h(\mathbf{u}_h^H), \tag{15}$$

where $\mathbf{z}_h|_{\mathbf{u}_h^H}$ is obtained from *Fully discrete dual equations*:

$$\left(\left.\frac{\partial \mathcal{R}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}\right)^T \mathbf{z}_h = \left(\left.\frac{\partial \mathcal{J}_h}{\partial \mathbf{u}_h}\right|_{\mathbf{u}_h^H}\right)^T. \tag{16}$$

In our earlier work, $\mathbf{z}_h$ is calculated on the embedded mesh. Various strategies have been proposed in the literature to simplify this process. For instance, [12, 20] used the error correction method which interpolates the dual solutions to the fine mesh, and in [25], a smoothing technique is proposed to modify the oscillations due to the interpolation. Even though, the mesh needs to be refined at first to make further calculations. In this

work, the dual solutions are obtained from the neural networks and can be used to generate the error indicators directly on the current mesh. The framework of our previous algorithm is briefly summarized as follows.

**Algorithm 1:** DWR for one-step mesh refinement

**Data:** Initial $\mathcal{T}_H$, $TOL$
**Result:** $\mathcal{T}_h$
1 Using the Newton-GMG to solve $\mathcal{R}_H(\mathbf{u}_H) = 0$ with residual tolerance $1.0 \times 10^{-3}$;
2 Interpolate solution $\mathbf{u}_H$ from the mesh $\mathcal{K}_H$ to $\mathcal{K}_h$ to get $\mathbf{u}_h^H$;
3 Record the residual $\mathcal{R}_h\left(\mathbf{u}_h^H\right)$;
4 Solve $\mathcal{R}_h(\mathbf{u}_h) = 0$ with residual tolerance $1.0 \times 10^{-3}$;
5 Using the Newton-GMG to solve the dual equation to get $\mathbf{z}_h$;
6 Calculate the error indicator for each element;
7 **while** $\mathcal{E}_{K_H} > TOL$ *for some* $K_H$ **do**
8 | Adaptively refine the mesh $\mathcal{K}_H$ with the process in [8];
9 **end**

Even though the dual-consistent DWR-based mesh adaptation method shows great potential for accurately resolving quantities of interest, the generation of high-quality mesh continues to be a computationally demanding process. We observe that machine learning techniques, specifically convolutional neural networks (CNNs), may enable the rapid attainment of dual solutions without compromising precision significantly. The application of CNNs in predicting dual solutions only affects the adaptation process, leaving the underlying physical mechanisms of the primal equations unaltered. Consequently, the derivation of dual solutions appears apt for replacement with CNNs-based methodologies. In the next section, we will introduce how to train neural networks to get the dual equations so that we can substitute the calculation on the uniformly refined mesh, i.e. step 4 and step 5.

## 3 CNNs-based Solutions of Dual Equations

Convolutional neural networks are very similar to the traditional fully connected neural networks as shown in Figure 1, consisting of learnable weights and biases in different layers. However, the practical issues often concern high-dimensional problems where the connection between different neurons and layers may become complicated. In [13], autodecoder neural networks are trained to generate indicators and a topology map converts the information to the structured mesh. In this work, since the adaptation is conducted on an unstructured mesh, the map relations are hard to construct. Hence, we build the architecture which learns from the data directly.

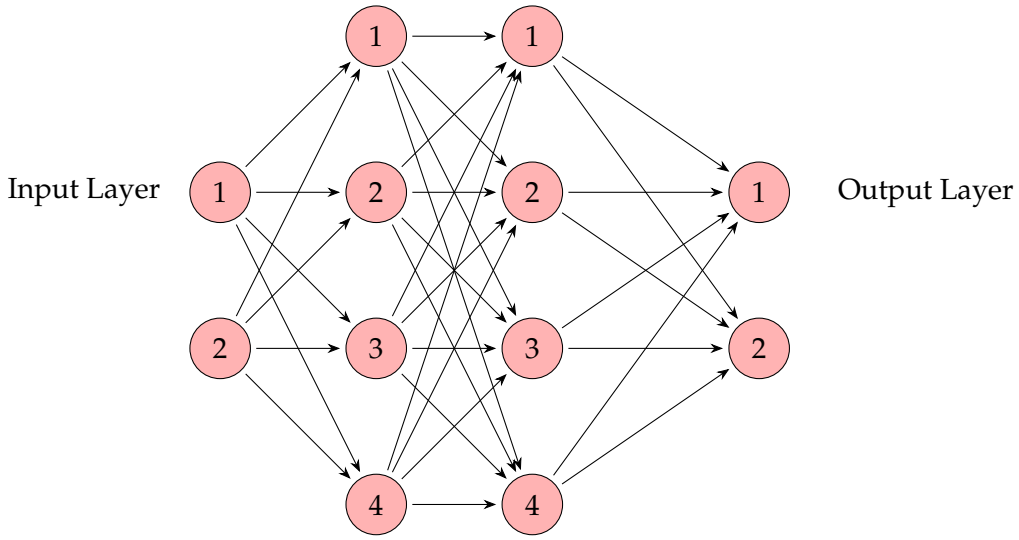



Figure 1: Fully connected neural networks.

## 3.1 Convolutional neural networks

In Equation (16), the dual equations are obtained from the primal solutions. In an abstract form, the dual equations can be treated as a functional of $\mathcal{R}_h, \mathbf{u}_h, \mathcal{J}_h$, i.e,

$$\mathbf{z}_h = \mathcal{N}(\mathcal{R}_h, \mathbf{u}_h, \mathcal{J}_h), \tag{17}$$

where the functional $\mathcal{N}(\cdot,\cdot,\cdot)$ is unknown. Since the information of mesh grids is not included in the training process, $\mathcal{J}_h$ is not suitable to be treated as input. Alternatively, while the quantity of interest is the integration around the boundary of the airfoil in this specific model, the location of the elements and the configurations can work well as inputs. Then the dual solutions can be seen as functional with the expression as follows,

$$\mathbf{z}_h = \widetilde{\mathcal{N}}(\mathcal{R}_h, \mathbf{u}_h, \vec{G}, Ma, \theta), \tag{18}$$

where $\vec{G}$ is the coordinate of the barycenter of elements, $Ma$ is the Mach number and $\theta$ is attack angle. The neural networks are trained to approximate the unknown functional $\widetilde{\mathcal{N}}(\cdot,\cdot,\cdot,\cdot,\cdot)$. The mechanism behind the relations is complicated and the dimension is high for training a satisfactory model within the fully connected neural networks framework. Besides, the data structure itself has different spatial and physical properties which are well-suited for the convolutional operation. Consequently, we consider using the CNNs for the training.

CNNs offer a more efficient approach to handling high-dimensional problems compared to fully connected networks. Instead of connecting all neurons from the preceding layer, CNNs establish connections only within local regions of the input. This specialized architecture consists of multiple learnable filters, each possessing parameters such

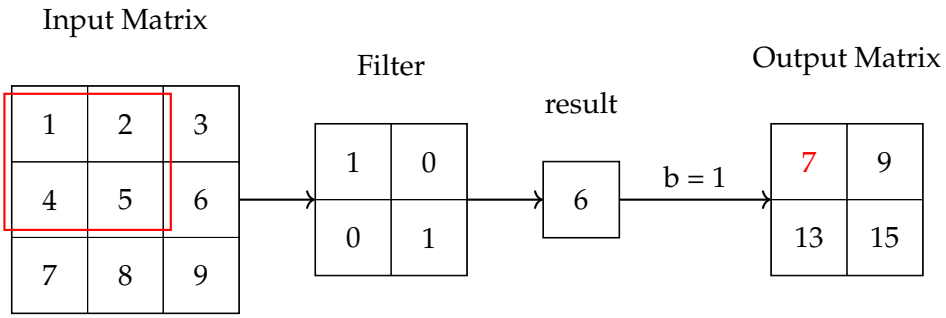



Figure 2: Convolutional operation.

as weights and biases. When the networks become deeper, the prediction in CNNs is quicker than that in fully connected neural networks. Inputs for CNNs are typically tensors or matrices. As filters slide across different portions of the input, they encode information and transmit it to the subsequent layer. This selective connectivity allows CNNs to exploit spatial hierarchies and local structures in the input data, resulting in more effective and computationally efficient learning. Similarly, a decoder process is also integrated into this framework to pass the encoded information to the final results.

While the training gets processed, the parameters will be updated till the preset terminal. In practical issues, connections between inputs and outputs are often hard to capture. By introducing the nonlinear activation function, CNNs can learn more complex patterns and relationships in data. Moreover, the activation function serves not only as a threshold to determine whether the output should be passed to the subsequent layer but also plays a pivotal role in the behavior and training dynamics of neural networks. Centralized activations, where the mean activation of neurons is close to zero, offer several benefits. For instance, when activations are centered around zero, the direction of gradient descent during training tends to be more consistent, potentially leading to faster convergence. Moreover, non-centralized activations can exacerbate the vanishing or exploding gradient problems, especially in deeper networks [26, 27]. This is crucial during the backpropagation process, where differentiable activation functions like the sigmoid, ReLU, or tanh enable gradient-based learning, guiding the network to iteratively adjust its weights based on the data. In this work, we adopted the exponential linear unit (ELU) as the activation function [28]. Unlike other activation functions, ELU allows a negative value range for inputs less than zero, thereby ensuring that the mean activation of the neurons remains closer to zero. This centralizing property of ELU helps in mitigating the bias shift effect considerably, offering a significant advantage over other schemes.

To speed up the training model, we utilized batch normalization, a technique that accelerates convergence by minimizing internal covariate shifts. After this process, a higher learning rate can be set without causing unstable training dynamics. Additionally, for fear of overfitting issues, max pooling methods are incorporated as well. Max pooling reduces the number of parameters while preserving the most salient features irrespective

of their spatial position. A brief framework of our CNNs can be found in Table 4 in the Appendix.

Since the dual equation (16) is solved through the information from primal solutions, we trained the model with inputs to be the two-dimensional location information, the Mach number and attack angle, the volume of a specific element, the four-dimensional solution information and four-dimensional residual information. Then the outputs are the dual solutions.

In the training process, we adopted the initialization method proposed in [22]. It provides a better weight initialization for Rectified Linear Unit(ReLU)-based networks, leading to faster convergence. Besides, we adopted the leaky ReLU activation function during the initialization process. In order to learn the characteristics of the data better, the inception module is introduced to help CNNs learn more effective and complex patterns. Besides, since the constructed neural networks contain multiple layers, the initialization method addresses the vanishing gradient problem more effectively for networks using ReLU activation functions, ensuring that the gradients are propagated effectively throughout the layers. Moreover, the implementation of residual connections within the network architecture plays an important role in mitigating the gradient vanishing phenomenon commonly encountered in deep neural networks. This method facilitates the effective propagation of gradients during backpropagation by introducing shortcut connections that bypass one or more layers. As depicted in Figure 23 in the Appendix, the residual connections are strategically added to the network. These connections allow the flow of information and gradients to circumvent the nonlinear transformations, thereby preserving the strength of the gradient signal across multiple layers. It also adds regularization to the model, preventing overfitting by providing a form of parameter sharing across layers [23].

Since the real value of dual solutions is mostly lower than 0.01, the general loss function may not be suitable to generate a reliable value. The mean square error is set as the loss function at first. Even though the convergence of the loss function is smooth, the trained model may not generate reliable values for the four dual variables simultaneously. Then the loss function we considered is the summation of relative errors in the four outputs, i.e.,

$$Loss = \sum_i \left| \frac{\tilde{z}_i - z_i}{z_i} \right|. \tag{19}$$

Here $z_i$ is the target value while $\tilde{z}_i$ is the predicted value. By using Bayesian retrieval methods, we found the Adam optimizer best fits the training process.

