Machine learning for adjoint vector in aerodynamic shape optimization

Mengfei Xu1 · Shufang Song1 · Xuxiang Sun1 · Wengang Chen1 · Weiwei Zhang1

Received: 28 May 2021 / Accepted: 22 June 2021 / Published online: 18 August 2021
© The Chinese Society of Theoretical and Applied Mechanics and Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract
Adjoint method is widely used in aerodynamic design because only once solution of flow field is required for it to obtain the gradients of all design variables. However, the computational cost of adjoint vector is approximately equal to that of flow computation. In order to accelerate the solution of adjoint vector and improve the efficiency of adjoint-based optimization, machine learning for adjoint vector modeling is presented. Deep neural network (DNN) is employed to construct the mapping between the adjoint vector and the local flow variables. DNN can efficiently predict adjoint vector and its generalization is examined by a transonic drag reduction of NACA0012 airfoil. The results indicate that with negligible computational cost of the adjoint vector, the proposed DNN-based adjoint method can achieve the same optimization results as the traditional adjoint method.

Keywords Machine learning · Deep neural network · Adjoint vector modelling · Aerodynamic shape optimization · Adjoint method

1 Introduction

Gradient-based optimization algorithm has become the most widely used engineering optimization algorithm because of its high efficiency. The computational cost of adjoint method is irrelevant to the number of design variables [1], and is merely proportional to the number of objective functions. In many design optimization problems, the number of objective functions is much lower than the number of design variable. Therefore, the adjoint-based optimization method is efficient and has been fully developed in the field of aerodynamic design. Pironneau [2] adopted the adjoint method in solving the aerodynamic shape optimization problem under Stokes flow. Jamson [3, 4] first proposed applying the continuous adjoint method to the aerodynamic shape optimization problem in the transonic velocity range. Later, Nielsen and Anderson [5] proposed the discrete adjoint method. Compared with the continuous adjoint equations that derived from the continuous governing equations, the discrete adjoint equations are derived from the discrete governing equations, and can be solved in the same approach as the governing equations. Since the continuous adjoint equations need to be discretized for numerically solving, the solution of discrete adjoint equations is of higher accuracy, which has been proved in Ref. [6]. Recently, Lyu et al. [7] employed the discrete adjoint method based on the Spalart–Allmaras turbulence model in the single-point and multipoint optimization of a National Aeronautics and Space Administration (NASA) common research model wing. Chen et al. [8, 9] studied the optimization problems of flutter suppression on the aircraft control surface and vortex shedding suppression in the flow past a blunt body, and improved the stability of fluid–structure interaction system by discrete adjoint method. He et al. [10] achieved aerothermal optimization for design variables of a U-shaped internal cooling passages based on discrete adjoint method. Kenway et al. [11] summarized the historical research developments and applications of adjoint method in computational fluid dynamics (CFD), studied and compared the various approaches to implement discrete adjoint method.

The current research is mainly focused on the applications of adjoint method. However, although the computational cost of adjoint vector is irrelevant to the number of design variables, it is still approximately equal to once flow-field
computation. Therefore, the computational cost cannot be ignored in the optimization process. To reduce the computational cost, Li et al. [12] constructed a reduced-order model (ROM) of the adjoint vector via proper orthogonal decomposition, and enabled the aerodynamic design optimization of RAE2822 and ONERA M6 wings by ROM. The computational time of the adjoint vector by ROM is only about 30\% of the traditional adjoint method (TAM). Chen et al. [13] considered the iterative solution of adjoint vector as a dynamic process, and reduced the iteration steps of adjoint vector by 60\%. The above methods can only save part of the solution time of the adjoint vector. If the adjoint vector can be predicted only by few samples, the efficiency of aerodynamic shape optimization will be remarkably improved, which is of great significance in the engineering field.

