RESEARCH Open Access

Deep learning prediction method for aerodynamic forces on morphing aircraft considering physical monotonicity

Jiachi Zhao[1], Lifang Zeng[1*], Aoxue Lin[1] and Xueming Shao[1]

[1] School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China

Abstract

Physical monotonicity is a pervasive phenomenon in the aerodynamic characteristics of aircraft, where the aerodynamic lift consistently increases with the angle of attack within the stalling range. Existing machine learning models for aerodynamic predictions often overlook this monotonicity, resulting in poor interpretability and credibility. To address this issue, we introduce a monotonic model, the Deep Lattice Network, which integrates the monotonicity constraint of the lift coefficient into machine learning based aerodynamic prediction framework. In this paper, we propose a novel deep learning model, Deep Lattice Cross Network, which aims to rapidly predict aerodynamic forces with high precision while ensuring monotonic constraints. Multi-Task Learning method is utilized to simultaneously predict both lift and drag coefficients, thereby enhancing the efficiency of the model. To optimize the training process and minimize costs, we adopt a unique two-phase deep network training strategy. Based on computational fluid dynamics simulation datasets of a morphing aircraft, the model is trained, and the efficacy of the model is tested by two interpolation and two extrapolation datasets. The results show a remarkable alignment with computational fluid dynamics outcomes across all test scenarios. Extended testing across a wider range of attack angles further highlights the superiority of the Deep Lattice Cross Network in upholding monotonicity. Incorporating monotonicity constraints not only improves predictive accuracy of the model but also greatly enhances its physical interpretability, which is crucial for advancing the development of more dependable aerodynamic prediction models.

Keywords: Monotonicity constraints, Morphing aircraft, Unsteady aerodynamic force, Deep learning

1 Introduction

Deep learning's capacity to predict aerodynamic parameters with high precision and swiftness is transforming the field, offering insights in real-time that were previously unattainable [1, 2]. Advances in physics-informed deep learning have further enhanced the interpretability and adherence of these predictions to fundamental physical laws [3].

The 'physics-informed neural networks' (PINNs) family is a well-known physics-informed deep learning algorithm [3, 4]. PINN is a new class of universal function

approximators proposed by Raissi et al. [4]. It has the ability to encode any underlying physical laws that govern a given data-set and can be described by partial differential equations. As soon as it was proposed, PINN drew a lot of interest. Many works based on it have been conducted with positive results in the field of fluid mechanics. Yang et al. [5] proposed a Bayesian PINN to solve forward and inverse Partial Differential Equations (PDE) problems and showed that Bayesian PINNs have the capability of avoiding overfitting and obtaining more accurate predictions in scenarios even with large noise. Raissi et al. [6] developed a physics-informed deep learning framework, hidden fluid mechanics, to extract velocity and pressure fields directly from images. Hidden fluid mechanics can encode the Navier–Stokes (NS) equations into neural networks while being agnostic to the geometry or the initial and boundary conditions, which is robust to low resolution and substantial noise in the observation data. Cai et al. [7] proposed a novel machine learning algorithm based on PINNs to extract information on the 3D velocity and pressure fields above an espresso cup from temperature data of Tomo-BOS experiments. Sun et al. [8] devised a structured deep neural network (DNN) architecture to approximate the solutions of the parametric NS equations, in which the boundary conditions are satisfied automatically. They tested the proposed methods on three flow cases relevant to cardiovascular applications, and the results showed great agreement with Computational Fluid Dynamics (CFD) simulations.

Beyond PINNs, a variety of physics-informed machine learning models have emerged, embedding physical constraints within DNNs to enhance their predictive power. Ling et al. [9] used a multiplicative layer with an invariant tensor basis to embed Galilean invariance into the network architecture, and the prediction accuracy in turbulence modelling was significantly improved in test cases. Sheng and Yang [10] presented a penalty-free neural network method to solve a class of second-order boundary-value problems on complex geometries. Numerical experiments demonstrated that their method was more accurate, flexible and robust than previous state-of-the-art methods. Darbon and Meng [11] proposed new connections between neural network architectures and viscosity solutions to certain Hamilton–Jacobi PDEs. Lu et al. [12] used multi-fidelity NNs to extract material properties from instrumented indentation data.

Karniadakis et al. emphasized that embedding physical constraints into DNNs is crucial for physics-informed machine learning [3]. Monotonicity, a typical prior knowledge and physical constraint, has been taken into account in many real-world machine learning applications such as loan approval, house pricing prediction, bankruptcy prediction, etc. [13], as well as physics-informed machine learning [1]. Many researchers have proposed DNN models with monotonic constraints. Among these monotonic neural networks, restricting network weights to non-negative values may be the simplest way for creating monotonic models [14, 15]. However, this simple monotonic constraint method may limit DNN performance and predictability. In order to increase the performance while maintaining the monotonicity constraint, many complex methods have been presented. Based on monotonic linear embedding and min-max-pooling, Sill proposed a three-layer network with monotone functions [16]. Daniels and Velikova extended Sill's technique to partially monotonic functions by combining min-max-pooling and partial monotonic linear embedding [15]. In addition to min-max-pooling, some models based on lattice, a kind of look-up table, have been proposed to ensure monotonicity.

Gupta et al. proposed monotonic lattice regression, which was achieved by adding linear inequality constraints to make adjacent look-up table parameters to be monotonic [17]. Canini et al. proposed multidimensional flexible monotonic networks based on ensembles of lattices [18]. Combining linear embeddings, calibrators and ensembles of lattices, Deep Lattice Networks (DLNs) were proposed and outperformed previous monotonic methods [19].

In the field of aircraft aerodynamics, monotonicity is evident. For instance, the lift coefficient (C_l) typically increases with the angle of attack within a certain range. In the current study, we introduce the monotonicity constraint into the deep network prediction model of a morphing aircraft, aiming to achieve more reliable and interpretable predictions of C_l. A novel machine learning model DLN [19] is utilized to guarantee the monotonicity of C_l with respect to angle of attack (α). Besides, Cross Layer [20] is used to predict drag coefficient (C_d) to make use of the limited training data effectively. Based on Lattice method and Cross Layer, a novel data-driven model, Deep Lattice Cross (DLC) Network, is proposed to predict the unsteady aerodynamic force of a three-dimensional morphing aircraft while taking monotonicity into account.

A unique two-phase training strategy based on the Multi-Task Learning (MTL) method is utilized in network training. This approach allows the DLC to make better use of limited training data while conserving computing resources [21]. Furthermore, this strategy refines the prediction accuracy for both C_l and C_d.