The datasets we used contain the data from simulations with different configurations. To mitigate the model overfitted on specific models, we adopted the cross-validation technique. As shown in Figure 3, the datasets are divided into 5 different subsets for instance. For each fold, the network is trained with the training sets and evaluated with the remaining validation set. Such a process is repeated 5 times such that every subset has been tested as a validation set. By averaging the model performance over different

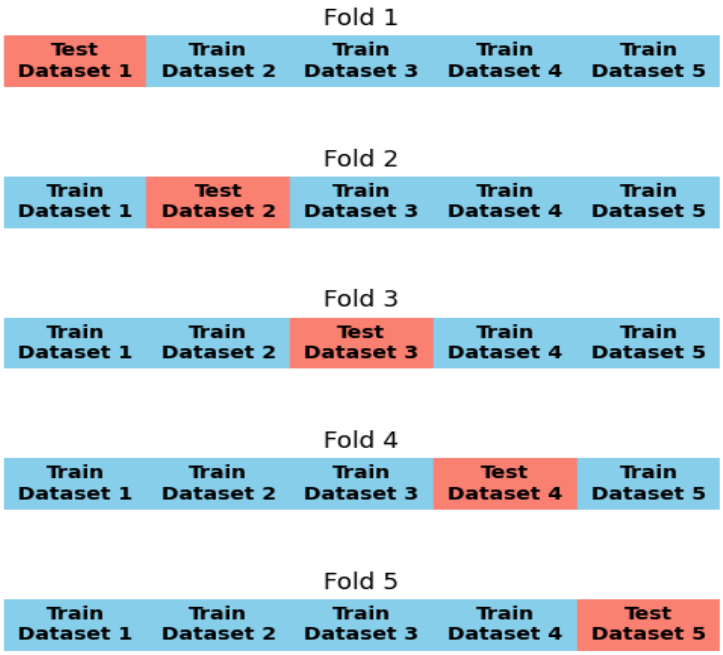


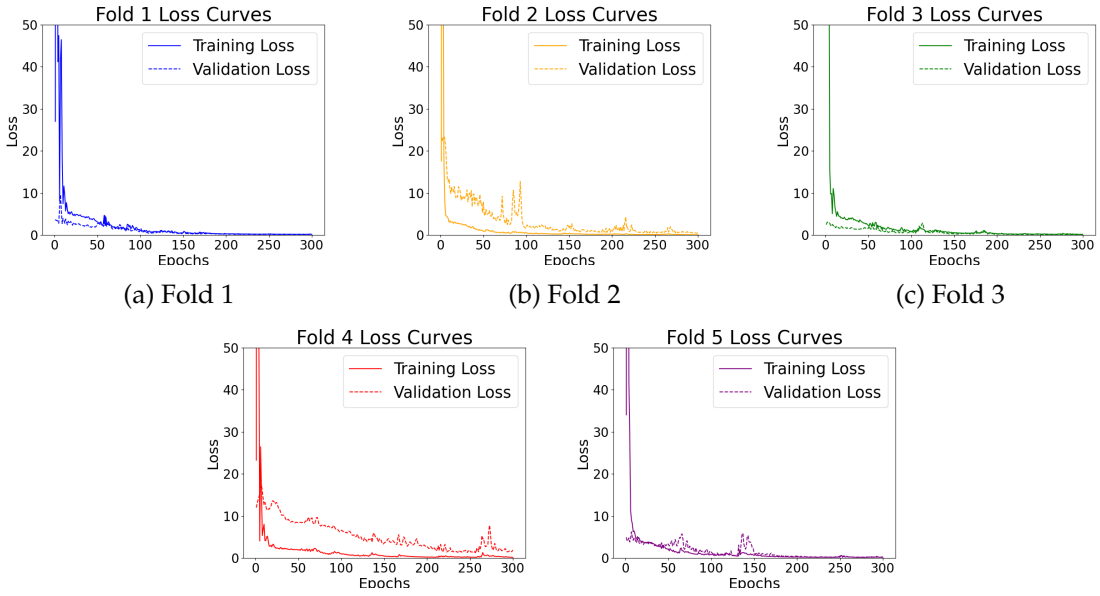

Figure 3: Schemetic of cross-validation.

subsets, we can reduce the variability and get a better understanding of how the model will perform on unseen data. Since the loss function adopted is a relative error, the The result of the training with this cross-validation process can be seen below. The behavior of this model can be seen in Figure 5. The figures indicate that the trained model can capture the structures of the dual solutions. Details about the practical calculation will be introduced in the numerical experiment section.

The result in Figure 4 is a test on the training for airfoil NACA0012 with 0.8 Mach number, 1.25° attack angle. The loss can converge to ideal precision when the iteration is processed. The Fold2 may not behave like other examples since the data size is not sufficiently large which will be remitted in a more complex data set.

## 3.2 Generation of datasets

The training datasets are as important as the model structure for machine learning projects. As described above, the equation (18) depicts the relation between the primal variables and dual variables, i.e., input-target pair for the learning process. Intuitively, the dual variables are predominantly influenced by the neighbor elements of the selected central. Then the input can be converted to the tensor as $reconstruction\_patch \times feature \times$




(a) Fold 1 (b) Fold 2 (c) Fold 3

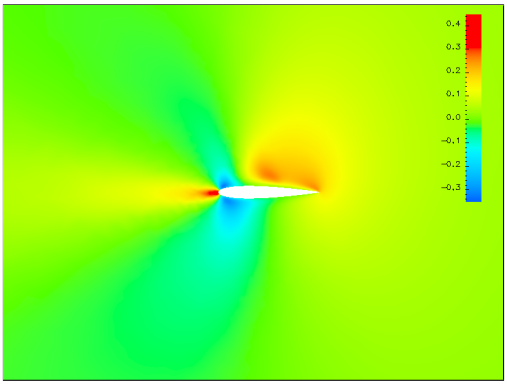
(d) Fold 4 (e) Fold 5

Figure 4: The subfigures (a) to (e) show the individual loss curves during the cross-validation process. The dataset only contains Mach number 0.8, attack angle 1.25°.

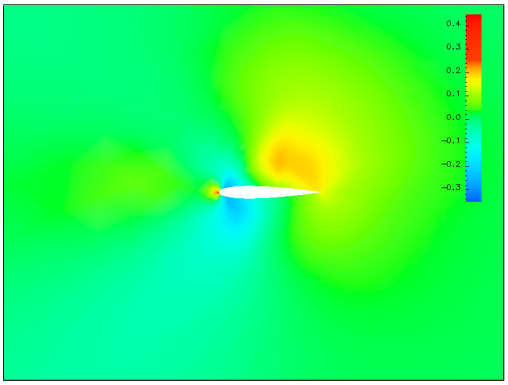
*dim*, generally $5\times3\times5$. Here feature includes the configurations, primal solutions, and primal residuals. For the training with various configurations, the reconstruction patch is only considered in a degenerate form, i.e., the tensor is only $1\times3\times5$. The performance of the model is still satisfactory since both primal and dual solutions are considered in first-order form. A higher dimensional reconstruction patch will be considered within a higher order numerical scheme like in [29]. Besides, the error estimate of the quantity of interest can be more accurate if the residuals of the dual equation have been taken into account [30]. However, the neural network and corresponding datasets should be further designed to support the calculation. A simplified case is considered with respect to the finite volume method in this work. The datasets are generated from the traditional GMG solver and saved as data files. The Newton-GMG solver we used contains a module that memorizes the cache information of specific elements, including location, Mach number, attack angle, and reconstruction patch. It not only helps to accelerate the simulation process but also makes it easy to load data for the learning process.

| **Mach Number** | 0.62 | 0.68 | 0.71 | 0.73 | 0.78 | 0.8 | 0.8 | 0.8 |
|---|---|---|---|---|---|---|---|---|
| **Attack Angle** | 1.29° | 0.82° | 0° | 1.05° | -2.27° | 1.25° | 0° | -1.25° |

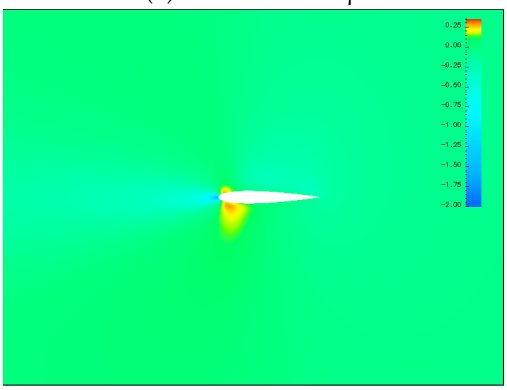
Table 1: Parameters of Data Sets



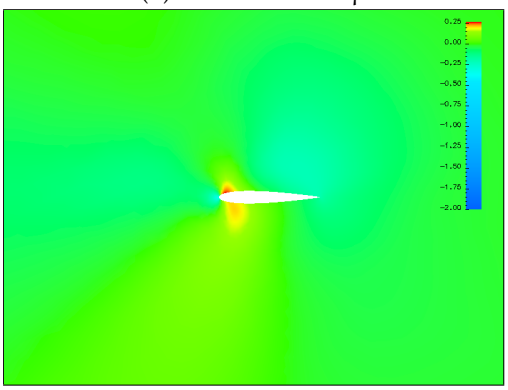
(a) GMG solver - $\rho$



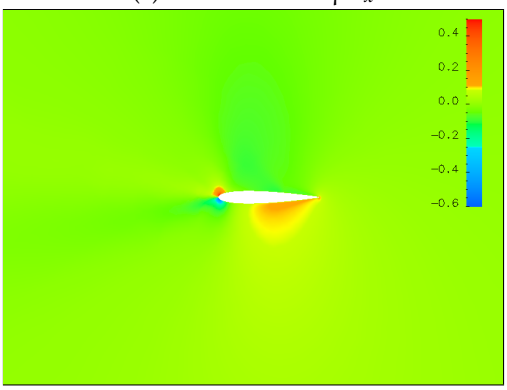
(b) CNNs solver - $\rho$



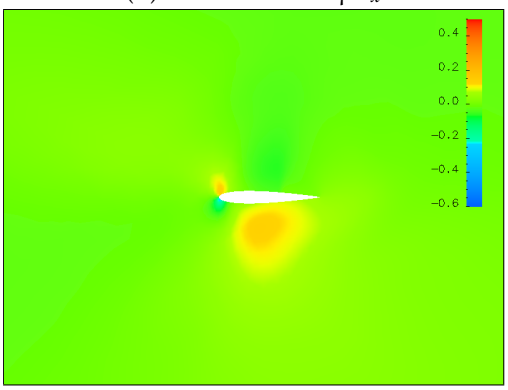
(c) GMG solver - $\rho u_x$



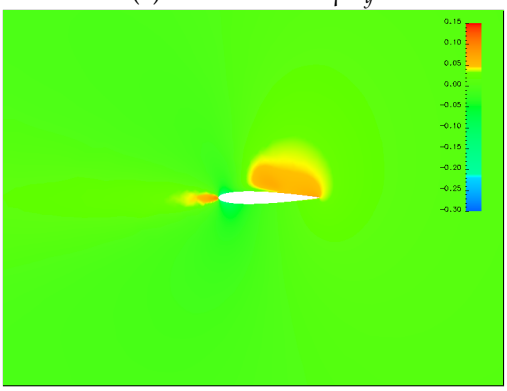
(d) CNNs solver - $\rho u_x$



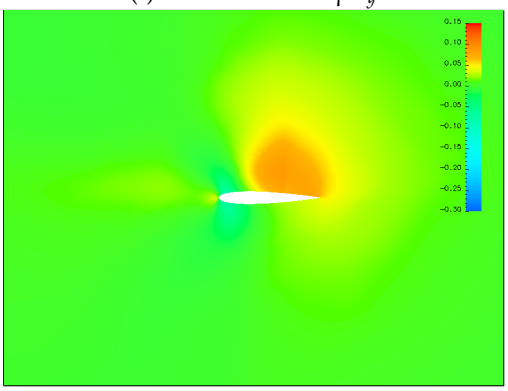
(e) GMG solver - $\rho u_y$



(f) CNNs solver - $\rho u_y$



(g) GMG solver - $E$



(h) CNNs solver - $E$

Figure 5: Left column: Dual variables generated by GMG solver; Right column: Dual variables generated by CNNs solver; Mach number 0.8, attack angle 1.25°.