In recent years, artificial intelligence has achieved outstanding performance beyond human in the fields of natural language processing [14] and image recognition [15, 16]. Data-driven methods are widely used in the construction of equations and models. Machine learning methods, such as artificial neural network (ANN), have been widely applied and studied in the field of fluid mechanics [17]. The study of neural network appeared as early as 1989, Hornik et al. [18] stated that any function can be approximated by a sufficiently large and deep neural network. Recently, there is also a theory [19] indicated that sparsely connected DNN can be information theory-optimal nonlinear approximator for various functions and systems. There are many applications of ANN in the field of fluid mechanics, such as in machine learning methods for turbulence modeling, Zhu et al. [20] used a single-layer radial basis neural network (RBFNN) to constructed a turbulent algebraic model, and realized the coupling calculation with CFD solver. Ling et al. [21] constructed a deep neural network with tensor bases as input features to predict the Reynolds stress of typical flow at low Reynolds numbers (Re). Beck et al. [22] employed the residual convolution neural network to close the stress term in large eddy simulations and realized the coupling calculation with CFD solver. In addition, related applications in fluid mechanics include the identification of ordinary differential equations [23], partial differential equations [24, 25], etc. For aerodynamic shape optimization, some researchers constructed surrogate model based on ANN. Su et al. [26] integrated RBFNN and mesh adaptive direct search algorithm, and proved the less computational cost compared with traditional genetic algorithm (GA) via an example of airfoil optimization. Pehlivanoglu et al. [27] used RBFNN as a local surrogate model and combined it with GA in the multi-element airfoil optimization. Kharal et al. [28] employed the fully connected neural network to construct the mapping between the aerodynamic coefficients at different angles of attack and airfoil parameters at fixed Mach number and Re, and realized the inverse design of the airfoil. Sekar et al. [29] used convolutional neural network to construct a mapping between pressure distribution and airfoil, and realized the inverse design of the airfoil. However, this method requires up to 1200 airfoils and corresponding pressure distributions as training samples even at the fixed angle of attack and Re, which is difficult to meet in actual industry. Researchers also carried out DNN in aerodynamic shape optimization, such as Yan et al. [30, 31] built a DNN with the design variables of the rocket as the input and the corresponding changes as the output. However, there are strict limitations on design variables and shape parameterization in this modeling strategy.

Above machine learning methods for aerodynamic shape optimization always require massive amounts of samples and lack of generalization of incoming state. In addition, there is a lack of effective surrogate model to predict adjoint vector. Therefore, DNN is presented to construct the mapping between flow field information and adjoint vector. The optimization is accelerated by replacing the solution of adjoint equations with the DNN-based adjoint method (DAM).

The structure of this paper is as follows. In Sect. 2, the basic theory of CFD numerical methods and adjoint method are briefly introduced. In Sect. 3, the framework of DNN and the modeling workflow of DAM are introduced firstly, and the impact of different input features on the accuracy of modeling is studied in Sect. 3.3. In Sect. 4, the drag reduction results of DAM are compared with that of TAM in the test. Finally, Sect. 5 summarizes the conclusions.

## 2 Numerical methods

### 2.1 Governing equations

As a preliminary attempt of machine learning for adjoint vector modelling, Euler equation is used as the governing equation to explore the feasibility of proposed method.

In the two-dimension Cartesian coordinate system, when the volume force and heat source are ignored, the integral form of steady Euler equation is

$$\int_{\partial \Omega} F(U) \cdot n \, d\Gamma = 0,$$

where $U$ is the vector of conservative variables, $\Omega$ is the control volume, $F(U)$ is the vector of viscous fluxes, $\Gamma$ represents the boundary of the control volume $\Omega$, and $n$ donates the unit outward normal vector to the boundary.

$$U = \begin{bmatrix} \rho \\ \rho v \\ \rho E \end{bmatrix}, \quad F(U) = \begin{bmatrix} p v \\ p v v^T + \rho \\ (p + \rho E)v \end{bmatrix}.$$


where \( \rho \) is the density, \( \mathbf{v} \) is the velocity vector, \( E \) is the total energy, and \( p \) is the pressure. To close the Eq. (1), we introduce the ideal gas equation

\[
p = (\gamma - 1)p \left( E - \frac{|\mathbf{v}|^2}{2} \right),
\]

(3)

Euler equation is discretized by the finite volume method, then Eq. (1) on the control volume \( V_i \) within boundaries can be written as

\[
\sum_{j=1,2,\ldots,N_i} \Phi_{ij} = 0,
\]

(4)

where \( N_i \) represents the number of boundaries around the control volume \( V_i \). Let \( r_i = \sum_{j=1,2,\ldots,N_i} \Phi_{ij}, \mathbf{R} = [r_1, r_2, \ldots, r_N]^T \) refers to spatial discrete terms, we can get

\[
\mathbf{R} = \mathbf{0},
\]

(5)

where \( N \) is the total number of control volumes.

### 2.2 Adjoint method

After the calculation of flow field is completed, we can obtain the objective function \( J \), that is

\[
J = J(U, x_{\text{grid}}),
\]

(6)

where \( x_{\text{grid}} \) represents coordinates of grid points. When the design variables change, it will affect the value of \( J \) by affecting \( U \) and \( x_{\text{grid}} \). Therefore, the derivative of the objective function with respect to design variables can be obtained according to chain derivation rule as follows

\[
\frac{\partial J}{\partial D_j} = \frac{\partial J}{\partial U} \frac{\partial U}{\partial D_j} + \frac{\partial J}{\partial x_{\text{grid}}} \frac{\partial x_{\text{grid}}}{\partial D_j}, \quad j = 1, 2, \ldots, N_D.
\]

(7)

Equation (7) can be shortened to

\[
\frac{\partial J}{\partial D_j} = \frac{\partial J}{\partial D_j} + \frac{\partial J}{\partial U} \frac{\partial U}{\partial D_j}, \quad j = 1, 2, \ldots, N_D,
\]