This paper is organized as follows. In Section 2, the folding wing model and the method of constructing aerodynamic parameters dataset are explained. In Section 3, we introduce the architecture of DLC model and the two-phase network training method. In Section 4, the performance of the trained model in different flow conditions is evaluated. Finally, in Section 5, we summarize the current work and demonstrate the results of our research.

2 Data preparation

In this section, the preparation of networks training and testing datasets is discussed. The three-dimensional morphing aircraft model is briefly introduced. The three-dimensional CFD method used to obtain the aerodynamic forces is then illustrated and validated with wind tunnel experiment results.

2.1 Morphing aircraft model

The morphing aircraft are designed to be conveniently loaded and be quickly launched by mother platform. A typical morphing aircraft model [22] is demonstrated in Fig. 1, which includes a pair of folding wings, a fixed fuselage and a pair of V-tails. To realize safe launch and steady flight, the wings should be quickly unfolded in the air. In the unfolding process, the wings substantially vary in an open "V" style. In low-speed flight, the wing is almost straight with low-sweep. In high-speed flight, the wing morphs to a high-sweep angle according to the flight velocity. When in loading or launching state, the wing is absolutely folded under the fuselage belly. The aerodynamic characteristics during the unfolding process are the fundament for dynamic modeling, flight control and folding driving system design. Aerodynamic forces and moments vary in the

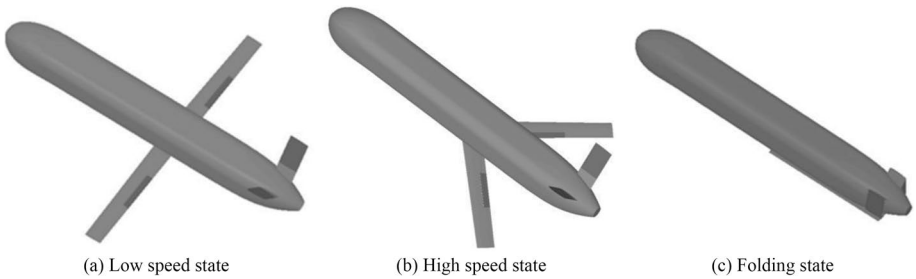

Fig. 1 Three typical states of the morphing aircraft

unfolding process. CFD method will be applied here to obtain the aerodynamic forces and moments in the unfolding process.

2.2 Computational fluid dynamics method and validation

The required aerodynamic parameters database of aircraft is obtained by CFD calculation. In numerical modeling, compressible Reynolds Averaged Navier–Stokes equations are solved with the finite volume method. The k-ω SST turbulence model is used for the current solver. The numerical simulations are carried out using the CFD software, ANSYS Fluent 2019 R1.

To simulate the dynamic unfolding process, structured overset meshes with a total cell number of 4.46 million were applied for all the following simulations. The overset meshes are divided into five components, one for the fuselage, two for the wings and two for the tails, as shown in Fig. 2a. Each component has an independent structured overset mesh. The background zone is divided into two parts, the inner structured zone and the far field unstructured zone, as shown in Fig. 2b.

The pressure far-field boundary condition ($Ma = 0.4$) is applied for the outer domain of the cuboid. All components of the aircraft are set as no-slip wall conditions. Overset boundary conditions are utilized for the outside domains of each component.

To verify the accuracy of the CFD method, a wind tunnel experiment was carried out for the 50% scaled aircraft model, in which four sweep states (unfolding angles $\beta = 0°, 30°, 60°, 90°$) are tested, respectively. In the experiment, the inflow Mach number is 0.4 and the aircraft model is under the angles of attack $\alpha = 4°$ and $8°$, respectively. Experimental data and CFD predictions for C_l and C_d are compared in Fig. 3. The results demonstrate that the simulation lines agree well with the experimental data both for C_l and C_d. The calculation accuracy of the CFD method is validated. For more details on aircraft models, wind tunnel experiments, and CFD methods, please refer to a previous article [22].

2.3 Dataset preparation

Nine cases with angles of attack $\alpha = [-4°, -2°, 1°, 2°, 4°, 6°, 8°, 10°, 12°]$ are selected as training dataset. Testing dataset contains 4 cases at $\alpha = [-5°, 3°, 9°, 14°]$. Two of them are interpolation datasets used to test the learning effect. The other two are extrapolation datasets used to test the generalization ability. Besides, two cases of