To develop a model capable of predicting dual solutions across varying Mach numbers and attack angles, we collated data generated by the GMG solver with the parameters in Table 1. The cumulative size of all files amounts to 212 MB. The training and simulation are conducted on the device with the following parameters:

- CPU: Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz 18 Cores.
- RAM: 64G.
- GPU: NVIDIA A100 Tensor Core GPU.
- RAM: 80G.

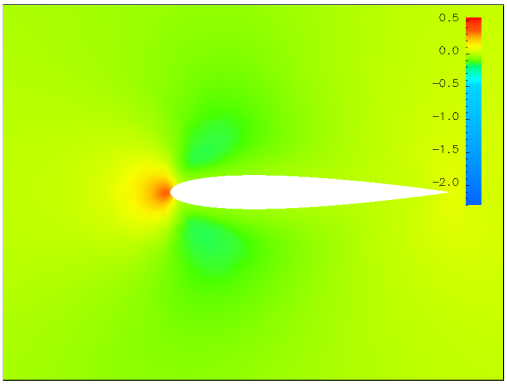
During the training process, the data files are processed together with Python and then converted to tensor form. The batch size is usually set as 16384. It should be noted that the quality of data is also a significant factor that influences the prediction of dual solutions. If the traditional GMG solver can not supply data with satisfactory quality, the performance of CNNs-based mesh adaptation will be influenced sharply.

### 3.3 Dual consistency for training datasets in CNNs

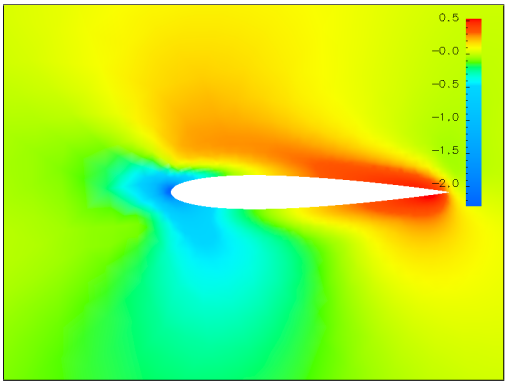
In [1], we analyzed the importance of dual consistency for the DWR-based adaptation in Newton-GMG solver. If the dual consistency is not satisfied, waste on the computational resources will occur [9, 11, 31]. Worse still, the adaptation may lead to a mesh with target function with lower error. We test the training data from NACA0012 with 0.5 Mach number, 0 attack angle, and zero normal velocity boundary condition. In this study, we validate that if the training datasets are obtained from a dual-inconsistent solver, the performance of CNNs prediction will perform badly as well. The convolutional neural networks can not capture the relation between dual and primal solvers. Then the training process misunderstands the distribution, leading to even worse dual variables. Conversely, in the datasets obtained from a dual-consistent solver, the smooth and symmetric properties can be preserved in this model as shown in Figure 6. Then, it indicated that a dual-consistent DWR-based solver should be constructed at first. Based on that, an efficient CNNs form dual solver can be expected.

It is worth noting that the datasets of dual solutions we obtained are from the GMG solver. As the dual-inconsistent scheme may pollute the adaptation around the boundary, some unexpected singularities may occur. Then the dual equations may generate a linear system with a loss of regularization. In order to resolve this issue, a regularization term was added to the dual equations, leading to a GMG solver with a more stable convergence rate. To handle more complicated boundary conditions, the boundary modification techniques proposed by Hartmann [32] should be introduced.

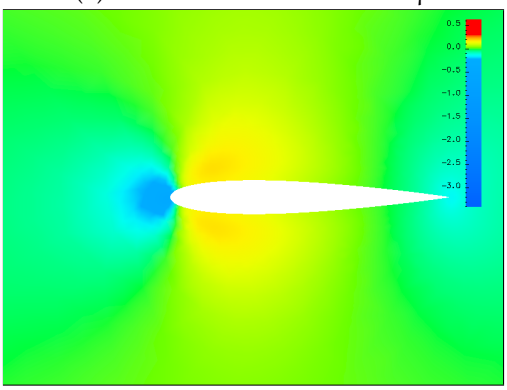
Dual consistency is a critical aspect of the DWR-based adaptation method, and the network is also rigorously designed to test this property. The outputs are designed as four 64 to 1 fully connected layers instead of a single 64 to 4 fully connected layer for



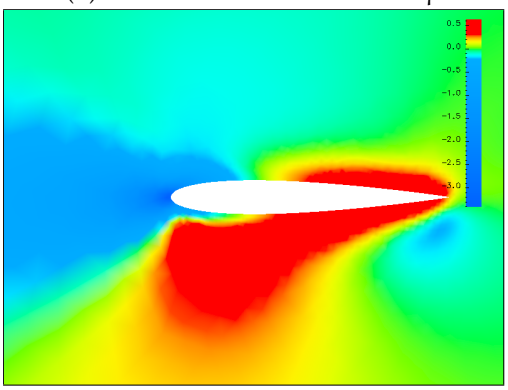
(a) Consistent CNNs solver - $\rho$



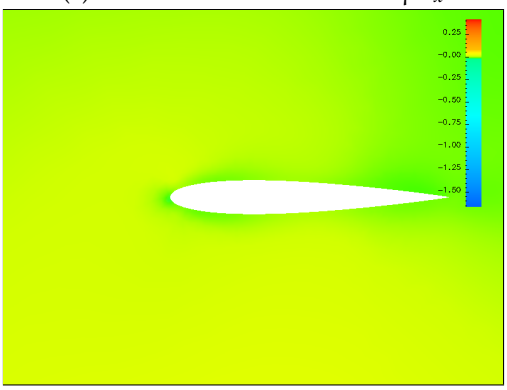
(b) Inconsistent CNNs solver - $\rho$



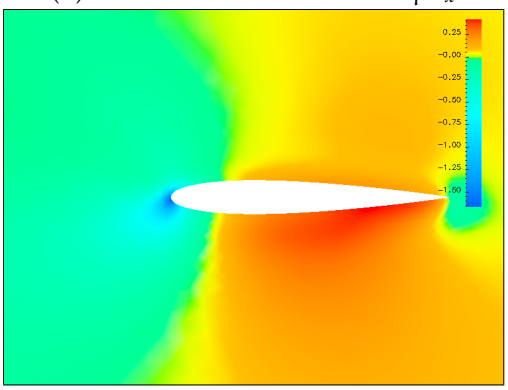
(c) Consistent CNNs solver - $\rho u_x$



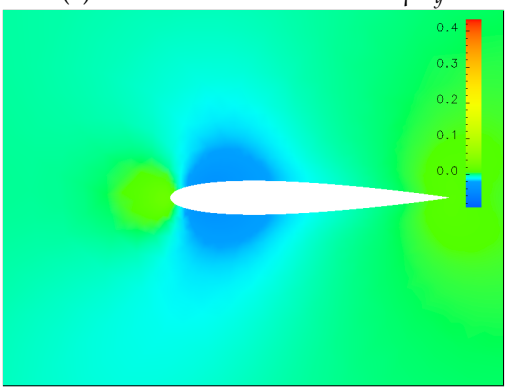
(d) Inconsistent CNNs solver - $\rho u_x$



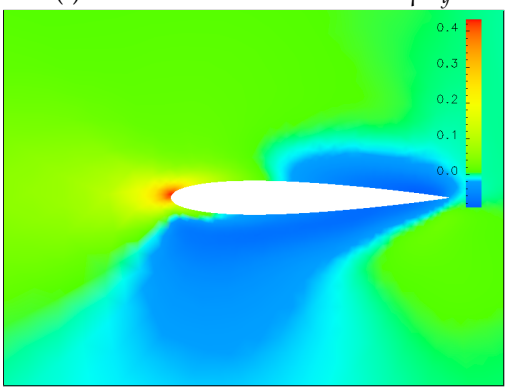
(e) Consistent CNNs solver - $\rho u_y$



(f) Inconsistent CNNs solver - $\rho u_y$



(g) Consistent CNNs solver - $E$



(h) Inconsistent CNNs solver - $E$

Figure 6: The dual variables generated by CNNs solver. Left column: dual-consistent datasets; Right column: dual-inconsistent datasets; Mach number 0.5, attack angle 0°

such a purpose as the outputs are mathematically independent. Then the network is designed to minimize the interference of unrelated information and maintain the mathematical independence among the variables. Additionally, it is worth mentioning that the trained model has the potential to be extended further. For instance, after obtaining the dual variables, it could be employed to directly calculate indicators on the element level. However, focusing on dual consistency, we choose to utilize the current network architecture.

# 4 Acceleration Strategy

The trained model above is saved as the open neural network exchange(ONNX) file form so that the C++ library maintained by our group AFVM4CMD can invoke the dual solver. AFVM4CFD is a sophisticated solver that can efficiently handle steady Euler equations. This solver incorporates modules such as k-exact reconstruction, Bezier curves, and geometrical multigrid techniques, thereby offering robust numerical solutions.

With the introduction of the trained model, which substitutes the conventional dual solver, the DWR-based mesh adaptation process is notably expedited. To further bolster performance and accelerate the simulation process, we implemented additional enhancements within this framework. The subsequent subsections will elaborate on this module that significantly optimizes the numerical experimentation process.

## 4.1 An Automatic Adjustment of Tolerance in an h-adaptive process

Even though the DWR-based mesh adaptation method generates high-quality indicators, formulating an appropriate tolerance presents challenges. Different from the general adaptation issues, setting the tolerance too low in DWR-based adaptation can result in subsequent steps resembling a uniform refinement. This disobeys the initial intent of economically solving the quantity of interest. Conversely, if the tolerance is set too high, the adaptation strategy may fail to capture the region that influences the target function significantly. Motivated by the decreasing threshold method proposed in [20], the indicators are plotted for analysis and a straightforward algorithm is proposed to adjust the tolerance dynamically. The indicator we adopted is the multiplication of dual solutions and residuals, i.e.,

$$\eta_{T_i} = \sum_{T_{ij}} |(\mathbf{z}_h)^T \mathcal{R}_h(\mathbf{u}_h^H)|, \tag{20}$$

where $T_{ij}$ is the subelements $T_i$. We recorded the indicators statistically in three consecutive refined meshes. The distribution concentrated on the left end. Then we tested the distribution function with the Kolmogorov-Smirnov method. The result shows that the

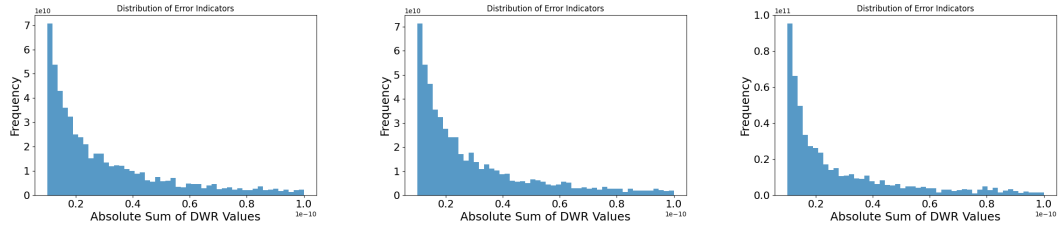



Figure 7: Indicators in three consecutive refined meshes

distribution may approximate Weibull distribution with KS-stat equals 0.98327 in this example.

However, when the configurations change, the distribution may approximate the gamma distribution in some circumstances. Then it is not robust to choose the tolerance with the expression of the distribution function. Alternatively, we designed different strategies. The indicators in each element of a given mesh are sorted at first. Then, tolerance is calculated from the indicators. For example, a fixed proportion of the element size can be selected, which leads to a steady growth of element size during the mesh adaptation process. In addition, the index which denotes the average number with a variation of significance can be calculated, which resembles the method in [19] if the distribution is known in advance. In this model, the tolerance can be calculated with the mathematical expression of Weibull distribution. In these ways, with the mesh adaptation processed, the tolerance can be adjusted dynamically. Since we are dealing with different configurations, the first case is the most robust which is applied in the simulation in this research.