(8)

where \( N_D \) is the number of design variables. \( \frac{\partial J}{\partial D_j} \) and \( \frac{\partial J}{\partial U} \frac{\partial U}{\partial D_j} \) on the right are easy to obtain, but it is difficult to calculate \( \frac{\partial U}{\partial D_j} \). Therefore, we introduce the adjoint vector \( \mathbf{A} \), which can be solved by

\[
\frac{\partial \mathbf{R}^T}{\partial U} \mathbf{A} = \frac{\partial J^T}{\partial U}.
\]

(9)

Transposing above formula and multiplying by \( \frac{\partial U}{\partial D_j} \), we can obtain

\[
\mathbf{A}^T \frac{\partial \mathbf{R}}{\partial U} \frac{\partial U}{\partial D_j} = \frac{\partial J}{\partial U} \frac{\partial U}{\partial D_j}.
\]

(10)

Considering and linearizing the governing equation \( \mathbf{R} = \mathbf{0} \), then we get the following equation

\[
\frac{\partial \mathbf{R}}{\partial U} \frac{\partial U}{\partial D_j} = -\frac{\partial \mathbf{R}}{\partial D_j}.
\]

(11)

Substituting Eq. (11) into Eq. (10), it can be written

\[
\frac{\partial J}{\partial D_j} = -\mathbf{A}^T \frac{\partial \mathbf{R}}{\partial D_j}, \quad j = 1, 2, \ldots, N_D.
\]

(12)

Substituting Eq. (12) into Eq. (8), we can get

\[
\frac{\partial J}{\partial D_j} = \frac{\partial J}{\partial D_j} - \mathbf{A}^T \frac{\partial \mathbf{R}}{\partial D_j}, \quad j = 1, 2, \ldots, N_D.
\]

(13)

Therefore, once the adjoint vector \( \mathbf{A} \) is obtained via Eq. (9), the gradients of all design variables can be obtained at negligible cost via Eq. (13). The original problem of solving \( N_D \) linear equations is transformed into solving only one linear equation. However, the solution of \( \mathbf{A} \) is still time consuming. Considering the definition of \( \mathbf{A} \) in Eq. (9), \( \frac{\partial \mathbf{R}^T}{\partial U} \) is a large sparse matrix of size \( N_e \times N_e \), and \( \frac{\partial J^T}{\partial U} \) is a vector of size \( N_e \times 1 \). \( N_e \) is the total number of grid points, which can be more than \( 10^4 \) in two-dimension case. TAM requires multi-step iteration to solve such a large sparse linear system. However, through DNN, the adjoint vector will be predicted quickly based on the flow field information. Then the gradients can be calculated via Eq. (13), so as to efficiently optimize aerodynamic shape. The accuracy verification of flow field solution and adjoint method program can be referred to Ref. [13].

### 3 Machine learning for adjoint vector

#### 3.1 Deep neural network

As one of the most well-known methods in machine learning, DNN possesses a strong capability to approximate nonlinear system, which benefits from its modular structure as shown in Fig. 1. DNN is composed of multiple neural layers, each with many neurons, and data is transmitted between layers through connection lines with weights. The number of neurons in the input layer depends on the number of features of the flow field on the grid points, while the number of neurons in the output layer corresponds to the number of adjoint vector’s components.

The input layer receives the data of flow-field features, and the sum of weighted data and bias is transferred to the next layer. Activation function is performed on each hidden layer. Take the first hidden layer as an example, the calculation from the inputs to the first hidden layer neuron can be expressed as

\[
\mathbf{A} = \sum_{j=1}^{N_D} \mathbf{A}^T \left( \frac{\partial \mathbf{R}}{\partial U} \right) \frac{\partial U}{\partial D_j}.
\]
where \( X \) indicates the input data. \( \omega_{j,i} \) represents the weight connecting the input layer neuron \((i = 1, 2, \ldots, N)\) and the first hidden layer neuron \((j = 1, 2, \ldots, S)\). \( N \) and \( S \) are the number of the neurons in the input layer and hidden layer, respectively. \( b \) refers to the bias vector. \( \sigma \) indicates the activation function in the hidden layer. Common choices of activation function are nonlinear functions such as ReLU, Sigmoid, and Tanh.

Finally, the data is transferred to the output layer and produces a prediction of adjoint vector. Once the loss function is determined, the partial derivatives of loss function with respect to the learnable parameters \( \omega \) and \( b \) can be obtained according to chain derivation rule. In order to achieve the minimum value of the loss function, the parameters will be updated via the error backpropagation algorithm and optimization algorithm (as shown in Ref. [32]) based on the partial derivatives. In this paper, the pytorch library [33] is used to build and train the DNN, and the adopted loss function is mean squared error loss (MSELoss) function, which is denoted by \( \Sigma^2 \text{Loss} \). Through the nonlinear transformation in multiple hidden layers, the data of flow-field features is mapped to a space on which the distribution is consistent with the adjoint vector.

