An investigation of neural networks for aerodynamic predictions

Kensley Balla*, Ruben Sevilla, Oubay Hassan, Kenneth Morgan

Zienkiewicz Centre for Computational Engineering, College of Engineering,
Swansea University, Bay Campus, SA1 8EN, Wales, United Kingdom.
*977456@swansea.ac.uk

Abstract

This work proposes a novel multi-output neural network for the prediction of the lift coefficient of aerofoils using inviscid compressible flow data. Contrary to existing neural networks that are designed to predict aerodynamic quantities of interest, the proposed network considers as output the pressure at a number of selected points on the aerofoil surface. The proposed approach is compared against the more traditional network where the lift coefficient is directly the only output of the network. Furthermore, a detailed comparison of the proposed neural network against the popular proper orthogonal decomposition method is presented. The numerical results, involving high dimensional problems with flow and geometric parameters, show the benefits of the proposed approach.

Key words: *neural network, proper orthogonal decomposition, lift, CFD, NURBS.*

1 Introduction

During the design and optimisation stages of aerodynamic components, the simulations to be performed involve a large number of parameters, both geometric and related to the flow conditions. In this scenario, the simulation of all possible configurations is commonly not affordable. The use of reduced order models (ROM) has become a popular alternative to alleviate the computational burden associated to CFD simulations involving a large number of parameters. In the last two decades, the use of artificial neural networks (NN) in CFD applications has grown significantly. One of the most attractive properties of ROMs and NNs is that, once the ROM or NN is built, predictions can be performed in almost real time. This enables the use of such models for the fast evaluations of an objective function in optimisation and inverse problems.

2 Governing equations

The fluid flow problems considered in this work are governed by the Euler equations for an inviscid compressible fluid. The strong form of the problem, in a computational domain $\Omega \subset \mathbb{R}^d$ and in the absence of external volume forces, can be written as

$$\begin{aligned} \boldsymbol{U}_t + \boldsymbol{\nabla} \cdot \boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{0} \quad & \text{in } \Omega \times (0, T] \\ \boldsymbol{U} = \boldsymbol{U}_0 \quad & \text{in } \Omega \times \{0\} \\ \boldsymbol{B}(\boldsymbol{U}, \boldsymbol{U}^\infty) = \boldsymbol{0} \quad & \text{in } \partial\Omega \times (0, T]. \end{aligned} \tag{1}$$

The vector of conservation variables, $\boldsymbol{U}$, and the flux tensor, $\boldsymbol{F}$, are given by

$$\boldsymbol{U} := \begin{Bmatrix} \rho \\ \rho \boldsymbol{v} \\ \rho E \end{Bmatrix}, \qquad \boldsymbol{F} := \begin{bmatrix} \rho \boldsymbol{v}^T \\ \rho \boldsymbol{v} \otimes \boldsymbol{v} + p \mathbf{I}_d \\ (\rho E + p) \boldsymbol{v}^T \end{bmatrix}, \tag{2}$$

Here, $\boldsymbol{U}_0$ denotes the initial condition, T is the final time and $\boldsymbol{B}$ is the generic flux used to define the boundary conditions over the boundary. In the above expressions ρ is the density, $\rho \boldsymbol{v}$ is the momentum, $\rho \boldsymbol{E}$ is the total energy per unit volume, p is the pressure and $\mathbf{I}_d$ is the identity matrix of dimension d. A vertex-centred finite volume (FV) solver is used to build the training and test datasets [1].

3 Artificial neural networks

Artificial NNs are an arrangement of neurons where neurons of each layer are connected to the neurons of the previous and next layers. The first and the last layers correspond to the inputs and outputs, and the remaining layers, called *hidden layers*, are numbered from $l = 1$ to $l = \mathtt{n_L}$, with $\mathtt{n_L}$ being the number of hidden layers [2]. During the so-called *forward propagation*, the value associated to each neuron is computed by using the values associated to the connected neurons in the previous layer, the weights of the connections and an *activation function* F^l. More precisely, the value of the j-th neuron in the layer $l+1$, denoted by z_j^{l+1}, is computed as

$$z_j^{l+1} = F^l\left(\sum_{i=1}^{\mathtt{n_N^l}} \theta_{ij}^l z_i^l + b_j^l\right), \tag{3}$$

where b^l is a bias that is introduced to enhance the approximation properties of the network, θ_{ij}^l denotes the weight of the connection between the i-th neuron of the layer l and the j-th neuron of the layer $l+1$ and $\mathtt{n_N^l}$ is the number of neurons in the layer l.
A *training case* is defined by a vector of N inputs, $\boldsymbol{x} = \{x_1, \ldots, x_N\}^T$, and a vector of M outputs, $\boldsymbol{y}(\boldsymbol{x}) = \{y_1(\boldsymbol{x}), \ldots, y_M(\boldsymbol{x})\}^T$. Given a set of $\mathtt{n_{Tr}}$ training cases, $\boldsymbol{x}^k = \{x_1^k, \ldots, x_N^k\}$ and $\boldsymbol{y}^k = \{y_1^k, \ldots, y_M^k\}$, for $k = 1, \ldots, \mathtt{n_{Tr}}$, the so-called *cost function* for a function approximant is defined as