To test the dynamic tolerance strategy, numerical experiments are conducted in this part. We compared the result with four times uniformly refined mesh. After introducing the dynamic tolerance strategy, the number of elements is growing steadily, which means that the corresponding growth of the number of elements can be implemented as expected to achieve the required error. From the result in Table 2, if the constant tolerance is set high, the adaptation may not capture the area that influences the target functional mostly, leading to a rough precision. Conversely, if the constant tolerance is set low, the adaptation at the beginning stage may behave like a uniform refinement, which takes a long time for the calculation of dual equations. However, the dynamic tolerance strategy preserves a stable growth rate, and an expected precision while saving time for generating dual solutions compared with the lower constant tolerance strategy. As shown in Figure 8, the dynamic tolerance strategy performs well for generating the mesh which calculates the target functional precisely.

From the above post-processing technique, we developed a more efficient framework for solving the quantity of interest with the CNNs solver. The algorithm is organized as follows:

| Tolerance | | Adaptation | | | |
|---|---|---|---|---|---|
| | | 1st | 2nd | 3rd | 4th |
| Dynamic | Elements | 11130 | 33945 | 104985 | 329901 |
| | Refined Rate | 0.69 | 0.68 | 0.70 | 0.71 |
| | Tolerance | $4.70\times10^{-11}$ | $2.24\times10^{-11}$ | $9.77\times10^{-12}$ | $1.50\times10^{-12}$ |
| | Time for Dual | 28 | 111 | 420 | 1811 |
| | Error | 0.01329 | 0.00305 | 0.00074 | 0.00010 |
| Constant 1 $3.0\times10^{-11}$ | Elements | 11961 | 31581 | 75363 | 148020 |
| | Refined Rate | 0.76 | 0.55 | 0.46 | 0.32 |
| | Time for Dual | 28 | 120 | 377 | 1186 |
| | Error | 0.01331 | 0.00312 | 0.00157 | 0.00133 |
| Constant 1 $3.0\times10^{-12}$ | Elements | 14529 | 50781 | 134772 | 298170 |
| | Refined Rate | 0.99 | 0.83 | 0.55 | 0.40 |
| | Time for Dual | 28 | 150 | 664 | 2442 |
| | Error | 0.01332 | 0.00294 | 0.00069 | 0.00024 |

Table 2: Comparison of dynamic and constant tolerance Strategies. The "Time for Dual" is recorded as wall time with unit "seconds".

**Algorithm 2:** DWR for one-step mesh refinement

**Data:** Initial $\mathcal{T}_H$, $TOL$
**Result:** $\mathcal{T}_h$
Using the Newton-GMG to solve $\mathcal{R}_H(\mathbf{u}_H)=0$ with residual tolerance $1.0\times10^{-3}$;
Convert the element information to Tensors' form;
Generating the dual solutions with the trained ONNX model;
Calculate the error indicator for each element;
Select the dynamic tolerance for adaptation;
**while** $\mathcal{E}_{K_H} > TOL$ *for some* $K_H$ **do**
  Adaptively refine the mesh $\mathcal{K}_H$ with the process in [8];
**end**

There are many different methods that can expedite the calculation of dual solvers. For example, the BiCG method in [29,33] is adopted to calculate the dual and primal problems at once and the multiple precision [10] is adopted to accelerate the dual solver. In our framework, based on the Newton-GMG method, the calculation of the primal equation and the dual equation is independent. Then the CNNs dual solver is constructed to accelerate the mesh adaptation. It should be noted that the Algorithm 1 of GMG dual solver and the constructed Algorithm 2 of CNNs dual solver share the same process of calculating the primal equation with CPU. In order to guarantee the fairness of comparison, the trained ONNX model is also invoked and executed by the CPU. However, the

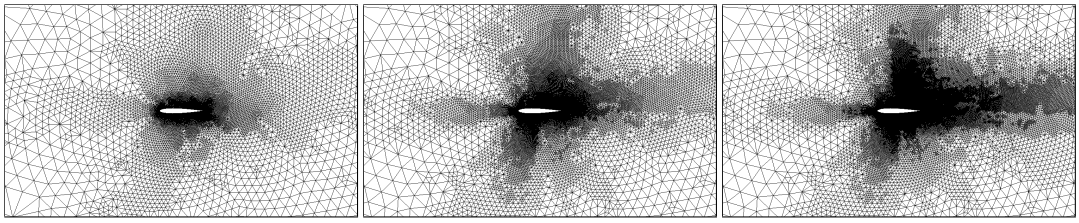


Figure 8: Consecutive refined meshes with dynamic tolerance strategy. NACA0012, Mach number 0.8, attack angle 1.25°.

model can be further accelerated by executing on GPU.

## 4.2 OpenMP parallelization of the algorithm

Parallel computing enables the simultaneous execution of computations, leading to significant reductions in time, especially for the CFD domain where simulations can be extremely time-consuming. Among these, OpenMP is a widely used parallel programming model that greatly facilitates the execution of tasks by dividing them into multiple threads that can run concurrently. However, it requires a careful design of the algorithm to avoid potential pitfalls. In this part, we are mainly concerned with the scalability of different parts of the algorithm.

The primal solver, Newton-GMG solver, contains reconstruction, update cell average, and geometrical multigrid module for each Newton iteration step. If the reconstruction patch should be built in each Newton iteration step, it will consume too much time. Then we build a cache module to reserve the information of the reconstruction patch in advance so that the identical information will not be calculated redundantly. Since the update of the cell average is independent for different elements, it exhibits a satisfactory performance of parallelization.

The speedup in parallel computing is generally defined as

$$Speedup = \frac{t(1)}{t(N)}, \tag{21}$$

where $t(1)$ is the time for running the algorithm with one processor and $t(N)$ for $N$ processors respectively.

Theoretically, the time used for parallel computing will be halved as the number of threads doubles. Then the ideal value of speedup should be $N$ when there exist $N$ processors. However, the performance of parallel computing may not behave as expected due to the complexity of the algorithm. To compare the efficiency of the parallelization in our framework, the problem size remains constant in this section. The experiments in this part are all conducted on RAE2822 airfoil whose Mach number is 0.729, attack angle 2.31° on the mesh with 232,704 elements.

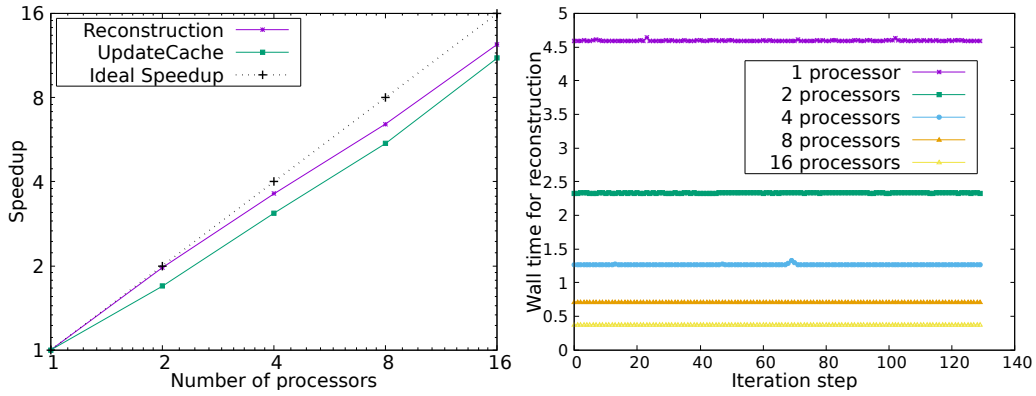



Figure 9: Left: Mean time spent on modules during the Newton iteration step with the different number of threads. Right: Time spent on the reconstruction during the whole Newton iteration process.

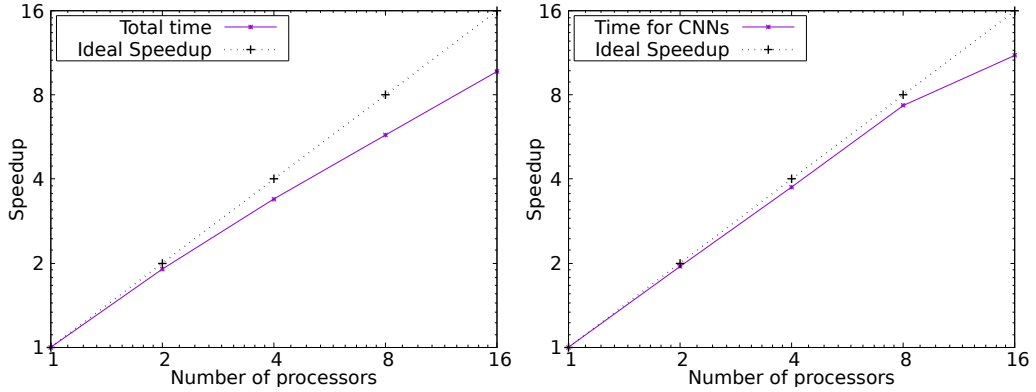



Figure 10: Left: Total time spent for the primal solver with the different number of threads. Right: Total time spent on the dual solver with the different number of threads.

The performance can be seen in Figure 9. It illustrates that the reconstruction part and update part are well suited for parallelization. With the increase in the number of threads, time spent on the calculation has been saved significantly. However, the lower-upper symmetric Gauss-Seidel iteration in each Newton iteration step is not suitable for parallelization, which cannot be optimized even if more threads are used. Then the scalability for the whole process of the primal solver is limited. Figure 10 illustrates the effect of scalability of primal solver and dual solver. It shows that parallel computing has significantly enhanced computational efficiency, and the scalability of the dual solver further demonstrates that the method of using neural networks to solve dual solutions can greatly accelerate the calculation. The initialization of the trained ONNX model influenced the scalability and will be improved upon in future work. Even though, paral-

lelization is still an effective technique for accelerating the simulation of our algorithm. Then the following numerical experiments are all conducted with parallelism.

# 5 Numerical Experiments

In this section, we will show the performance of CNNs for generating dual solutions with significant time savings. Then the generalizations of the model are demonstrated in three different aspects. Firstly, even when we deal with the adaptation with data not included in the training sets, the behavior still preserves well till the expected precision. Secondly, the model is trained with different Mach numbers and attack angles. The configurations beyond the training categories still perform well with the trained model. Thirdly, since the framework is constructed on an unstructured mesh, it is very important to show that the model works well for a more complicated geometry.

## 5.1 Time consumption of CNNs dual solver

The most important reason we adopt the CNNs solver is its significant acceleration in generating dual solutions. The trained models are tested on the configurations shown in Table 3. In each adaptation step, the GMG solver, and CNNs dual solver generate high-quality meshes which can derive the quantity of interest with comparable precision. The data reveals that the CNNs dual solver can dramatically reduce computational time, particularly as the adaptive process progressively advances. In the Three-Airfoils model, which contains much more complicated geometries, the CNNs solver shows great potential for time-saving.

| Configurations | Wall time for dual in different step (seconds) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1st | | 2nd | | 3rd | | 4th | |
| | CNNs | GMG | CNNs | GMG | CNNs | GMG | CNNs | GMG |
| Case 1 | 9 | 28 | 25 | 111 | 81 | 420 | 295 | 1811 |
| Case 2 | 9 | 29 | 28 | 114 | 93 | 373 | 382 | 2253 |
| Case 3 | 9 | 31 | 27 | 125 | 95 | 519 | 387 | 2708 |
| Case 4 | 9 | 32 | 25 | 113 | 78 | 411 | 258 | 1544 |
| Case 5 | 422 | 2324 | 1145 | 9286 | 6521 | 64613 | - | - |

Table 3: Comparision of CNNs solver and GMG solver with wall time for dual in different adaptation steps. Case 1 for NACA0012, 0.8 Mach number, 1.25° attack angle, Case 2 for NACA0012, 0.76 Mach number, 1.05° attack angle, Case 3 for NACA0012, 0.82 Mach number, -1.65° attack angle, Case 4 for RAE2822, 0.729 Mach number, 2.31° attack angle and Case 5 for 3-NACA0012, 0.8 Mach number, 1.25° attack angle.