In general, the deeper the hidden layer is, the stronger the power of DNN is. However, as the hidden layers deepen, the training becomes complicated due to the phenomenon of gradient disappearance and gradient explosion. This phenomenon can be avoid by using batch normalization method [34], which normalizes layer inputs, accelerates training effectively, and improves the accuracy. Also, in order to facilitate training, the inputs and outputs of DNN are linearly scaled in interval \([0,1]\).

The overall hyperparameters of the constructed DNN is briefly described as follows. The DNN is composed of 5 hidden layers, of which the number of layer neurons is 100, 100, 80, 50, and 30, respectively. Batch normalization method is adopted for all hidden layers. In the output layer, there are 4 neurons corresponding to the four components of the adjoint vector. The training process is divided into two stages. In the first stage, the batch size, the initial learning rate, and the number of training epochs are set to \(2^{15}, 0.03, \) and \(1000\), and the second stage to \(2^{12}, 0.003, \) and \(600\), respectively. During the training process, when no decrease of minimum value of validation loss is seen for 40 training epochs, the learning rate is halved. The training is carried on graphic processing unit (GPU) of GTX1650.

### 3.2 Modeling workflow

The specific modeling workflow is shown as Fig. 2. The class-shape-transform (CST) methodology [35] is used to parameterize the geometric shape of airfoil. The upper surface and lower surface of the airfoil are both parameterized by 6 CST parameters. Samples are calculated by CFD solver at different boundary conditions of CST parameters, Mach number (\(Ma\)) and angle of attack (\(\alpha\)). The adjoint vector \(A\) can be obtained by adjoint solver base on the flow-field solution \((\rho, u, v, p)\). The total number of mesh elements is 13, 490. Thus, at each boundary condition, 13, 490 couples of the flow-field features are used as input of DNN.
as inputs and the corresponding $\mathbf{A}$ as outputs are used for training or validation. The proposed DNN is trained to locally map from $(\rho, u, v, p, ...)$ to $\mathbf{A}$. Therefore, the iterative solution of adjoint vector by TAM can be replaced by predicting $\hat{\mathbf{A}}$ via DNN based on the data of $(\rho, u, v, p, ...)$. Finally, the estimated gradients can be obtained.

There are two main advantages of this approach. First of all, the training benefits from the massive amounts of data in CFD. Naturally, each flow-field solution is composed of tens of thousands of values on mesh cell. Secondly, the local mapping possesses a certain generalization and is irrelevant to the computational mesh.

### 3.3 Input features design

According to the definition of $\mathbf{A}$ in Eq. (9), when the objective function $J$ is determined, $\partial R^T/\partial U$ and $\partial f^T/\partial U$ can be written as a function of the whole-field $(\rho, u, v, p)$. In consequence, $\mathbf{A}$ can also be uniquely determined from the whole-field. However, the constructed DNN maps from the local $(\rho, u, v, p, ...)$ to the local $\mathbf{A}$, which is incorrect in physics. In order to address this problem, the spatial gradients of $(\rho, u, v, p)$ are added to the inputs of DNN, as shown in Table 1. The incompleteness of mapping can be compensated to a certain extent by this strategy. In addition, the accuracy of DNN is also improved.

In order to verify the modeling capability of features, training data of the NACA0012 is obtained at $Ma = 0.8$, $\alpha = 3^\circ$ with drag coefficient $C_d$ as the objective function. DNN is trained with the basic flow features as inputs and features including spatial gradients as inputs respectively. The absolute mean errors are 0.3158 and 0.1775, respectively. In Fig. 3, the contour images of the first component of $\mathbf{A}$ is compared with the ground truth. By adding spatial gradients to the inputs, the accuracy of the adjoint vector modeling is significantly improved. Therefore, the following work adopt these 12 features including spatial gradients as inputs of DNN.

Furthermore, DNN of the above architecture is used to learn the mapping between the data of flow-field features including spatial gradients and $\mathbf{A}$ in two cases. As shown in Table 2, data of training is obtained at subsonic incoming state with lift coefficient $C_l$ as the objective function in the first case, and at transonic incoming state with $C_d$ as the objective function in the second case. Then $\mathbf{A}$ and the estimated gradients produced by the DNN are compared with the ground truth.

| Table 1 Design of flow-field features |
|---------------------------------------|
| Basic flow features $(\rho, u, v, p)$ |
| Features including spatial gradients $(\rho, u, v, p, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial p}{\partial x}, \frac{\partial p}{\partial y})$ |

| Table 2 Information of two cases |
|----------------------------------|
| Airfoil | $Ma$ | $\alpha$ | Objective function |
|--------|-----|------|------------------|
| Case 1 | NACA0012 | 0.5  | 3$^\circ$ | $C_l$ |
| Case 2 | NACA0012 | 0.75 | 2$^\circ$ | $C_d$ |

![Fig. 3 Comparison of first component of $\mathbf{A}$. a Ground truth. b By basic flow features. c By features including spatial gradients](image-url)
Fig. 4 Comparisons of the traditional adjoint method and DNN-based adjoint method in case 1. a First component of $A$. b Second component of $A$. c Third component of $A$. d Fourth component of $A$. e Gradients of 12 CST parameters.
Fig. 5 Comparisons of the traditional adjoint method and DNN-based adjoint method in case 2. 