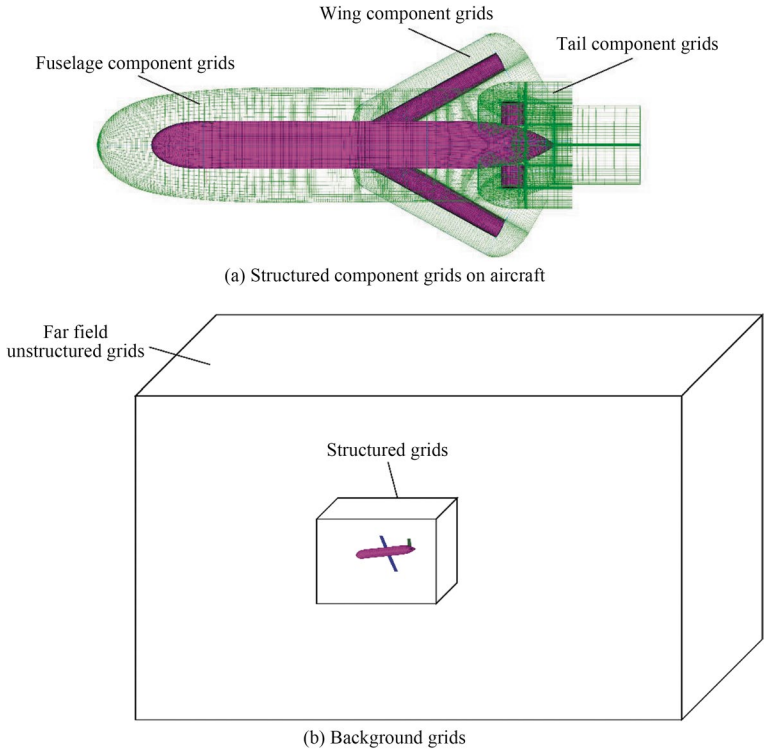

Fig. 2 Structured overset grids for CFD

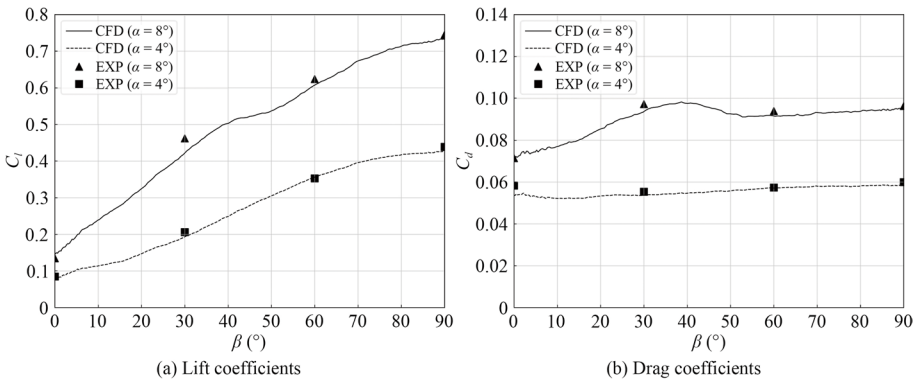

Fig. 3 Comparison between CFD results and experimental data under different unfolding angles

extremely small $\alpha = [-10°, -15°]$ are used to verify the monotonicity trend of aerodynamic parameters. The simulation time step is $\Delta t = 0.01$ s, including 201 time points during a whole morphing process when $T = 2$ s. Figure 4 shows seven aerodynamic parameters calculated by the CFD method. There are three input parameters for the DLC model: angle of attack α, time t and unfolding angle β. The outputs are the aerodynamic parameters C_l and C_d during the process of unfolding.

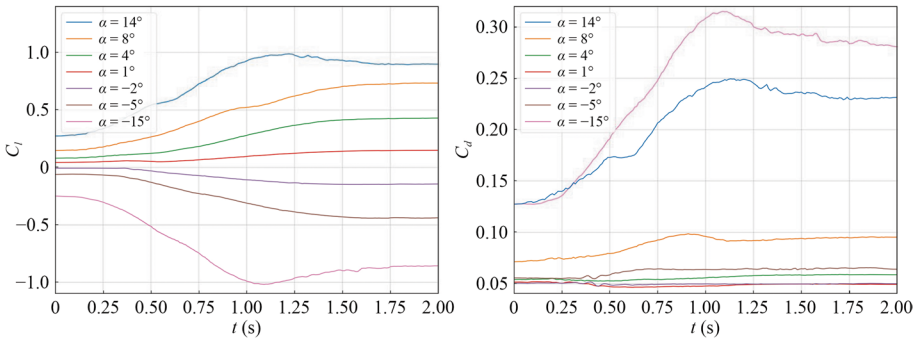

Fig. 4 Unsteady aerodynamic parameters in the unfolding process

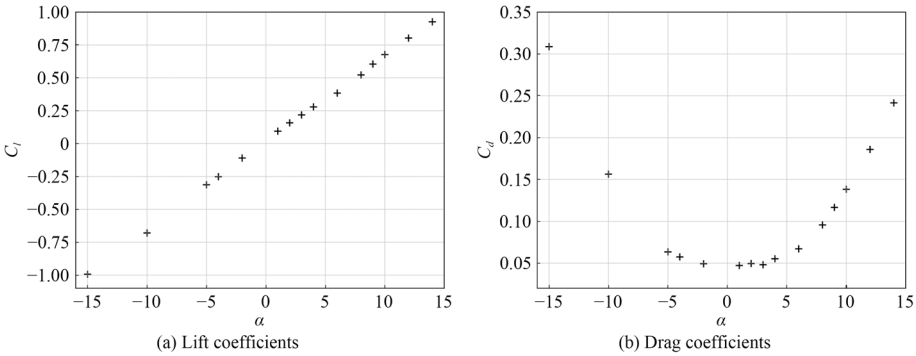

Fig. 5 Unsteady aerodynamic parameters at the unfolding time $t = 1.0$ s

3 Deep learning method

3.1 Deep Lattice Cross Network

Previous study on three-dimensional morphing aircraft unsteady aerodynamic force prediction focused on solving sparse input features problems using Cross Layer and got satisfactory results [20]. Although Cross Layer has incorporated some prior knowledge such as processing α and t, β separately, it still resembles a black box, lacking in interpretability. As depicted in Figs. 4 and 5, we possess crucial prior knowledge: C_l exhibits a monotonic increasing trend with respect to α, whereas C_d does not. Thus, ensuring the monotonicity of C_l within a particular range of α can make the prediction model more reliable and interpretable.

In this research, Lattice Layer [17, 19] is utilized to guarantee the monotonicity of C_l. For the prediction of C_d, Cross Layer [20] is still used to make use of the limited training data effectively. Besides, the MTL method is utilized to complete the prediction for both C_l and C_d.

The overall structure of DLC is shown in Fig. 6. t and β are processed by shared bottom *NShare* and then input into two shared towers: NCl, NCd. For the prediction of C_d, α is input into $N\alpha$ and Cross operation is performed to obtain the result value. For prediction of C_l, α is input into Lattice Layer to guarantee the monotonicity and inner product is performed to integrate the outputs of *NCl* and Lattice Layer. Multilayer Perceptron (MLP) layers are used in $NCl, NCd, N\alpha$ and *NShare*.

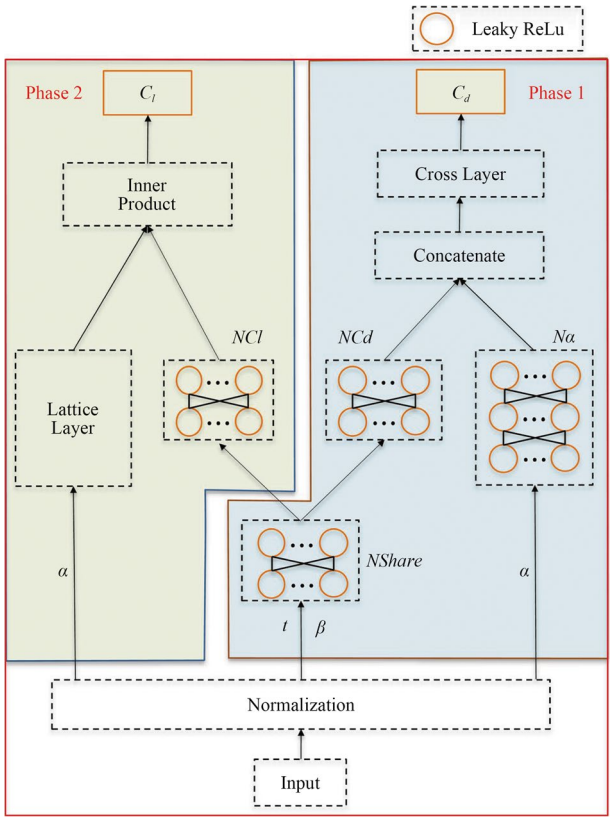

Fig. 6 Overall structure of DLC

Cross Layer [20] was proposed to effectively fuse the external flow field and aircraft properties while making full use of sparse input features and limited training data. The Cross Layer is made up of product part, linear part and bias term, which has the formula:

$$C_d = \sum_{i=1}^{d} \sum_{j=i+1}^{d} w_{Pij} x_i x_j + \boldsymbol{w_L x} + b_{C_d}, \tag{1}$$

where $\boldsymbol{x}$ denotes the vector concatenated by $N\alpha$ with NCd. $x_i, x_j \in \boldsymbol{x}$, and $\boldsymbol{w_L}$ denotes the weight matrix of linear part. w_{Pij} denotes the element of product part weight matrix. b_{C_d} denotes the bias term. The architecture of Cross Layer is shown in Fig. 7.

3.2 Lattice Layer

The classical DLN [19] model has been modified to make the lattice method with monotonicity guarantees more suitable for predicting aerodynamic parameters. Three types of modules are contained in the Lattice Layer: Monotonic Embedding Layer (MEL), Calibration Layer and Ensemble of Lattices Layer.

Different from Linear Embedding Layer used in DLN, MEL is the simplest neural network that only contains input and output layers without hidden layers. To ensure that the outputs of MEL are monotonic with respect to inputs, network weight values

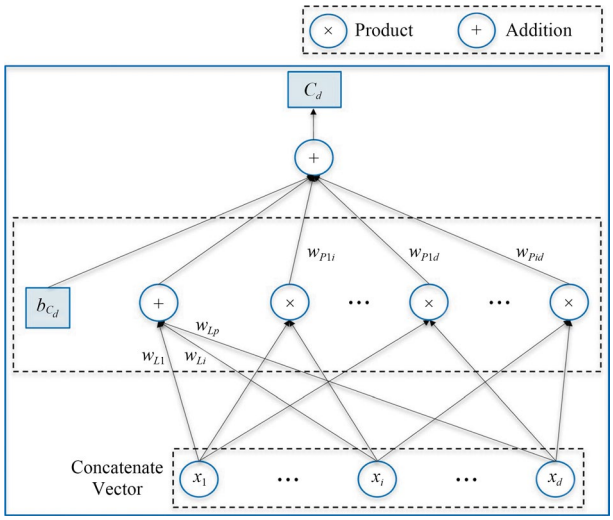

Fig. 7 Architecture of Cross Layer for C_d prediction

are constrained to be non-negative. Without activation function, MEL only performs linear mapping. The mapping formula is:

$$y_E = W_{mo}x_E + b, \tag{2}$$

where W_{mo} denotes the non-negative weight matrix, b denotes the bias, x_E is the input vector, and y_E is the output vector. Basically, MEL plays a role in embedding input features in Lattice Layer to expand the dimension of input features and enhance the learning ability of the deep network. Therefore, the length of the output vector y_E is generally greater than the length of the input vector x_E. In this research, as shown in Fig. 6, the input of Lattice Layer only contains one feature α.

Calibration Layer is a one-dimensional piecewise linear transform, which has a fixed input range $[x_{CL,min}, x_{CL,max}]$ and a fixed key point K_p. The two-dimensional coordinates of the k-th $(k = 1, \cdots, K_p)$ key point are (a_k, b_k). a is uniformly distributed within $[x_{CL,min}, x_{CL,max}]$ and has $a_1 = x_{CL,min}, a_k = x_{CL,max}$. In order to constrain the Calibration Layer to be monotonic with monotonic inputs, linear inequality constraints are added.

$$b_k \leq b_{k+1}, (k = 1, \ldots, K_p - 1). \tag{3}$$

The range of output $[y_{CL,min}, y_{CL,max}]$ is generally set as [0, 1]. An intuitive Calibration Layer example with five fixed key points is shown in Fig. 8.

Ensemble of Lattices Layer consists of G lattices. Lattice is a linearly interpolated multidimensional look-up table. For an S-dimensional lattice, the input is also S-dimensional and has a parameter vector $\boldsymbol{\theta_L}$ that contains 2^S parameters. In this study, all parameters $\boldsymbol{\theta_L}$ are used to process input α and are independent of inputs t and β. Multilinear interpolation method is used in this research [19]. For the S-dimensional input $\boldsymbol{x_L}$ representing the angle of attack α processed by MEL and Calibration Layer, it is not a single value, but a vector: $\boldsymbol{x_L}(d) \in [0,1]$, where d denotes the d-th component of $\boldsymbol{x_L}$. The interpolated value $f_L(\boldsymbol{x_L}) = \varnothing(\boldsymbol{x_L})^T \boldsymbol{\theta_L}$, and $\varnothing(\boldsymbol{x})$ is the result of a non-linear feature transformation

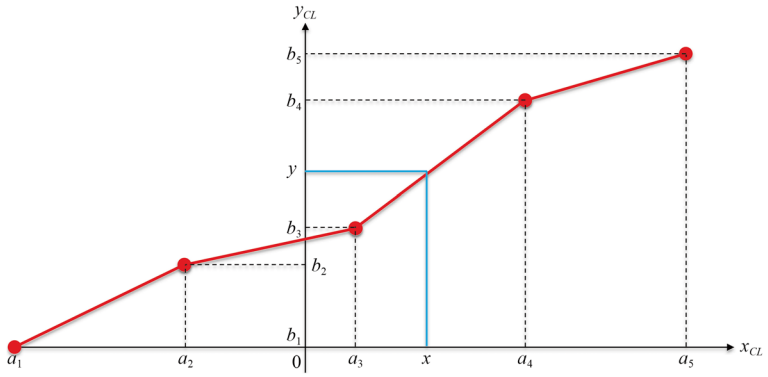

Fig. 8 A Calibration Layer with five fixed key points

that transforms the S-dimensional input $\boldsymbol{x_L}$ to 2^S-dimensional feature, which has the formula:

$$\varnothing(\boldsymbol{x_L})[j] = \prod_{d=1}^{S} \boldsymbol{x_L}(d)^{v_j(d)}(1-\boldsymbol{x_L}(d))^{1-v_j(d)}, \tag{4}$$

where j is the j-th parameter in 2^S-dimensional feature $\varnothing(x)$. $v_j = [p_1, \cdots, p_S]$, which can be regarded as the vertex coordinates of an S-dimensional cube with a side length of 1.