## 5.2 Generalization and model performance

### 5.2.1 Generalization on the training set

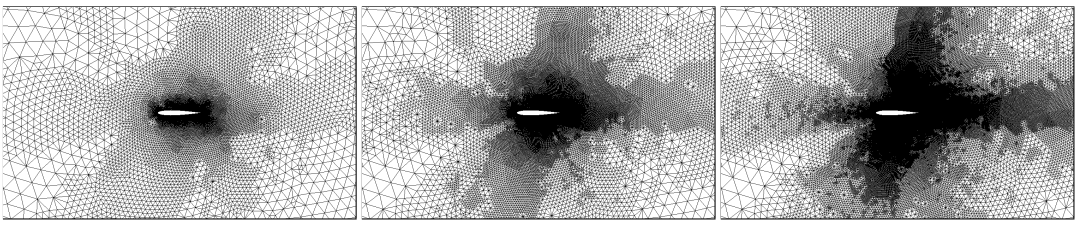


Figure 11: Consecutive refined meshes with CNNs dual solver. NACA0012, Mach number 0.8, attack angle 1.25°.

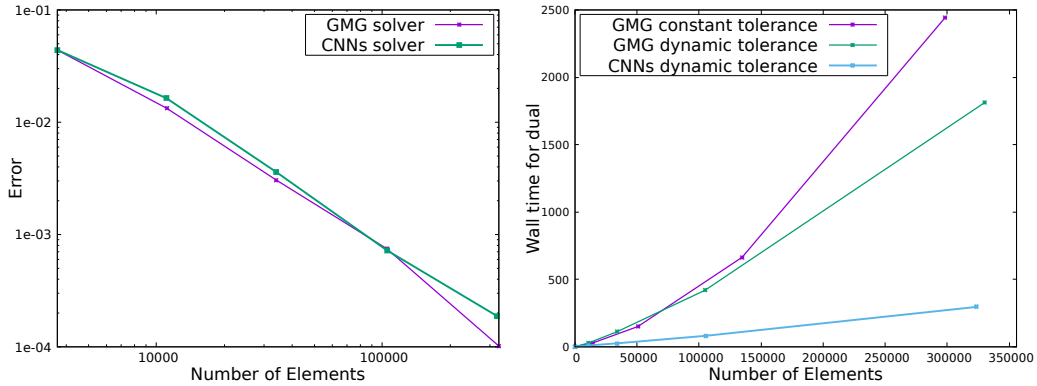



Figure 12: Comparison of error and time cost with the dual solver of GMG form and CNNs form. NACA0012, Mach number 0.8, attack angle 1.25°.

In Figure 5, the CNNs captured the main structure of dual solutions. With the substituted CNNs form the dual solver, the adaptation gets processed steadily. In Figure 11, it is shown that the dual solver of CNNs form can generate a mesh that balances the dual equations and residuals similar to the GMG solver. The result can be seen from Figure 12 that the error of CNNs dual solver is comparable to the GMG solver. Here we use the mesh with 4 times uniform refinement with 930,816 elements to generate a reference value, which is 0.026201. However, comparing the time for generating the refined mesh, the CNNs form can save time by an order of magnitude. The results in Figure 12 indicate that time spent on the GMG solver grows exponentially, while the CNNs solver grows linearly. This can be explained that the CNNs solver calls the trained model element by element, then the time complexity of the CNNs solver is just $O(n)$, where $n$ is the number of elements. It is worth noting that to train the CNNs dual solver, we only use the datasets from the mesh with one and two times uniform refinement. In other words, the

third and fourth-time refinement is the prediction beyond the training set. As a result, the trained CNNs solver is capable of generating credible dual solutions in this framework.

### 5.2.2 Generalization on the configurations

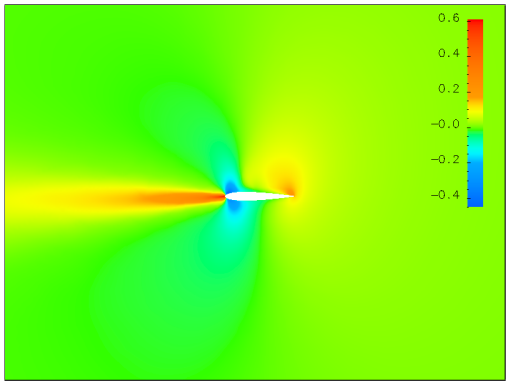
In a shape-optimal design problem, the configurations should be changed for numerous experiments. If the CNNs model needs to be trained every time, the time spent on the training is still expensive. Nonetheless, since the configurations can be treated as input to the CNNs, we can train the model with various configurations so that it can handle different simulations. The CNNs architecture we built is capable of extracting the features between different Mach numbers and attack angles. We tested the trained model with the following configurations:

- NACA0012, 0.76 Mach, 1.05° attack angle ;
- NACA0012, 0.82 Mach, -1.65° attack angle ;
- RAE2822, 0.729 Mach, 2.31° attack angle.

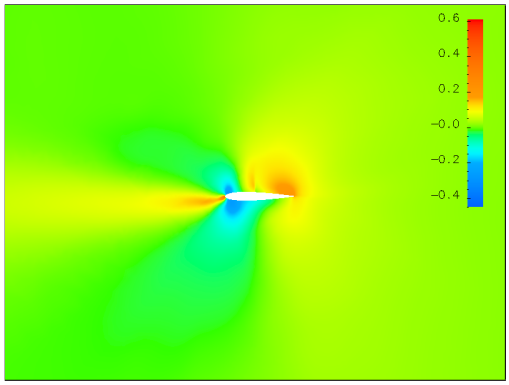
Results from Figure 15 demonstrate that, compared to the models in Figure 4, the newly trained models exhibit more significant fluctuations in loss reduction. This increased volatility is attributable to the larger dataset used in this training, presenting greater challenges. Despite these challenges, the final convergence outcomes are satisfactory, resulting in a trained model that is more versatile, and capable of handling a broader range of Mach numbers and attack angles effectively. Then we test the experiment whose configurations are not included in the training sets. Firstly, the Mach number is set at 0.76 while the attack angle is set at 1.05. The distribution of dual variables can be seen in Figure 13. Similarly, as seen in Figure 14, the CNNs solver generates mesh which solved the target functional in a comparable error with the GMG solver. However, the CNNs solver saves time sharply in this experiment.

For fear that the convolutional neural networks may misunderstand the relation between primal and dual solver, we shall apply the model for other configurations. According to our experience, if the training sets are not sufficient enough, the CNNs dual solver may focus the refinement area on the top part of the airfoil. Then, we used the same ONNX model trained above to configurations with minus attack angle. Here we test the simulation with Mach number 0.82, and attack angle $-1.65$. In Figure 16, the trained model performs well on the different configurations, showing that the ability of our CNNs form dual solver can detect the main region for solving the target functional precisely. The precision and time spent for the adaptation are shown in Figure 17. The time spent on CNNs dual solver saves time for an order of magnitude while the precisions are comparable, which further verifies the reliability of our CNNs solver.

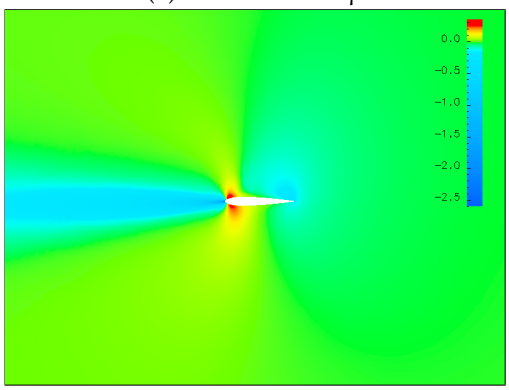
Moreover, the trained model is tested on the RAE2822 airfoil. It demonstrates the model also performs satisfactorily since the precision is comparable to the GMG solver while saving time significantly as shown in Figure 18. The consecutive refined meshes



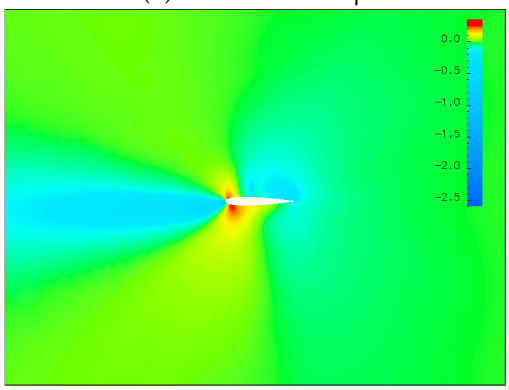
(a) GMG solver - $\rho$



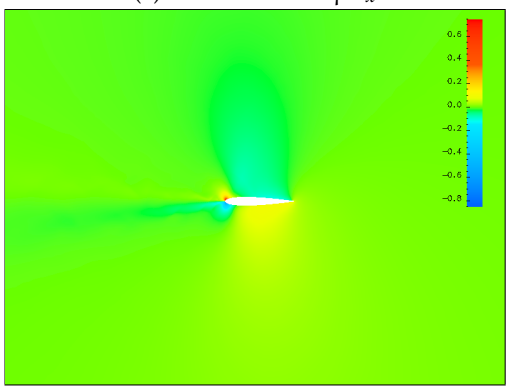
(b) CNNs solver - $\rho$



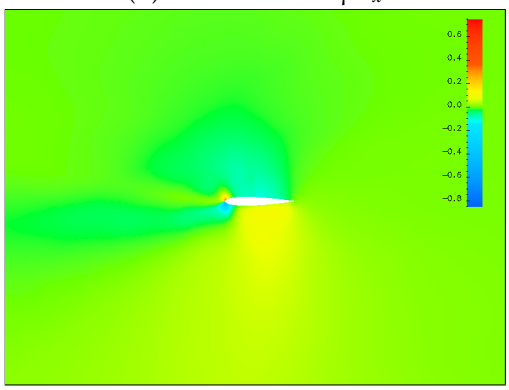
(c) GMG solver - $\rho u_x$



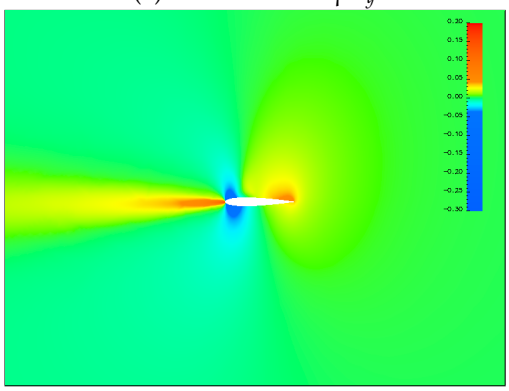
(d) CNNs solver - $\rho u_x$



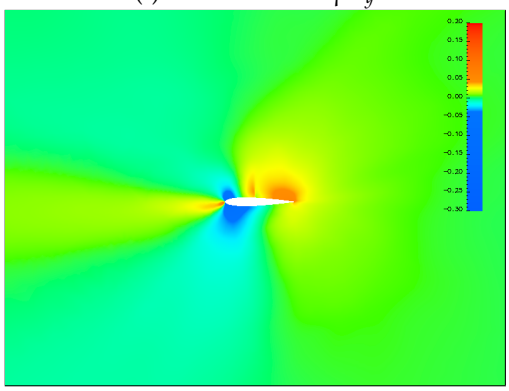
(e) GMG solver - $\rho u_y$



(f) CNNs solver- $\rho u_y$



(g) GMG solver - $E$



(h) CNNs solver - $E$

Figure 13: Left column: Dual variables generated by GMG solver; Right column: Dual variables generated by CNNs solver; Mach number 0.76, attack angle 1.05°.