(a) First component of $A$. (b) Second component of $A$. (c) Third component of $A$. (d) Fourth component of $A$. (e) Gradients of 12 CST parameters.
The contour image comparisons of $\mathbf{A}$ and CST gradients in case1 and case2 are illustrated in Figs. 4 and 5, respectively. The first four contour images correspond to the four components of $\mathbf{A}$. From the comparisons, it is incredibly accurate for adjoint vector modeling via DNN in training cases, regardless of subsonic or transonic incoming states and $C_l$ or $C_d$ as objective functions. From Fig. 4a–d and Fig. 5a–d, the contour images of $\hat{\mathbf{A}}$ have almost the same distribution as those of $\mathbf{A}$, which is calculated by TAM. The gradients of CST obtained from $\mathbf{A}$ and $\hat{\mathbf{A}}$ match well as shown in Figs. 4e and 5e.

4 Drag reduction about NACA0012 airfoil by DNN-based adjoint method

4.1 Analysis of required sample number at fixed incoming state

The powerful fitting capability of the DNN has been proved in the previous section. Next, we study on how the number of samples for training and validation affects the generalization of the DNN. By Latin-hypercube sampling [36] within the variation $\pm 30\%$ of CST parameters of NACA0012 airfoil, 30 sample airfoils are randomly selected. All sample airfoils are different from NACA0012 airfoil, as shown in Fig. 6.

The data of flow-field features and $\mathbf{A}$ of all sample airfoils are obtained by CFD solver at $Ma = 0.78$, $\alpha = 2.5^\circ$, and then added successively to the data set for training and validation. The sizes of total input data and output data are $(13490 \times n, 12)$ and $(13490 \times n, 4)$, respectively, where $n$ indicates the number of sample airfoils for training and validation. 20% of the data is randomly selected as the validation set, and the remaining 80% as the training set. During the training process, the validation loss, which is calculated on the validation set, will not be backpropagated to identify the network weights and biases, but only as reference to prevent overfitting. Due to the randomness in training-validation-split process, batch selection during training and the initialization of model parameters, the training process is repeated 5 times. The minimum test loss is regarded as the performance of DNN. Figure 7 illustrates the function between the minimum test loss and the number of sample airfoils. This curve can be well fitted by the power function, that is

$$\Sigma_{\text{loss}}^2 = 3.584 \times 10^{-4} \times n^{-1.336} + 8.884 \times 10^{-5}.$$ 

It can be seen from Fig. 7 that when the number of sample airfoils increases from 1 to 5, the minimum test loss decreases significantly. After that, as the number of sample airfoils increases, the minimum test error decreases more slowly. Therefore, at fixed incoming state, it is a reasonable choice considering accuracy and cost to randomly sample 10 airfoils as training and validation sets.

4.2 Analysis of prediction accuracy

In the previous section, we explored the relationship between the generalization and the number of sample airfoils at the fixed incoming state. In this section, the generalization of the DNN is studied at the different incoming state. By Latin-hypercube sampling within the range of $Ma = 0.75 - 0.78$, $\alpha = 2^\circ - 3^\circ$, and $\pm 30\%$ variation CST parameters of NACA0012 airfoil, 30 sample combinations are randomly selected. The data of flow-field features and $\Lambda$ at different combinations are obtained as training and validation sets.
The training-validation distributions of incoming states and airfoils are shown in Fig. 8. Data of 5 sample combinations is randomly selected as the validation set, and the remaining data of 25 sample combinations as the training set. Similarly, the function between the minimum validation loss in 5 times and the number of sample airfoils for training is studied and shown in Fig. 9. The curve is fitted by the power function:

\[ \Sigma_{\text{Loss}}^2 = 1.378 \times 10^{-4} \times n^{-1.061} + 3.679 \times 10^{-5}, \]

where \( n \) indicates the number of sample airfoils for training. The MSELoss continued to decrease with more samples used for training, from \( \Sigma_{\text{Loss}}^2 = 9.608 \times 10^{-5} \) at \( n = 16 \) to \( \Sigma_{\text{Loss}}^2 = 8.490 \times 10^{-5} \) at \( n = 25 \), while the training costs are 180.6 s and 233.7 s, respectively. Since the training cost is acceptable if compared with the adjoint solving, it is recommended to use more samples in order to achieve a better performance. Therefore, all 25 samples are taken as training set in the following work.