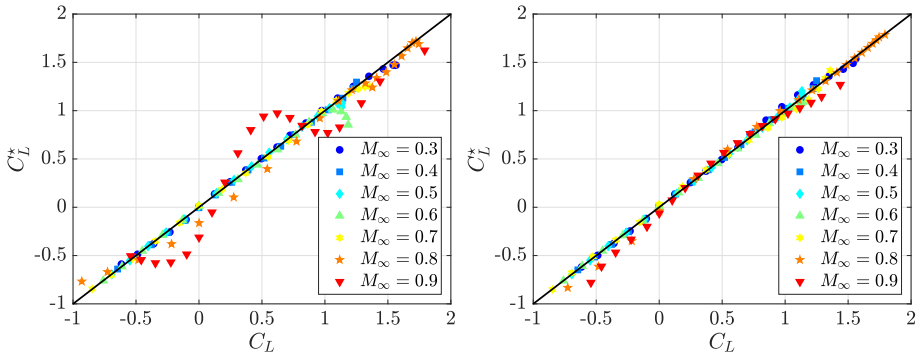
$$C(\boldsymbol{\theta}) = -\frac{1}{\mathtt{n_{Tr}}} \sum_{k=1}^{\mathtt{n_{Tr}}} \sum_{i=1}^{\mathtt{n_N^{n_L+1}}} \left[y_i^k(\boldsymbol{x}^k) - h_i^k(\boldsymbol{\theta})\right]^2 + \frac{\lambda}{2\mathtt{n_{Tr}}} \sum_{l=0}^{\mathtt{n_L}} \sum_{i=1}^{\mathtt{n_N^{l+1}}} \sum_{j=1}^{\mathtt{n_N^l}} (\theta_{ij}^l)^2. \tag{4}$$

The values h_i^k correspond to the predicted outputs, computed in the forward propagation, starting from the input values $z_i^0 = x_i^k$ for $k = 1, \ldots, \mathtt{n_{Tr}}$ and $i = 1, \ldots, \mathtt{n_N^0}$. It is worth noting that the number of neurons in the input layer is taken as $\mathtt{n_N^0} = N$ and, similarly, the number of neurons in the output layer is taken as $\mathtt{n_N^{n_L+1}} = M$.
The goal of the so-called *training stage* is to obtain the weights associated to all the connections of the NN that minimise the cost function of equation (4). In this work, the backpropagation momentum gradient descent is employed where a low pass filter is applied to alleviate the common difficulty of gradient based methods to reach a global minimum in problems where multiple local minimums are present.
When the NN is designed as a function approximant, the activation functions are selected as the log sigmoid function for $l = 0, \ldots \mathtt{n_L} - 1$ and a linear function in the outputs. In contrast, for NN designed as classifiers, all the activation functions are taken as the sigmoid for $l = 0, \ldots \mathtt{n_L}$.

4 Numerical examples

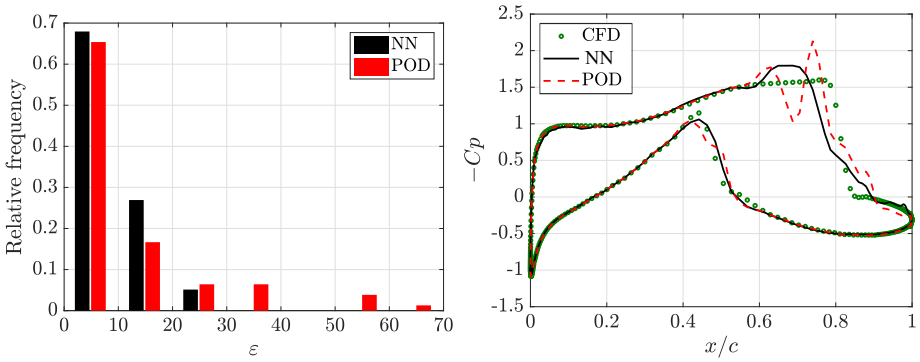
4.1 The benefits of using a multi-output neural network

The first example considers the computation of the lift coefficient of a NACA0012 aerofoil at a free stream Mach number, M_∞, and an angle of attack, α, in predefined intervals $I_M = (0.3, 0.9)$ and $I_\alpha = (-5°, 16°)$, respectively. It should be noted that this range of the inflow conditions considered leads to both subsonic and transonic flows.
Two NNs are considered and compared. The first network considers M_∞ and α as inputs and the lift coefficient, C_L, as a single output. The second network considers M_∞ and α as inputs and the output is the pressure at a user defined set of points on the aerofoil. The set of points considered corresponds to the 300 mesh nodes used to discretise the aerofoil.
To illustrate the performance of both networks in the different flow regimes considered, figure 1 shows the regression plots for the lift coefficient using the two networks, where different symbols are used for different free-stream Mach numbers to show the accuracy of the predictions in terms of the flow regime.



(a) Network 1 (b) Network 2

Figure 1: Regression plot for the lift coefficient as a function of the free-stream Mach number M_∞.