In order to ensure the monotonicity of $f_L(\boldsymbol{x_L})$ with respect to a specified input $\boldsymbol{x_L}(d)$, for all k, k', one must check that $\boldsymbol{\theta_L}(k') > \boldsymbol{\theta_L}(k)$ for each pair of adjacent k and k' in the lattice direction of the input, i.e., $v_k(d) = 0, v_{k'}(d) = 1$ and $v_k(m) = v_{k'}(m)$ for all $m \neq d$. $\boldsymbol{x_L}$ consists of multiple Calibration Layer outputs concatenated together, as the output of Calibration Layer is only a value. Therefore, for all $_d$, the monotonicity of $f_L(\boldsymbol{x_L})$ with respect to $\boldsymbol{x_L}(d)$ should be ensured.

For example, when $S = 2$, a schematic diagram of the unit lattice hypercube is shown in Fig. 9. In this study, $_S$ is set to 3 to achieve higher prediction accuracy while ensuring reasonable computational complexity. A 2×2 lattice interpolated with multilinear interpolation has the formula:

$$\begin{aligned} f_L(\boldsymbol{x_L}) =& \boldsymbol{\theta_L}(1)(1-\boldsymbol{x_L}(1))(1-\boldsymbol{x_L}(2)) + \boldsymbol{\theta_L}(2)(1-\boldsymbol{x_L}(1))\boldsymbol{x_L}(2) + \\ & \boldsymbol{\theta_L}(3)\boldsymbol{x_L}(1)(1-\boldsymbol{x_L}(2)) + \boldsymbol{\theta_L}(4)\boldsymbol{x_L}(1)\boldsymbol{x_L}(2). \end{aligned} \tag{5}$$

Assume that $f_L(\boldsymbol{x_L})$ is monotonically increasing with respect to $\boldsymbol{x_L}(1)$, the partial derivative must satisfy $\partial f_L(\boldsymbol{x_L})/\boldsymbol{x_L}(1) > 0$. For $v_k(1) = 0, v_{k'}(1) = 1$ and $v_k(2) = v_{k'}(2)$, according to Fig. 9, it can be concluded that $\boldsymbol{\theta_L}(4) > \boldsymbol{\theta_L}(2)$ and $\boldsymbol{\theta_L}(3) > \boldsymbol{\theta_L}(1)$. As $\boldsymbol{x_L}(2) \in [0,1]$, it can be proven that:

$$\partial f_L(\boldsymbol{x_L})/\boldsymbol{x_L}(1) = (1-\boldsymbol{x_L}(2))(\boldsymbol{\theta_L}(3) - \boldsymbol{\theta_L}(1)) + \boldsymbol{x_L}(2)(\boldsymbol{\theta_L}(4) - \boldsymbol{\theta_L}(2)) > 0. \tag{6}$$

Thus $f_L(\boldsymbol{x_L})$ is monotonically increasing with respect to feature $\boldsymbol{x_L}(1)$. Readers can refer to Gupta et al. for details [17].

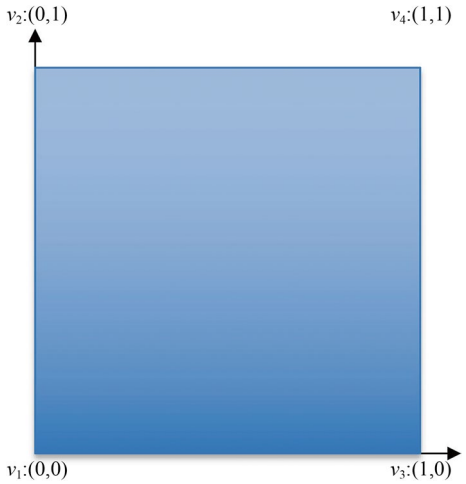

Fig. 9 Two-dimensional lattice diagram

As all three modules (Monotonic Embedding Layer, Calibration Layer, and Ensemble of Lattices Layer) can ensure that the output is monotonic with respect to a specified input, the Lattice Layer can as well. In this research, the input of Lattice Layer only contains α, considerably reducing the complexity of ensuring monotonicity.

3.3 Two-phase training method

The DLC model will be used to map unsteady aerodynamic parameters of the morphing aircraft at various angles of attack. The mapping functions of the aerodynamic parameter prediction layers are:

$$\begin{cases} C_l = f_{Cl}(\alpha, \beta, t, \boldsymbol{\theta_{Cl}}, \boldsymbol{\theta_{share}}) \\ C_d = f_{Cd}(\alpha, \beta, t, \boldsymbol{\theta_{Cd}}, \boldsymbol{\theta_{share}}) \end{cases}, \tag{7}$$

where C_l, C_d denote the aerodynamic parameters, lift coefficients and drag coefficients. α denotes the angle of attack, t denotes time, and β denotes the wing unfolding angle. $\boldsymbol{\theta_{Cl}}, \boldsymbol{\theta_{Cd}}$ are the trainable network parameters only for predictions of C_l and C_d, and $\boldsymbol{\theta_{share}}$ is the trainable network parameters for $Nshare$.

For the training process of DNN, the selection and adjustment of loss functions are an important part of model tuning [23]. In this study, after repeated experiments, Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE) were used to train C_l and C_d separately. However, it's difficult to balance these two loss functions to obtain satisfactory training results if C_l and C_d are trained simultaneously. In order to solve this problem and reduce the training cost of finding the optimal equilibrium point that may exist, two-phase training method is introduced in this model.

As shown in Fig. 6, the training process of DLC is divided into two phases. For phase 1, network parameters $\boldsymbol{\theta_{Cd}}$ and $\boldsymbol{\theta_{share}}$ are trained. All network parameters that take part in C_d prediction are trained in this phase. MAPE is chosen as the loss function of phase 1, which is defined as:

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{C-\widehat{C}}{\widehat{C}}\right|, \tag{8}$$

where C denotes the prediction values and $\widehat{C}$ denotes the real result based on CFD simulation.