Figure 10 shows the convergence of training loss and validation loss during training. The oscillation of validation loss appears around \( 4 \times 10^{-3} \) firstly and eventually disappear due to the learning rate reducing. The batch size and learning rate are reset at the 1000th epochs of the training process as illustrated above. After a brief oscillation and increase, the training and validation losses eventually converge to a smaller value. By adopting this training strategy, the network weights and biases can be quickly identified. For the explanation, we believe it can speed up training by setting a larger batch size at the beginning. When the networks weights and biases change stably, reducing the batch size appropriately can make the DNN focus more on the learning of the local extremum of \( A \). Therefore, the training and validation losses can be further reduced. During the training process, the DNN achieves the smallest value of validation loss \( (8.490 \times 10^{-5}) \) at 1247th step, while the training loss is
Fig. 11  Comparisons of the traditional adjoint method and DNN-based adjoint method on NACA0012 at $Ma = 0.78$, $\alpha = 2.5^\circ$. a First component of $A$. b Second component of $A$. c Third component of $A$. d Fourth component of $A$. e Gradients of 12 CST parameters
4.237 \times 10^{-5}. The contour images of validation results are shown in the appendix.

$\mathbf{\hat{A}}$ and gradients of NACA0012 airfoil at $Ma = 0.78$, $\alpha = 2.5^\circ$ are predicted by this trained DNN. The prediction only takes 0.015 s by DNN on GPU of GTX1650 while numerical solving $\Lambda$ needs about 70.4 s on CPU of I7. Since the combination of NACA0012 airfoil and this incoming state is different from the sample combinations, it is a test case for the DNN. The contour images of $\mathbf{\hat{A}}$ are compared with its ground truth as shown in Fig. 10a–d. It can be observed that DNN can accurately predict $\mathbf{\hat{A}}$, even for the extreme values in the tail region. Therefore, the gradients calculated based on the precisely predicted $\mathbf{\hat{A}}$ match well with the TAM results as shown in Fig. 10e. The error of gradient can be accepted in engineering.

### 4.3 Aerodynamic shape optimization by DNN-based adjoint method

Through the verification of the gradients on NACA0012 airfoil, the initial direction of optimization is ensured. Next, the DAM is used to guide the drag reduction about NACA0012 airfoil at $Ma = 0.78$, $\alpha = 2.5^\circ$ in which a certain number of constrains must be satisfied. The lift constrain and area constrain are considered as an equality. The change of CST parameter is limited to be within 30%. The initial
lift coefficient $C_l$ is 0.5845, the initial drag coefficient $C_d$ is 0.0379, and the initial dimensionless area $A_0$ is 0.0821. The mathematical model of the optimization is expressed as follows:

$$\begin{align*}
\min_{D} & \quad C_d(D), \\
\text{s.t.} & \quad C_l(D) \geq C_{l0}, \\
& \quad A(D) \geq A_0, \\
& \quad |\Delta CST| \leq 0.3CST_0,
\end{align*}$$

where $D$ represents design variable vector, which is CST parameter vector in this paper. We adopted penalty function method to constrain the optimization problem. The initial penalty factor and the factor expansion are set to 1 and 10, respectively. Details of setting can be found in Ref. [13].

In Fig. 11, the optimization results of DAM with TAM are compared. As shown in Fig. 11a and b, optimal airfoil and pressure distribution of DAM are almost the same as those of TAM. There are very slight differences in the upper surface of airfoil, relatively larger differences in the lower surface, and main differences of pressure coefficient appear in the upper surface of airfoil. Therefore, the gradients of upper surface of airfoil are much more important and DNN should focus more on the upper surface during training process. The predicted gradients of the optimal airfoil by DAM basically match the ground truth (Fig. 11c). Therefore, it is foreseeable that if the optimization constraints are properly released, DAM can still obtain the same optimized shape as TAM. Figure 11d illustrates the $C_d$ iteration of DAM and TAM, which takes 22 steps and 18 steps to converge, respectively. Airfoils with a lower $C_d$ but outside the constraint boundary are first obtained by both methods. Then the airfoil shape is gradually optimized to the inside of the constraint boundary as the penalty factor increases. Finally, the optimal airfoils that satisfies the constrains but with a relatively larger $C_d$ are obtained by both methods (Fig. 12).

In Table 3, the aerodynamic coefficients of the optimal airfoil by DAM and TAM are compared. The $C_d$ is reduced by 55.6% via TAM and 55.4% via DAM. Therefore, the slight difference in the lower surface of optimal airfoil has little impact on $C_d$, which proves the high accuracy of the DAM. The computational cost of TAM mainly includes three parts: calculation for the flow field, calculation for the adjoint vector with objective function $C_l$, and calculation for the adjoint vector with objective function $C_d$. However, as a preliminary attempt, for now DAM only replace the last calculation by DNN. The efficiency of DAM can be further improved if calculation for the adjoint vector with objective function $C_l$ is also replaced. As shown in Table 4, the computational cost per step of DAM is about 65.7% that of TAM. Because TAM takes more steps in optimization, the total computational cost of DAM is about 80% that of TAM.