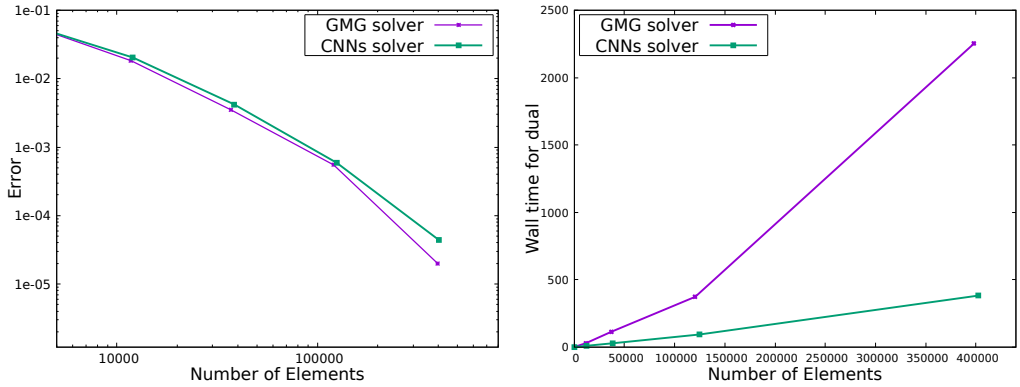
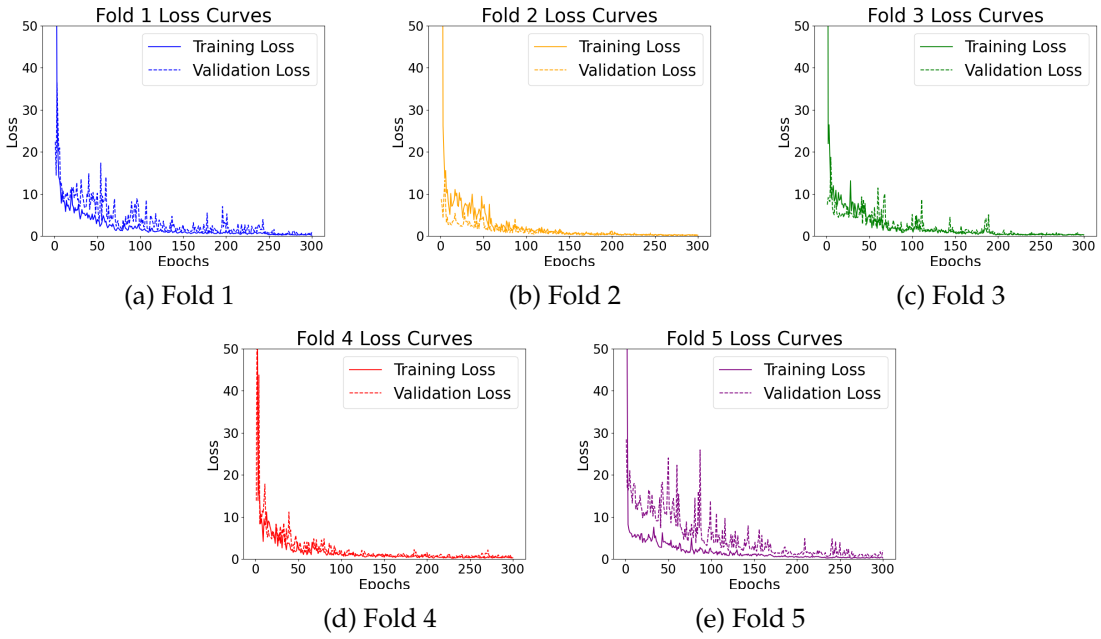



Figure 14: Comparison of error and time cost with the dual solver of GMG form and CNNs form. NACA0012, Mach number 0.76, attack angle 1.05.

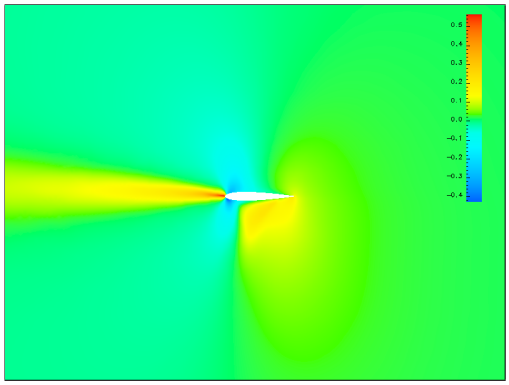



(a) Fold 1 (b) Fold 2 (c) Fold 3

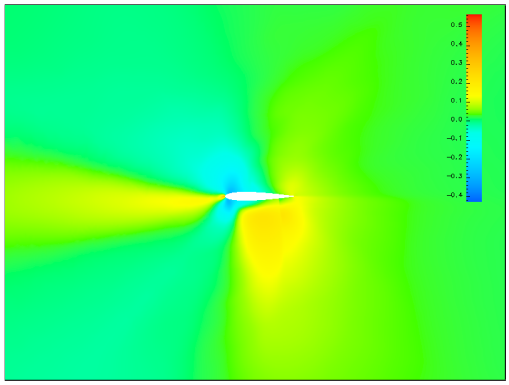
(d) Fold 4 (e) Fold 5

Figure 15: The subfigures (a) to (e) show the individual loss curves during the cross-validation process. Datasets contain Mach number and attack angle from Table 1.

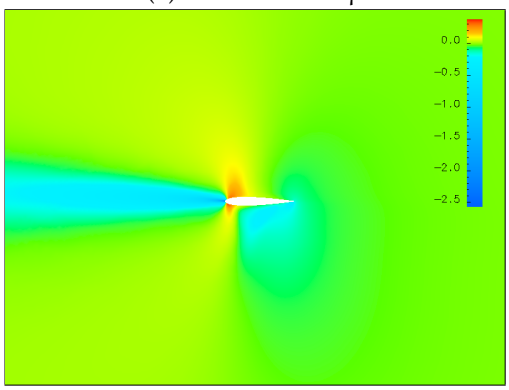
with the CNNs dual solver are illustrated in Figure 19. The dual solver can also detect the main regions that influence the target functional most.



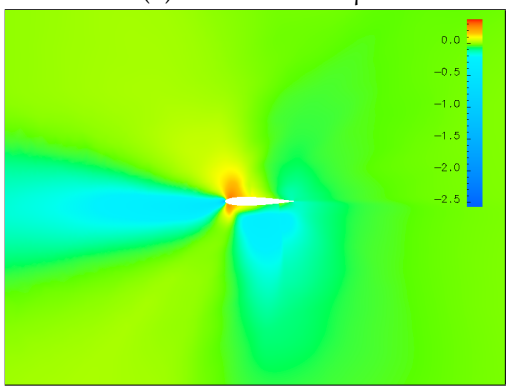
(a) GMG solver - $\rho$



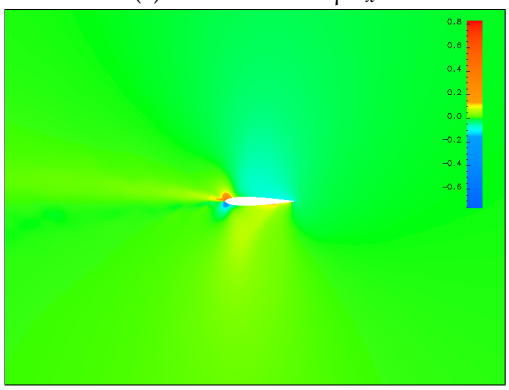
(b) CNNs solver - $\rho$



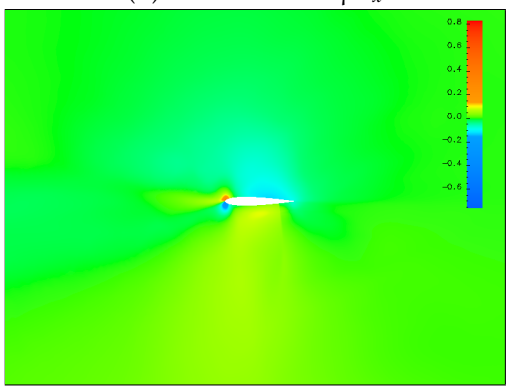
(c) GMG solver - $\rho u_x$



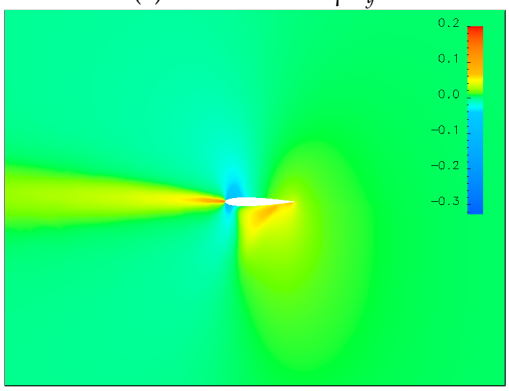
(d) CNNs solver - $\rho u_x$



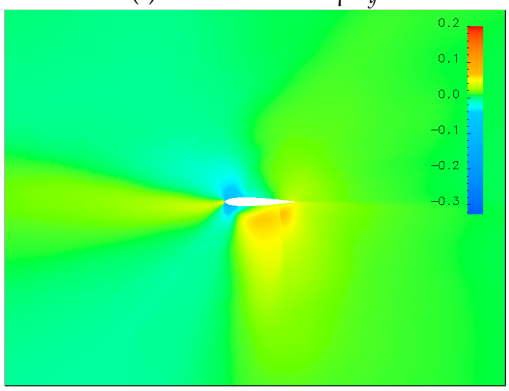
(e) GMG solver - $\rho u_y$



(f) CNNs solver - $\rho u_y$



(g) GMG solver - $E$



(h) CNNs solver - $E$

Figure 16: Left column: Dual variables generated by GMG solver; Right column: Dual variables generated by CNNs solver; Mach number 0.82, attack angle -1.65°.

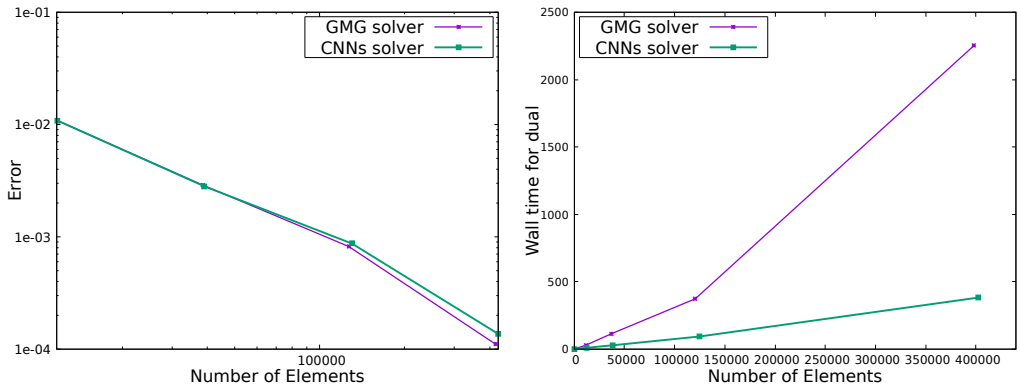



Figure 17: Comparison of error and time cost with the dual solver of GMG form and CNNs form. NACA0012, Mach number 0.82, attack angle -1.65.

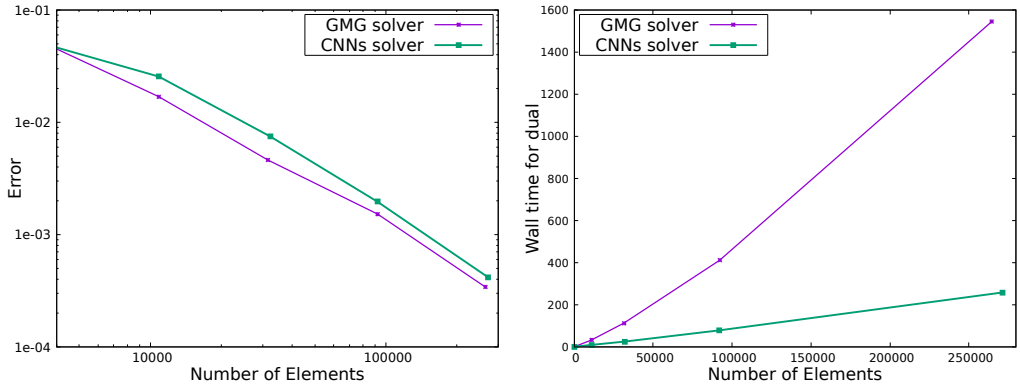



Figure 18: Comparison of error and time cost with the dual solver of GMG form and CNNs form. RAE2822, Mach number 0.729, attack angle 2.31.