Network parameters $\boldsymbol{\theta_{Cd}}$ and $\boldsymbol{\theta_{share}}$ are fixed after trained in phase 1. For phase 2, $\boldsymbol{\theta_{Cl}}$ are trained and MAE is chosen as the loss function, which is defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|C-\widehat{C}\right|. \tag{9}$$

MAE is an absolute measure of prediction error, while MAPE is a relative measure that expresses the error as a percentage of the actual value. MAE is more straightforward to interpret as it is in the same units as the data, whereas MAPE provides a relative measure that can be useful for comparing models across different scales or datasets. Since the optimization objective is to reduce the relative error between the prediction and CFD results, MAPE was selected as the overall evaluation metric to assess the quality of training results.

Network weights and biases are trained by the backpropagation algorithm. The optimizer used in this research is Adaptive Moment Estimation, a stochastic gradient descent method that calculates the adaptive learning rate of each parameter [24]. TensorFlow is applied to establish the deep network structure and train this model [25].

The number of layers and nodes, the activation function of each layer, and the learning rate are the primary hyperparameters that are crucial for deep network training. The hyperparameters are repeatedly optimized to obtain a suitable set. The learning rate is set as 0.001 to balance training speed and accuracy, the batch size is set as 64, and the activation function in all MLP layers is a nonlinear activation function, Leaky ReLU. The Multi-Task Cross model (MTC) [20] is used to demonstrate the constraints of DLC on monotonicity. For each experiment, 1000 training epochs are conducted until the loss function on the validation set no longer falls in both DLC and MTC models.

4 Results and discussion

4.1 Validation of the deep lattice cross model

Figure 10 shows the comparison between CFD simulation values and DLC prediction results of 50 random validation samples. From Fig. 10, both prediction results of C_l and C_d agree well with CFD results on the random validation samples. The prediction errors of C_l and C_d are relatively modest, and the CFD simulation results correspond well with the DLC model predictions.

The generalization ability and scalability of DLC are then evaluated by testing datasets that have not been used in training, as shown in Fig. 11. The testing datasets consist of two interpolation datasets, $\alpha = [3°, 9°]$, which are used to verify the learning effect, and other two extrapolation datasets, $\alpha = [-5°, 14°]$, which aim to test the generalization ability. Prediction results in Fig. 11 illustrate very low errors for both C_l and C_d during a

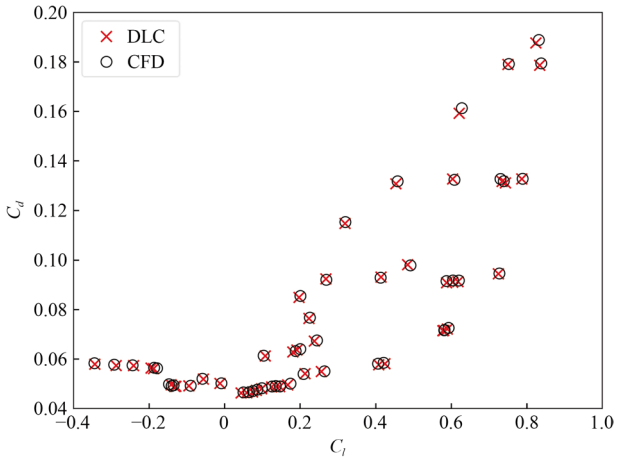

Fig. 10 Comparison of lift coefficients and drag coefficients for the validation samples

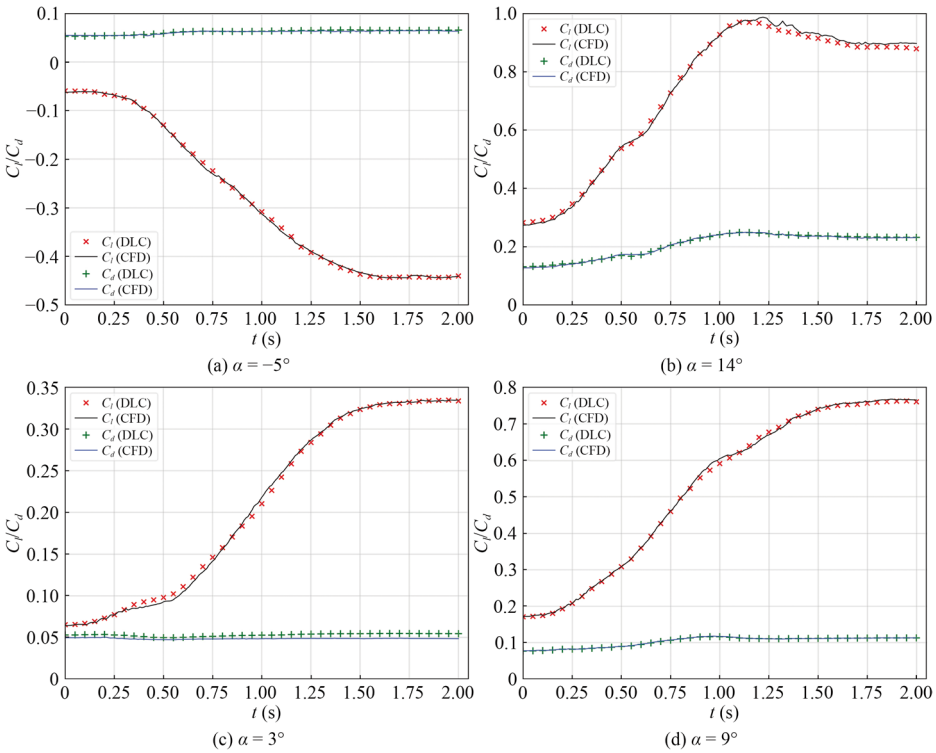

Fig. 11 Comparison between the computational fluid dynamics values and the predicted values for the four testing datasets

whole unfolding process for all four test datasets. Specifically, for C_l, results show good agreement in all four test cases. As for C_d, when α is small, the prediction error is slightly larger, as shown in Fig. 11c. For larger α, the C_d prediction curve fits well with CFD results.