5 Conclusion

DNN is constructed to learn a nonlinear mapping between the local flow field information and the adjoint vector. By adding the spatial gradients of the local flow field to the input features of DNN, the prediction accuracy is significantly improved. The function between the minimum test loss and the number of samples for training and validation is studied and well fitted by a power function. By training on data of different combinations of transonic incoming state and airfoils, DNN possesses the generalization of $Ma$, $\alpha$, and shape. Through the drag reduction about NACA0012 airfoil at the test incoming state, it is verified that the optimal airfoil and the pressure distribution by DAM match well with those by TAM. The results indicate that by learning a small number of samples in a certain range, the DNN can precisely predict the adjoint vector and achieve efficient aerodynamic shape optimization. The following work will be extended to the adjoint vector modelling based on unsteady Navier–Stokes equation.

Appendix

See Figs. 13, 14, 15, and 16.
Fig. 13 Contour images of first component of $\mathbf{A}$ on validation sets. a $Ma = 0.784$, $\alpha = 2.048^\circ$. b $Ma = 0.758$, $\alpha = 2.257^\circ$. c $Ma = 0.755$, $\alpha = 2.895^\circ$. d $Ma = 0.782$, $\alpha = 2.276^\circ$. e $Ma = 0.764$, $\alpha = 2.850^\circ$
Fig. 14 Contour images of second component of $A$ on validation sets. 

- **a** $Ma = 0.784$, $\alpha = 2.048^\circ$.  
- **b** $Ma = 0.758$, $\alpha = 2.257^\circ$.  
- **c** $Ma = 0.755$, $\alpha = 2.895^\circ$.  
- **d** $Ma = 0.782$, $\alpha = 2.276^\circ$.  
- **e** $Ma = 0.764$, $\alpha = 2.850^\circ$
Fig. 15 Contour images of third component of $A$ on validation sets. a $Ma = 0.784$, $\alpha = 2.048^\circ$. b $Ma = 0.758$, $\alpha = 2.257^\circ$. c $Ma = 0.755$, $\alpha = 2.895^\circ$. d $Ma = 0.782$, $\alpha = 2.276^\circ$. e $Ma = 0.764$, $\alpha = 2.850^\circ$
Fig. 16 Contour images of fourth component of $A$ on validation sets. a $Ma = 0.784$, $\alpha = 2.048^\circ$. b $Ma = 0.758$, $\alpha = 2.257^\circ$. c $Ma = 0.755$, $\alpha = 2.895^\circ$. d $Ma = 0.782$, $\alpha = 2.276^\circ$. e $Ma = 0.764$, $\alpha = 2.850^\circ$. 
Acknowledgements This work was supported by the National Numerical Wind tunnel Project (Grant NNW2018-ZT1B01) and the National Natural Science Foundation of China (Grant 91852115).