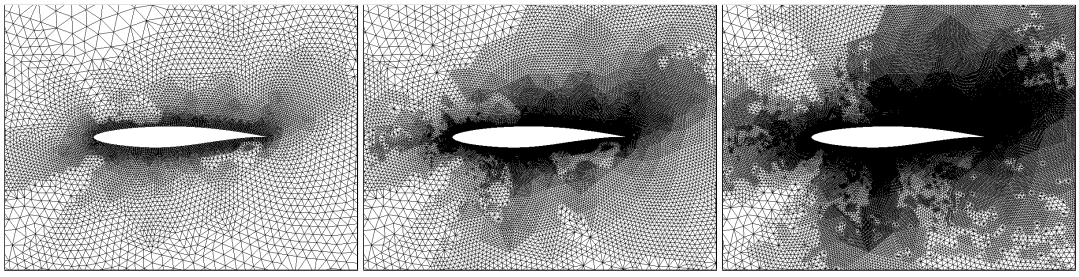


Figure 19: Consecutive refined meshes with CNNs dual solver. RAE2822, Mach number 0.729, attack angle 2.31°.

### 5.2.3 Generalization on the multiple airfoils

The benefit of our dual solver is the saving on the time for adaptation while the cost for the training is acceptable. In the training process, the dual consistency makes the model more reliable. Moreover, the attention mechanism is introduced in this work. It can still work well for multi-airfoil problems even on the unstructured mesh. In this part, we will show the feasibility of handling complicated geometry.

We chose a model with 3 NACA0012 airfoils, and the target functional is the drag of the leading airfoil. According to the simulation in Figure 21, the CNNs form dual solver captured the main region and focused on the leading airfoil.

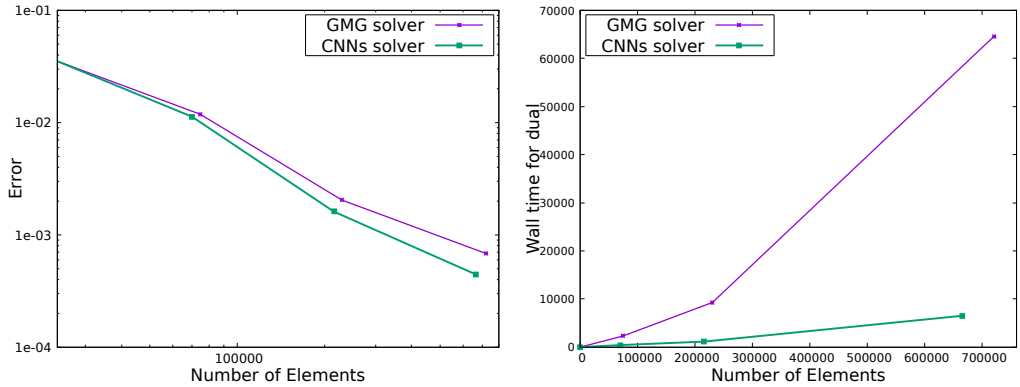



Figure 20: Comparison of error and time cost with the dual solver of GMG form and CNNs form. Three NACA0012 airfoils, Mach number 0.8, attack angle 1.25°.

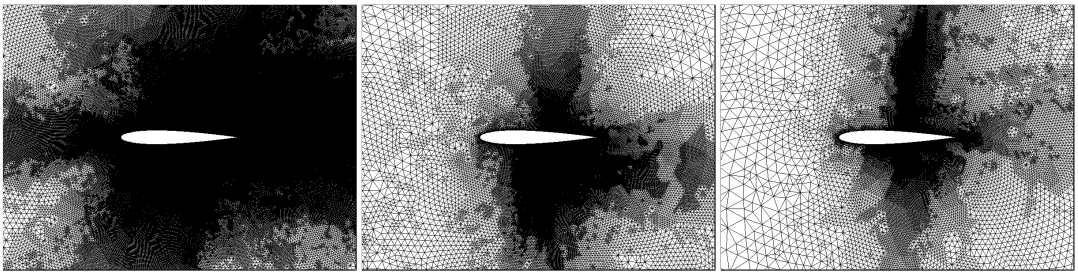
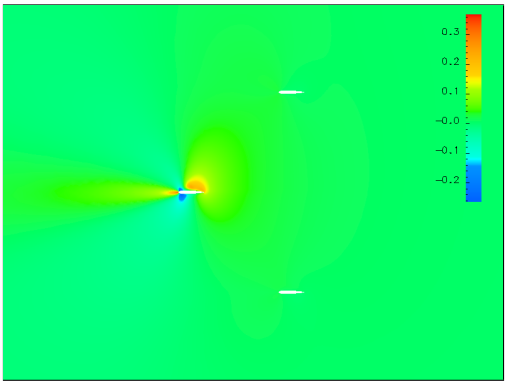
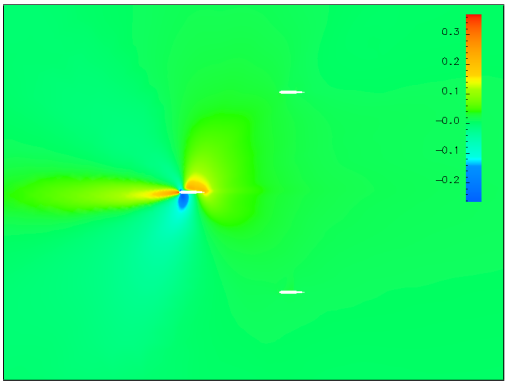


Figure 21: Meshes around the three different airfoils; Left: the leading airfoil; Middle: the upper airfoil; Right: the below airfoil.

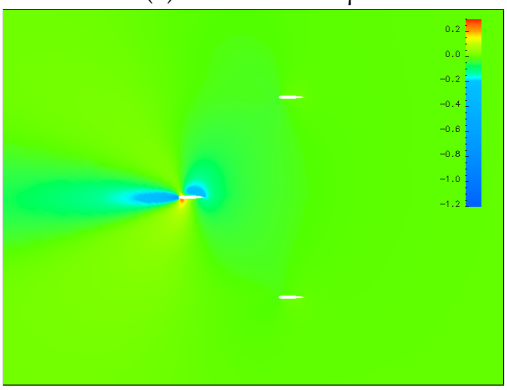
It preserves a comparable precision similar to the GMG solver while saving time sharply as seen in Figure 20. The initial mesh for this model is 23,466. So the uniform refinement for this model takes a significant time for calculation. Then it is not suitable to compute the reference value with 5 times uniform refinement. For this sake, we only conduct the adaptation 3 times. As mentioned above, the training only needs data from the first two refinements. It will not cost too much computational resources. In this



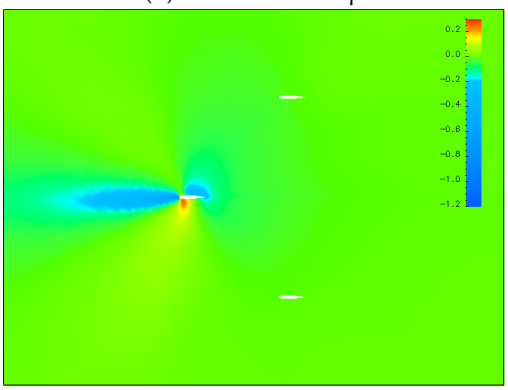

(a) GMG solver - $\rho$



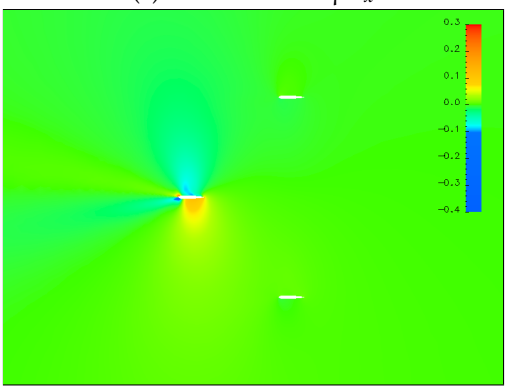

(b) CNNs solver- $\rho$



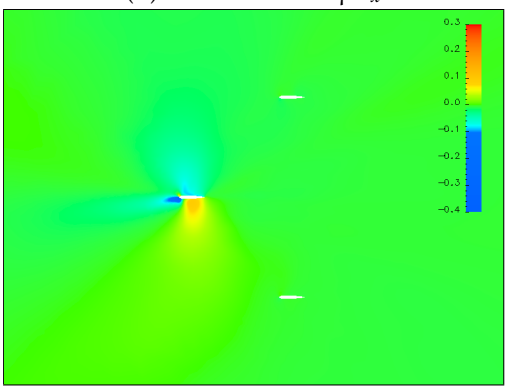

(c) GMG solver - $\rho u_x$



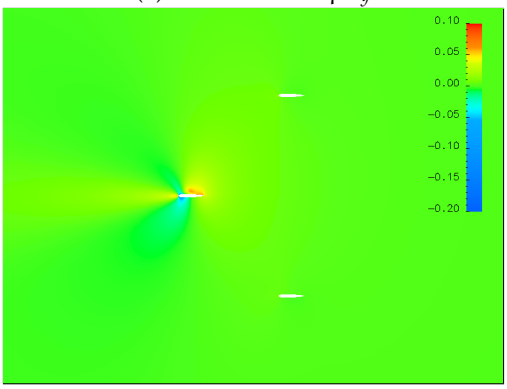

(d) CNNs solver - $\rho u_x$



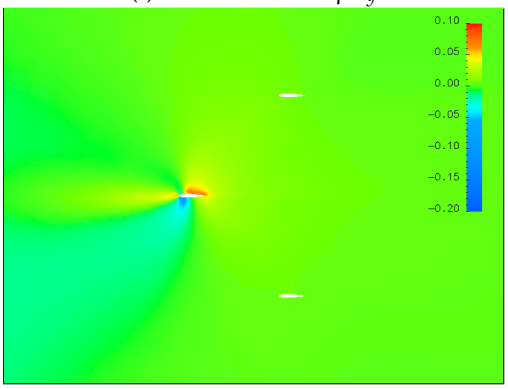

(e) GMG solver -$\rho u_y$




(f) CNNs solver - $\rho u_y$




(g) GMG solver - $E$




(h) CNNs solver- $E$

Figure 22: Three Airfoils. Left column: Dual variables generated by GMG solver; Right column: Dual variables generated by CNNs solver; Mach number 0.8, attack angle 1.25°.

model, the precision from the CNNs form mesh generates a mesh whose quantity of interest is more accurate than the GMG form solver.

The meshes around the different airfoils are shown in Figure 21. The leading airfoil takes a balance between the residual and dual solutions. The refined areas not only centered around the leading edge but also around the upper half. However, the dual solutions are not distributed around the remaining two airfoils. Then the refined areas are only centered around the shock waves.

## 5.3 Software

### 5.3.1 AFVM4CFD

AFVM4CFD is a library maintained by our group. It is an efficient solver which can solve the steady Euler equations with the *h*-adaptation method. Different modules such as k-exact reconstruction, parametric curves, and DWR-based *h*-adaptivity are integrated into this library. It is worth mentioning that with the AFVM4CFD, the Euler equations can be solved well with a satisfactory residual that gets close to the machine precision. In this work, we further improved the efficiency by introducing the CNNs module. Now the shape optimal design is under construction.

### 5.3.2 Training Model

The module build for the training can be seen from table 4. The details of the code can be found at https://github.com/ShanksFeng/CNNdualsolver.git. We will build more modules in the training in the future so that the performance will be better. The training can be conducted on different frameworks of the primal solver as long as the data structure is matched.

# 6 Conclusion

Based on the result of this study, the efficiency issues of the dual-consistent dual-weighted residual-based h-adaptive method proposed in [1] have been successfully addressed. It is demonstrated from the numerical results that the use of convolutional neural networks as a solver for dual equations, designing an automatic adjustment strategy for tolerance in the h-adaptive process, and introducing the OpenMP parallelization technique can significantly accelerate the whole framework. The feasibility of each approach was thoroughly discussed. Three salient features can be observed from these techniques as follows, i). the CNNs help us reduce time complexity to $O(n)$, ii). the algorithm executes automatically without manual intervention by the dynamic tolerance strategy. iii). the parallel computing technique significantly enhanced computational efficiency. Additionally, this study validated the importance of dual consistency in generating reliable training datasets. The trained model demonstrated generalization capabilities, allowing for precise simulation of configurations beyond the trained categories. It is worth noting

that the method can be conducted on the unstructured meshes, an example of multiple airfoils is illustrated. The enhanced efficiency of the method is crucial for the further study of shape optimization.