As shown in Fig. 12, three time nodes during the unfolding process, i.e., $t = [0.5\ \text{s}, 1.0\ \text{s}, 2.0\ \text{s}]$, are extracted to obtain the curves of C_l, C_d and lift-drag ratio, K,

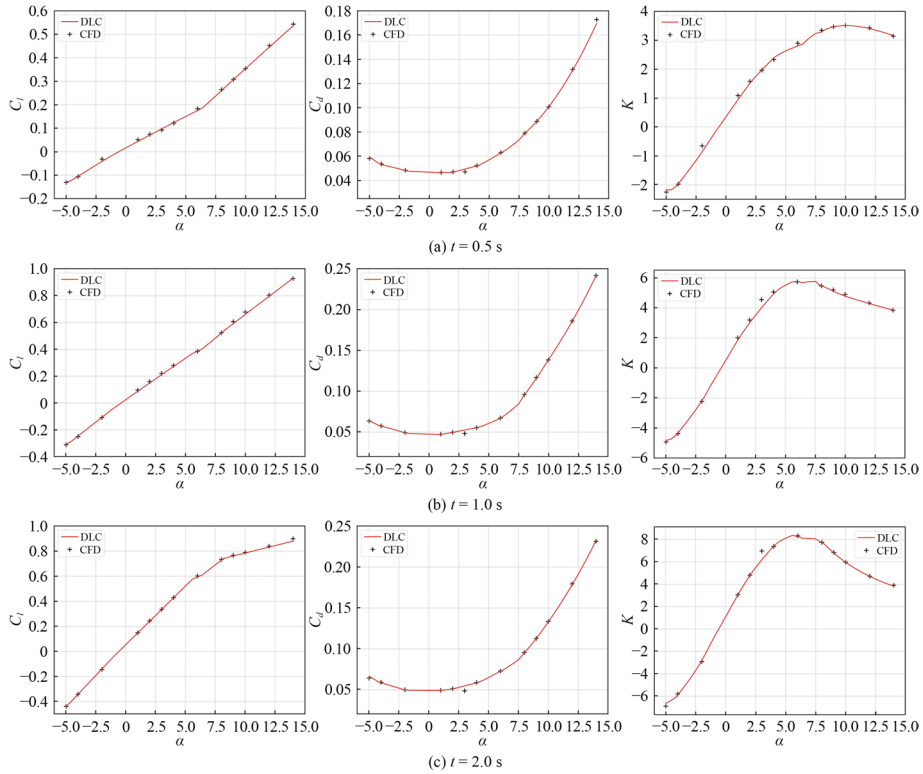

Fig. 12 Comparison of aerodynamic parameters between the DLC predictions and CFD results at **a** $t = 0.5$ s, **b** $t = 1.0$ s, and **c** $t = 2.0$ s

Table 1 Mean absolute percentage error comparison of lift coefficients and drag coefficients on test datasets

Model	DLC	MLP	MTC
C_l	1.35%	2.39%	1.76%
C_d	3.10%	6.11%	3.97%

changing with the angles of attack respectively. It can be observed that the predicted values of the DLC model fit the CFD results well. For K, DLC can accurately predict the value of α that corresponds to the maximum K. When α is large or small, the predicted K is still consistent with the CFD result.

Table 1 displays the average MAPE of C_l and C_d at four angles of attack in the test datasets. It demonstrates that the DLC model has the highest prediction accuracy compared to the other two non-monotonic prediction models. The prediction accuracy of the MTC model has also improved when compared to MLP. Specifically, DLC reduces the sum MAPE of C_l and C_d by around 1.3% compared to MTC, and around 4% compared to MLP.

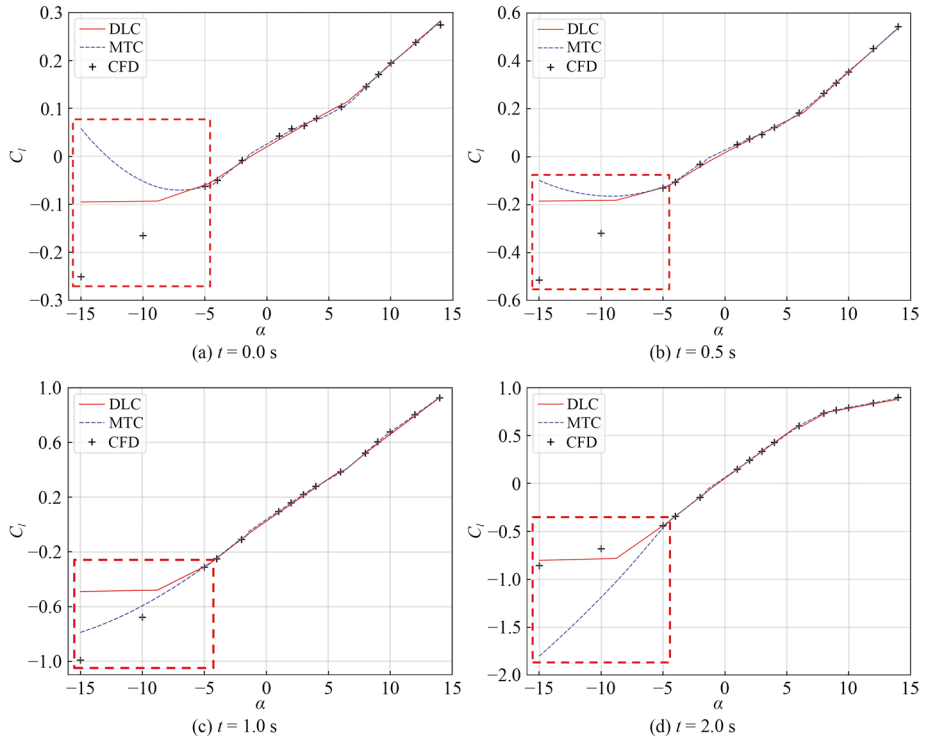

Fig. 13 Comparison of C_l in a wider range between the DLC and MTC predictions at **a** $t = 0.0$ s, **b** $t = 0.5$ s, **c** $t = 1.0$ s, and **d** $t = 2.0$ s

4.2 Monotonicity verification of various prediction models

The monotonicity of DLC and another non-monotonic prediction model, MTC, is compared. Comparisons of the predicted C_l at unfolding time $t = [0.0\text{ s}, 0.5\text{ s}, 1.0\text{ s}, 2.0\text{ s}]$ in a wider α range are shown in Fig. 13. In order to verify the role of DLC in ensuring monotonicity, α is extended to a very small value of $-15°$ that was not covered in the training and testing datasets.

According to Fig. 13, all three prediction models can describe the monotonicity characteristics of C_l for α in the range $-5°$–$14°$, which is covered in the training and testing datasets. However, the reasons that lead to the monotonicity of prediction results for these prediction models are different. For MTC, the reason is that the training data is monotonic. As for DLC, it is mainly due to the limitations of the model structure.