(a) overall performance (b) $M_\infty = 0.77$, $\alpha = 2.7°$

Figure 2: The relative frequency of the error in the test dataset, measured in lift counts is shown in (a), and the comparison of pressure coefficient predictions, C_p of the two ROMs with the CFD solver is shown in (b)

4.2 A comparison on the performance of neural network and proper orthogonal decomposition

The second example considers the prediction of the lift on a RAE2822 aerofoil at a free stream Mach number, M_∞, and an angle of attack, α, in predefined intervals $I_M = (0.3, 0.9)$ and $I_\alpha = (-5°, 12°)$ respectively.

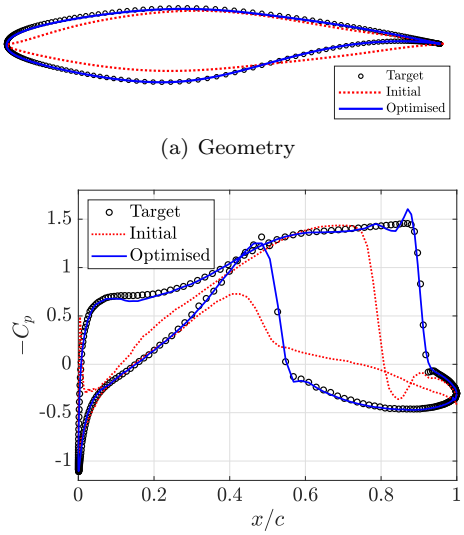
This example is used to compare the performance of the NN with multiple outputs proposed in the previous example against a popular reduced order modelling technique, the POD. In both ROMs, the prediction of the pressure is performed at the 300 points used to discretise the aerofoil and the lift coefficient is computed. Figure 2(b) describes how the ROMs pressure prediction oscillates near the shock, with more oscillations observed in the POD. The histogram of figure 2(a) show that both approaches are able to provide an error below 10 lift counts for almost 70% of the test cases, with the NN achieving a marginally higher percentage.

4.3 An inverse shape optimisation using geometrically parametrised aerofoils

The last example employs a trained NN algorithm to the inverse shape design of an aerofoil, with the goal to show the potential and reliability of the proposed NN for the fast evaluation of the objective function in an optimisation process. This example can be broken down into

two stages. The first stage involves the prediction of the lift coefficient for an aerofoil that is parametrised using the control points of the NURBS describing the aerofoil. The base geometry corresponds to an approximation of the NACA0012 aerofoil using two cubic B-splines with eight control points to define the top and bottom curves respectively, as proposed in [3]. The variation of the position of the control points with respect to the base geometry, namely $(\pm\delta x_i, \pm\delta y_i)$ is considered to be the input of the neural network, where $(\delta x_i, \delta y_i) \in [0, 0.1c]^2$ and c denotes the chord of the aerofoil. In the second stage, the trained NN algorithm is used to perform an inverse shape design given a pressure distribution and using a gradient-free algorithm, namely the modified cuckoo search (MCS), proposed in [4]. The reference pressure distribution used in MCS is that of an RAE2822 aerofoil, unseen in the training stage of the NN.

Figure 3(a) offers a visual comparison of the target geometry, the initial geometry used in the MCS algorithm and the final optimised geometry while figure 3(b) shows their corresponding pressure coefficient plots with minimal relative error in the $\mathcal{L}^2((0, c))$ norm, as measured by the objective function. It is worth mentioning that the optimisation process using the MCS algorithm took only 55 seconds due to the almost negligible cost of evaluating the objective functions using the proposed NN.



(a) Geometry

(b) Pressure coefficient

Figure 3: Target, initial and optimised geometry and pressure coefficient obtained using the MCS and the proposed NN to predict the values of the objective function.

5 Conclusions

A new multi-output NN has been proposed to predict the lift coefficient of aerofoils using inviscid compressible flow data. The proposed approach has been compared to a single-output NN and to the POD using numerical examples with inflow and geometric parameters. In all cases considered, the proposed NN produces more accurate predictions for the same number of training cases or snapshots. Finally, the potential of the proposed NN approach to enable the fast evaluation of the objective function in an inverse shape design problem has been demonstrated. Using a gradient-free optimisation algorithm and the predictions given by the proposed NN, it was possible to perform an inverse identification of the aerofoil geometry for a given pressure distribution in less than one minute.

References

[1] K. A. Sørensen, O. Hassan, K. Morgan, N. P. Weatherill, A multigrid accelerated hybrid unstructured mesh method for 3D compressible turbulent flow, Computational Mechanics 31 (1-2) (2003) 101–114.

[2] M. Hagan, H. Demuth, M. Beale, O. DeJesus, Neural network design, 2nd edition, Martin Hagan, 2014.

[3] R. Sevilla, S. Fernández-Méndez, Numerical integration over 2D NURBS-shaped domains with applications to NURBS-enhanced FEM, Finite Elements in Analysis and Design 47 (10) (2011) 1209–1220.

[4] S. Walton, O. Hassan, K. Morgan, M. Brown, Modified cuckoo search: A new gradient free optimisation algorithm, Chaos, Solitons & Fractals 44 (9) (2011) 710–718.