Nonetheless, as the geometric information is not effectively integrated within the neural networks, unexpected oscillations may still occur in the vicinity of the boundary. This highlights the potential for future investigations utilizing physics-informed neural networks, aiming for a more accurate prediction of the dual variables. Furthermore, there is room for optimizing the architecture of our neural networks, with the potential to achieve more efficient adaptation in subsequent studies.

## CRediT authorship contribution statement

**Jingfeng Wang**: Formal analysis, Methodology, Software, Writing - original draft. **Guanghui Hu**: Conceptualization, Methodology, Software, Supervision, Writing - review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgement

Thanks to the support from National Natural Science Foundation of China (Grant Nos. 11922120 and 11871489), FDCT of Macao S.A.R. (Grant No. 0082/2020/A2), MYRG of University of Macau (MYRG-GRG2023-00157-FST-UMDF) and Guangdong-Hong Kong-Macao Joint Laboratory for Data-Driven Fluid Mechanics and Engineering Applications (2020B1212030001).

#### References

[1] Jingfeng Wang and Guanghui Hu. Towards the efficient calculation of quantity of interest from steady Euler equations I: a dual-consistent DWR-based h-adaptive Newton-GMG solver. *arXiv preprint arXiv:2302.14262*, 2023.

[2] Antony Jameson. Aerodynamic shape optimization using the adjoint method. *Lectures at the Von Karman Institute, Brussels*, 2003.

[3] Guanghui Hu, Ruo Li, and Tao Tang. A robust high-order residual distribution type scheme for steady Euler equations on unstructured grids. *Journal of Computational Physics*, 229(5):1681–1697, 2010.

[4] Guanghui Hu, Ruo Li, and Tao Tang. A robust WENO type finite volume solver for steady Euler equations on unstructured grids. *Communications in Computational Physics*, 9(3):627–648, 2011.

[5] Guanghui Hu. An adaptive finite volume method for 2D steady Euler equations with WENO reconstruction. *Journal of Computational Physics*, 252:591–605, 2013.

[6] Xucheng Meng, Yaguang Gu, and Guanghui Hu. A fourth-order unstructured NURBS-enhanced finite volume WENO scheme for steady Euler equations in curved geometries. *Communications on Applied Mathematics and Computation*, pages 1–28, 2021.

[7] Guanghui Hu, Xucheng Meng, and Nianyu Yi. Adjoint-based an adaptive finite volume method for steady Euler equations with non-oscillatory k-exact reconstruction. *Computers & Fluids*, 139:174–183, 2016.

[8] Guanghui Hu and Nianyu Yi. An adaptive finite volume solver for steady Euler equations with non-oscillatory k-exact reconstruction. *Journal of Computational Physics*, 312:235–251, 2016.

[9] Ralf Hartmann. Adjoint consistency analysis of discontinuous Galerkin discretizations. *SIAM Journal on Numerical Analysis*, 45(6):2671–2696, 2007.

[10] Chengyu Liu and Guanghui Hu. An MP-DWR method for h-adaptive finite element methods. *Numerical Algorithms*, pages 1–21, 2023.

[11] Vít Dolejší and Filip Roskovec. Goal-oriented anisotropic hp-adaptive Discontinuous Galerkin Method for the Euler Equations. *Communications on Applied Mathematics and Computation*, 4:143–179, 2022.

[12] David Venditti and David Darmofal. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics*, 164(1):204–227, 2000.

[13] Guodong Chen and Krzysztof J Fidkowski. Output-based adaptive aerodynamic simulations using convolutional neural networks. *Computers & Fluids*, 223:104947, 2021.

[14] Guodong Chen and Krzysztof Fidkowski. Output-based error estimation and mesh adaptation using convolutional neural networks: Application to a scalar advection-diffusion problem. In *AIAA Scitech 2020 Forum*, page 1143, 2020.

[15] Ayan Chakraborty, Thomas Wick, Xiaoying Zhuang, and Timon Rabczuk. Multigoal-oriented dual-weighted-residual error estimation using deep neural networks. *arXiv e-prints*, pages arXiv–2112, 2021.

[16] Joseph Wallwork, Jingyi Lu, Mingrui Zhang, and Matthew Piggott. E2N: Error estimation networks for goal-oriented mesh adaptation. *arXiv preprint arXiv:2207.11233*, 2022.

[17] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *Osdi*, volume 16, pages 265–283. Savannah, GA, USA, 2016.

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[19] Marian Nemec and Michael Aftosmis. Toward automatic verification of goal-oriented flow simulations. Technical Report NASA/TM-2014-218386, 2014.

[20] Marian Nemec and Michael Aftosmis. Adjoint error estimation and adaptive refinement for embedded-boundary cartesian meshes. In *18th AIAA computational fluid dynamics conference*, page 4187, 2007.

[21] Rohit Chandra, Leo Dagum, David Kohr, Ramesh Menon, Dror Maydan, and Jeff McDonald. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE*

*international conference on computer vision*, pages 1026–1034, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] Ruo Li, Xin Wang, and Weibo Zhao. A multigrid block LU-SGS algorithm for Euler equations on unstructured grids. *Numerical Mathematics: Theory, Methods and Applications*, 1:92–112, 2008.

[25] Toru Yamahara, Kazuhiro Nakahashi, and Hyoungjin Kim. Adaptive mesh refinement using viscous adjoint method for single-and multi-element airfoil analysis. *International Journal of Aeronautical and Space Sciences*, 18(4):601–613, 2017.

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[27] Ludovic Trottier, Philippe Giguere, and Brahim Chaib-Draa. Parametric exponential linear unit for deep convolutional neural networks. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 207–214. IEEE, 2017.

[28] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[29] Vít Dolejší, Ondřej Bartoš, and Filip Roskovec. Goal-oriented mesh adaptation method for nonlinear problems including algebraic errors. *Computers & Mathematics with Applications*, 93:178–198, 2021.

[30] Wolfgang Bangerth and Rolf Rannacher. *Adaptive finite element methods for differential equations*. Springer Science & Business Media, 2003.

[31] Vít Dolejší and Georg May. *Goal-Oriented Anisotropic Mesh Adaptation*, pages 185–215. Springer International Publishing, Cham, 2022.

[32] Ralf Hartmann and Tobias Leicht. Generalized adjoint consistent treatment of wall boundary conditions for compressible flows. *Journal of Computational Physics*, 300:754–778, 2015.

[33] Vít Dolejší and Georg May. An anisotropic hp-mesh adaptation method for time-dependent problems based on interpolation error control. *Journal of Scientific Computing*, 95(2):36, 2023.

# 7 Appendix

| Layer Type | Layer Name | Operation | Output Shape | Activation |
|---|---|---|---|---|
| Input | input | - | batch_size× recon_size × 3×5 | - |
| Encoding | | | | |
| Conv2d | enc1 | convolution(F =64 × recon_size × 1× 3) | batch_size× 64× 3×5 | - |
| Inception | enc2 | convolutions with different kernel, Pooling | batch_size× 512 × 3×5 | - |
| Inception | enc3 | convolutions with different kernel, Pooling | batch_size× 1024 × 3×5 | - |
| ResBlock | enc4 | Normalization, Res Connect | batch_size× 512 × 3×5 | ELU |
| ResBlock | enc5 | Normalization, Res Connect | batch_size× 1024 × 3×5 | ELU |
| ResBlock | enc6 | Normalization, Res Connect | batch_size× 2048 × 3×5 | ELU |
| Decoding | | | | |
| ResBlock | dec1 | Normalization, Res Connect | batch_size× 1024 × 3×5 | ELU |
| Inception | dec2 | convolutions with different kernel, Pooling | batch_size× 2048 × 3×5 | - |
| ResBlock | dec3 | Normalization, Res Connect | batch_size×512 × 3×5 | ELU |
| Inception | dec4 | convolutions with different kernel, Pooling | batch_size× 1024× 3×5 | - |
| Linear | fc1 | Fully Connect | batch_size×64 | ELU |
| Linear | fc2_1 | Fully Connect | batch_size×1 | - |
| Linear | fc2_2 | Fully Connect | batch_size×1 | - |
| Linear | fc2_3 | Fully Connect | batch_size×1 | - |
| Linear | fc2_4 | Fully Connect | batch_size×1 | - |
| Output | output | - | batch_size×4 | - |

Table 4: Network architecture

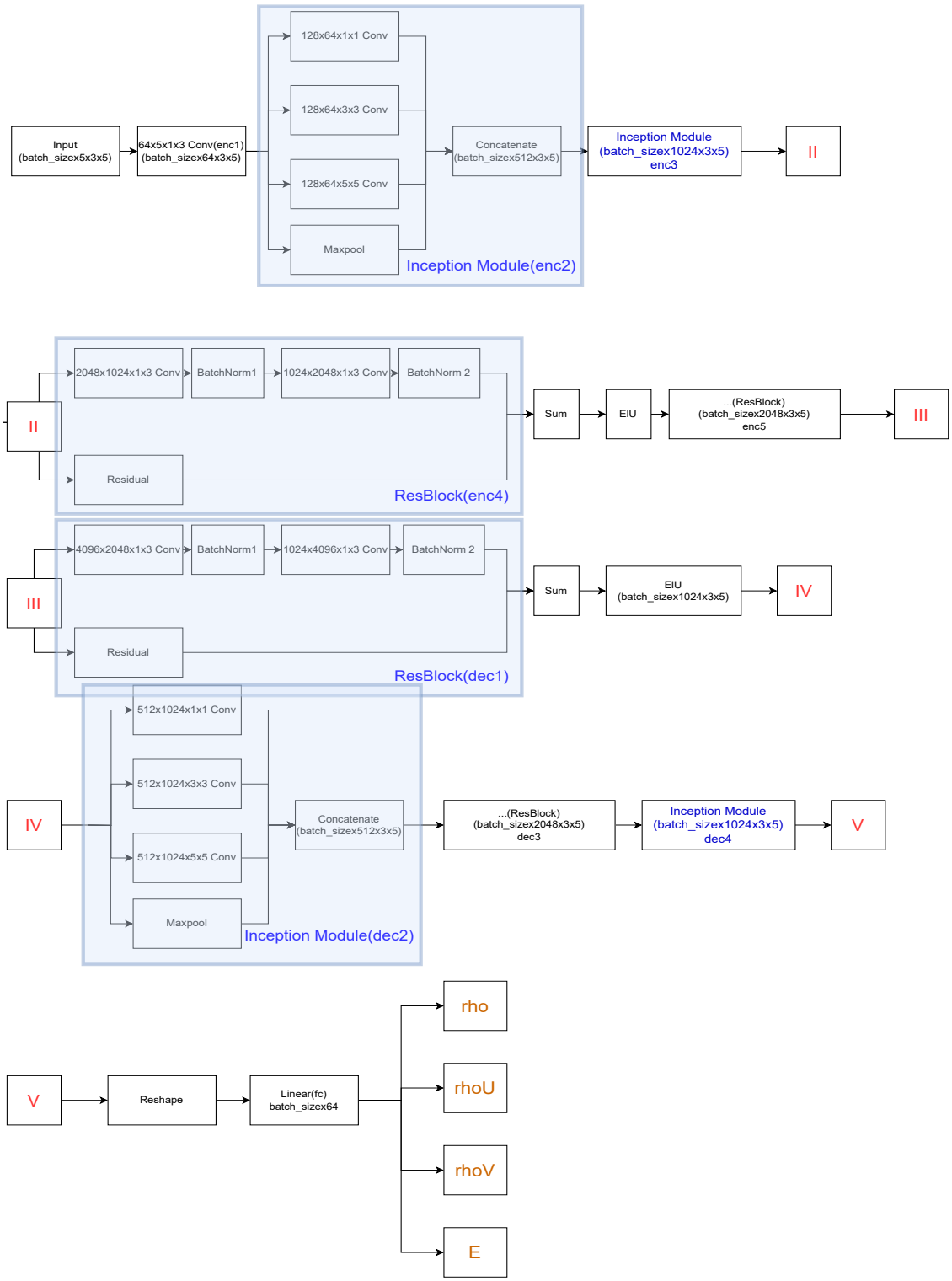



Figure 23: Framework of neural network.