When $\alpha \in (-15° - -5°)$, CFD results indicate that C_l still maintains a monotonic increasing trend. CFD results illustrate the trend of C_l changes when the value of α is small, in order to demonstrate the rationality of ensuring monotonicity of the prediction model. Within these small angles of attack range, MTC results show obvious non-monotonic characteristics in Fig. 13a and b, while results of DLC can keep the monotonicity even at a quite small α. As shown in Fig. 13c and d, the prediction results of MTC also exhibit monotonicity in the latter half of the unfolding process. This suggests that monotonic datasets may make the non-monotonic network exhibit monotonicity, but this is not guaranteed. In order to ensure that the

prediction results exhibit monotonicity with respect to specific inputs, imposing constraints on neural networks is a useful and feasible method.

5 Conclusions

Lift coefficients always increase with the angle of attack within a reasonable range. To enhance the reliability and interpretability of predictions for morphing aircraft that exhibit highly unsteady aerodynamic characteristics, we integrated a monotonicity constraint into a deep learning framework, the Deep Lattice Cross Network. This model employs the Deep Lattice Network and its enhancements to ensure the monotonic behavior of lift coefficient relative to the angle of attack, while utilizing the Cross Layer for predicting the non-monotonic drag coefficient. By incorporating the Multi-Task Learning method, the model efficiently predicts both lift and drag coefficients, optimizing the use of limited training data and conserving computational resources through a two-phase deep network training strategy that yields accurate results for both parameters.

Two interpolation and two extrapolation datasets are utilized to evaluate the generalization ability and scalability of the Deep Lattice Cross Network. For all four testing cases, the prediction results show high agreement with computational fluid dynamics results. Specifically, the Deep Lattice Cross Network reduces the sum Mean Absolute Percentage Error of C_l and C_d by around 1.3% when compared to other state-of-the-art prediction models. However, while the model excels in ensuring monotonicity, its generalization capabilities across an extremely broad range of angles of attack ($\alpha = -10°$ and $-15°$) require further refinement.

The Deep Lattice Cross Network, with its embedded monotonicity constraints, offers superior predictive accuracy. This deep learning model, designed to uphold physical monotonicity, is not only applicable to lift prediction in aircraft but also holds promise for a variety of aerodynamic parameter predictions exhibiting monotonic traits, potentially serving as a submodule within broader aircraft design models.

Acknowledgements
Not applicable.

Authors' contributions
Jiachi Zhao: Conceptualization, Validation, Formal analysis, Visualization, Writing; Lifang Zeng: Validation, Formal analysis, Writing; Aoxue Lin: Data Preparation; Xueming Shao: Funding acquisition, Supervision.

Funding
This research was supported by the National Natural Science Foundation of China (Grants No. 12202384 and No. U2241274) and the Defense Industrial Technology Development Program (Grants No. JCKY2023205B013 and No. JCKY2021205B003).

Data availability
The datasets used and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 2 September 2024 Revised: 2 November 2024 Accepted: 18 November 2024
Published online: 15 May 2025

References

1. Sekar V, Jiang Q, Shu C et al (2019) Fast flow field prediction over airfoils using deep learning approach. Phys Fluids 31:057103
2. Zhu L, Zhang W, Kou J et al (2019) Machine learning methods for turbulence modeling in subsonic flows around airfoils. Phys Fluids 31:015105
3. Karniadakis GE, Kevrekidis IG, Lu L et al (2021) Physics-informed machine learning. Nat Rev Phys 3:422–440
4. Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 378:686–707
5. Yang L, Meng X, Karniadakis GE (2021) B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. J Comput Phys 425:109913
6. Raissi M, Yazdani A, Karniadakis GE (2020) Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science 367(6481):1026–1030
7. Cai S, Wang Z, Fuest F et al (2021) Flow over an espresso cup: inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks. J Fluid Mech 915:A102
8. Sun L, Gao H, Pan S et al (2020) Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. Comput Methods Appl Mech Eng 361:112732
9. Ling J, Kurzawski A, Templeton J (2016) Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J Fluid Mech 807:155–166
10. Sheng H, Yang C (2021) PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries. J Comput Phys 428:110085
11. Darbon J, Meng T (2021) On some neural network architectures that can represent viscosity solutions of certain high dimensional Hamilton-Jacobi partial differential equations. J Comput Phys 425:109907
12. Lu L, Dao M, Kumar P et al (2020) Extraction of mechanical properties of materials through deep learning from instrumented indentation. Proc Natl Acad Sci U S A 117(13):7052–7062
13. Sivaraman A, Farnadi G, Millstein T et al (2020) Counterexample-guided learning of monotonic neural networks. In: Proceedings of the 34th international conference on neural information processing systems (NIPS'20). Curran Associates Inc, Red Hook, pp 11936–11948
14. Archer NP, Wang S (1993) Application of the back propagation neural network algorithm with monotonicity constraints for two-group classification problems. Decis Sci 24(1):60–75
15. Daniels H, Velikova M (2010) Monotone and partially monotone neural networks. IEEE Trans Neural Netw 21(6):906–917
16. Sill J (1998) Monotonic networks. In: Proceedings of the 1997 conference on advances in neural information processing systems 10 (NIPS'97). MIT Press, Cambridge, pp 661–667
17. Gupta M, Cotter A, Pfeifer J et al (2016) Monotonic calibrated interpolated look-up tables. J Mach Learn Res 17(109):1–47
18. Canini K, Cotter A, Gupta MR et al (2016) Fast and flexible monotonic functions with ensembles of lattices. In: Proceedings of the 30th international conference on neural information processing systems (NIPS'16). Curran Associates Inc, Red Hook, pp 2927–2935
19. You S, Ding D, Canini K et al (2017) Deep lattice networks and partial monotonic functions. In: Proceedings of the 31st international conference on neural information processing systems (NIPS'17). Curran Associates Inc, Red Hook, pp 2985–2993
20. Zhao J, Zeng L, Shao X (2023) A novel prediction method for unsteady aerodynamic force on three-dimensional folding wing aircraft. Aerosp Sci Technol 137:108287
21. Vandenhende S, Georgoulis S, Proesmans M et al (2020) Revisiting multi-task learning in the deep learning era. arXiv preprint, arXiv:2004.13379
22. Zeng L, Liu L, Shao X et al (2023) Mechanism analysis of hysteretic aerodynamic characteristics on variable-sweep wings. Chin J Aeronaut 36(5):212–222
23. Wang Q, Ma Y, Zhao K et al (2022) A comprehensive survey of loss functions in machine learning. Ann Data Sci 9(2):187–212
24. Kingma DP, Ba JL (2014) ADAM: a method for stochastic optimization. arXiv preprint, arXiv:1412.6980
25. Abadi M, Agarwal A, Barham P et al (2016) TensorFlow: large-scale machine learning on heterogeneous systems. arXiv preprint, arXiv:1603.04467