References

1. Giles, M.B., Pierce, N.A.: An Introduction to the adjoint approach to design. Flow Turbulence Combustion 65, 393–415 (2000). https://doi.org/10.1023/A:1011430410075
2. Pironneau, O.: On optimum profiles in Stokes flow. J. Fluid Mech. 59, 117–128 (1973). https://doi.org/10.1017/S002211207300145X
3. Jameson, A.: Aerodynamic design via control theory. J. Sci. Comput. 3, 233–260 (1988). https://doi.org/10.1007/BF01061285
4. Jameson, A.: Optimum aerodynamic design using CFD and control theory. In: 12th Computational Fluid Dynamics Conference (1995). https://doi.org/10.2514/6.1995-1729
5. Nilsen, E.J., Anderson, W.K.: Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations. AIAA J. 37, 1411–1419 (1999). https://doi.org/10.2514/2.640
6. Nadaraja, S., Jameson, A.: A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. In: 38th AIAA Aerospace Sciences Meeting and Exhibit (2000)
7. Lyu, Z., Kenway, G.K.W., Martins, J.R.R.A.: Aerodynamic shape optimization investigations of the common research model wing benchmark. AIAA J. 53, 968–985 (2015). https://doi.org/10.2514/1.J053318
8. Chen, W., Gao, C., Gong, Y., et al.: Shape optimization to improve the transonic fluid-structure interaction stability by an aerodynamic unsteady adjoint method. Aerospace Sci. Technol. (2020). https://doi.org/10.1016/j.ast.2020.105871
9. Chen, W., Li, X., Zhang, W.: Shape optimization to suppress the lift oscillation of flow past a stationary circular cylinder. Phys. Fluids (2019). https://doi.org/10.1063/1.5095841
10. He, P., Mader, C.A., Martins, J., et al.: Aerothermal optimization of internal cooling passages using a discrete adjoint method. In: 2018 Joint Thermophysics and Heat Transfer Conference (2018)
11. Kenway, G.K.W., Mader, C.A., He, P., et al.: Effective adjoint approaches for computational fluid dynamics. Prog. Aerospace Sci. (2019). https://doi.org/10.1016/j.paerosci.2019.05.002
12. Li, J., Cai, J., Qu, K.: Adjoint-based two-step optimization method using proper orthogonal decomposition and domain decomposition. AIAA J. 56, 1133–1145 (2018). https://doi.org/10.2514/1.J055773
13. Chen, W., Zhang, W., Liu, Y., et al.: Accelerating the convergence of steady adjoint equations by dynamic mode decomposition. Struct. Multidiscip. Optim. 62, 747–756 (2020). https://doi.org/10.1007/s00158-020-2531-5
14. Devlin, J., Chang, M.-W., Lee, K., et al.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805. (2018)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 84–90 (2017). https://doi.org/10.1145/3065386
16. Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.: YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:200410934. (2020)
17. Brunton, S.L., Noack, B.R., Koumoutsakos, P.: Machine learning for fluid mechanics. Annu. Rev. Fluid Mech. 52, 477–508 (2020). https://doi.org/10.1146/annurev-fluid-011719-060214
18. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. 2, 359–366 (1989). https://doi.org/10.1016/0893-6080(89)90020-8
19. Bölcskei, H., Grohs, P., Kutyniok, G., et al.: Optimal approximation with Sparsely connected deep neural networks. SIAM J. Math. Data Sci. 1, 8–45 (2019). https://doi.org/10.1137/18m118709x
20. Zhu, L., Zhang, W., Kou, J., et al.: Machine learning methods for turbulence modeling in subsonic flows around airfoils. Phys. Fluids 31, 015105 (2019). https://doi.org/10.1063/1.5061693
21. Ling, J., Kurzawski, A., Templeton, J.: Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J. Fluid Mech. 807, 155–166 (2016). https://doi.org/10.1017/jfm.2016.615
22. Beck, A., Flad, D., Munz, C.-D.: Deep neural networks for data-driven LES closure models. J. Comput. Phys. (2019). https://doi.org/10.1016/j.jcp.2019.108910
23. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., et al.: Neural ordinary differential equations. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems (2018). https://doi.org/10.5555/3327757.3327764
24. Rudy, S.H., Brunton, S.L., Proctor, J.L., et al.: Data-driven discovery of partial differential equations. Sci. Adv. 3, e1602614 (2017). https://doi.org/10.1126/sciadv.1602614
25. Raissi, M., Karniadakis, G.E.: Hidden physics models: machine learning of nonlinear partial differential equations. J. Comput. Phys. 357, 125–141 (2018). https://doi.org/10.1016/j.jcp.2017.11.039
26. Su, W., Gao, Z., Zuo, Y.: Application of RBF Neural Network Ensemble to Aerodynamic Optimization. In: 46th AIAA Aerospace Sciences Meeting and Exhibit (2008) DOI: https://doi.org/10.2514/6.2008-154
27. Pehlivanoglu, Y.V., Yagiz, B.: Aerodynamic design prediction using surrogate-based modeling in genetic algorithm architecture. Aerosp. Sci. Technol. 23, 479–491 (2012). https://doi.org/10.1016/j.ast.2011.10.006
28. Kharal, A., Saleem, A.: Neural networks based airfoil generation for a given C-P using Bezier-PARSEC parameterization. Aerosp. Sci. Technol. 23, 330–344 (2012). https://doi.org/10.1016/j.ast.2011.08.010
29. Sekar, V., Zhang, M.Q., Shu, C., et al.: Inverse design of airfoil using a deep convolutional neural network. AIAA J. 57, 993–1003 (2019). https://doi.org/10.2514/1.J057894
30. Yan, X.H., Zhu, J.H., Kuang, M.C., et al.: Missile aerodynamic design using reinforcement learning and transfer learning. Sci. China Inf. Sci. 61, 3 (2018). https://doi.org/10.1007/s11432-018-9463-x
31. Yan, X.H., Zhu, J.H., Kuang, M., et al.: Aerodynamic shape optimization using a novel optimizer based on machine learning techniques. Aerosp. Sci. Technol. 86, 826–835 (2019). https://doi.org/10.1016/j.ast.2019.02.003
32. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint: arXiv:14126980 (2014)
33. Paszke, A., Gross, S., Massa, F., et al.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv preprint arXiv:191201703 (2019)
34. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint: arXiv:150203167 (2015)
35. Kulfan, B.: A Universal Parametric Geometry Representation Method - “CST”. In: 45th AIAA Aerospace Sciences Meeting and Exhibit (2007) DOI: https://doi.org/10.2514/6.2007-62
36. Helton, J.C., Davis, F.J.: Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Reliab. Eng. Syst. Saf. 81, 23–69 (2003). https://doi.org/10.1016/s0951-8320(03)00058-9

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.