scientific reports

OPEN

Learning aerodynamics with neural network

Wenhui Peng[1]✉, Yao Zhang[2], Eric Laurendeau[2] & Michel C. Desmarais[1]

We propose a neural network (NN) architecture, the Element Spatial Convolution Neural Network (ESCNN), towards the airfoil lift coefficient prediction task. The ESCNN outperforms existing state-of-the-art NNs in terms of prediction accuracy, with two orders of less parameters. We further investigate and explain how the ESCNN succeeds in making accurate predictions with standard convolution layers. We discover that the ESCNN has the ability to extract physical patterns that emerge from aerodynamics, and such patterns are clearly reflected within a layer of the network. We show that the ESCNN is capable of learning the physical laws and equation of aerodynamics from simulation data.

In recent years, significant efforts have been conducted to apply NNs to fundamental science research. Two representative works revolutionized fundamental scientific research. Noe et al. created PauliNet, a deep learning method that is capable of solving Schrödinger equation[1]. Scientists from DeepMind designed the AlphaFold, a type of NN that can generate models of proteins far more accurate than any that have come before[2]. A diverse collection of intersections between NNs and physics is presented in the review paper by Carleo et al. These applications range from statistics and quantum physics to high energy and cosmology[3].

The fluid dynamics community is no exception. The potential of using NNs to tackle fluid mechanics problems has lately been gained increasingly attention[4,5]. Jin et al. proposed NSFnets that applied NNs to solve the Navier-Stokes equations[6]. Kochkov et al. applied NNs to accelerate Computational Fluid Dynamics (CFD) simulations, and achieved a significant reduction in computation cost[7]. Li et al. further improved the computational efficiency by approximating the Navier-Stokes equations with the Fourier neural operator[8]. Once trained, these NN models can make inference within seconds and can be extremely efficient compared with traditional CFD approaches[9–15].

However, these NN models suffer from generalization problems: their prediction accuracy drops dramatically once the parameters of the partial differential equations change[7]. To resolve this issue, Raissi et al. proposed the Physics-Informed Neural Networks (PINNs). They developed a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations[16]. In the PINN framework, prior physical knowledge is introduced as constraint, such that NNs are trained to solve supervised learning tasks while respecting the given laws of physics described by general nonlinear partial differential equations[16]. With the physical constrains, the PINNs manage to achieve state-of-the-art prediction accuracy[6,17–21].

Despite these significant improvements towards prediction accuracy, scientists are not fully convinced with the results of NNs due to the lack of interpretability. How do PINNs solve physical problems? Do they make inference based on physical principles or just probability? Answering these questions is necessary, since people need interpretations to understand the logic driving the learned model, and make sure that the predicted results are trustworthy. However, interpreting the complex interactions between parameters and layer nodes has always been a challenging task[22–25]. This is due to the nature of NNs, also known as the ”black-box” issue: instead of solving the tasks with logical steps, NNs learn by examples and adjust parameters to improve their performance over time[26].

Recent progress has been made to interpret how NNs solve scientific problems. In the field of mathematics, NNs have been shown to be able to aid mathematicians in discovering new conjectures and theorems[27]. In the field of chemistry, NNs have been shown to be able to capture the complex pattern of electrons moving around the nucleus[1]. In the domain of game theory, evidence has been found that human knowledge is acquired by the AlphaZero neural network as it trains on the game of chess[28].

Is it possible that the NNs are able to develop physical insight in the process of solving physical problems, just like in other scientific domains?

Motivated by this question, in this work, we pierce the “black box” of a NN and investigate how NN solves a physical problem in aerodynamics. Our contributions are as follow: (1) we proposed a NN model that adopts a novel physics-informed structured input, the ESCNN, it outperforms existing state-of-the-art NNs in the airfoil lift coefficient prediction task. (2) We provide insight explanations as to how the ESCNN succeeds in making accurate predictions with standard convolution layers.

[1]Department of Computer Engineering, Polytechnique Montreal, Montreal, QC, Canada. [2]Department of Mechanical Engineering, Polytechnique Montreal, Montreal, QC, Canada. ✉email: wenhui.peng@polymtl.ca

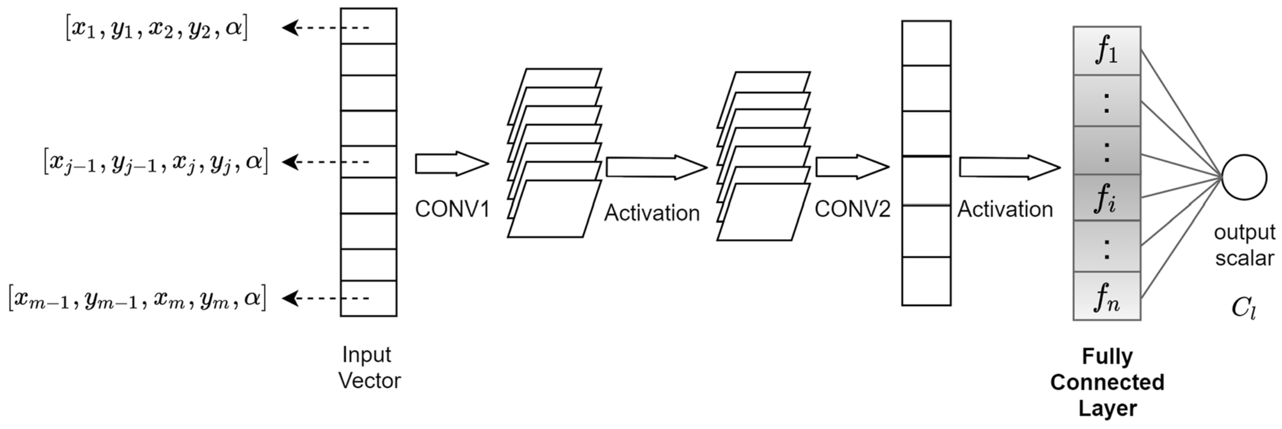
$[x_1, y_1, x_2, y_2, \alpha]$
$[x_{j-1}, y_{j-1}, x_j, y_j, \alpha]$
$[x_{m-1}, y_{m-1}, x_m, y_m, \alpha]$
Input Vector
CONV1
Activation
CONV2
Activation
f_1
f_i
f_n
Fully Connected Layer
output scalar
C_l

Figure 1. ESCNN architecture.

Model	Parameters	Error
Linear regression	322	9.82%
MLP	961	8.43%
Multilayer perceptron	131,969	5.58%
AeroCNN[33]	266,145	3.46%
ESCNN	1561	0.97%

Table 1. Performance benchmarks of different models.

Predicting lift coefficient with neural networks

Calculating the airfoil lift coefficient is one of the most critical tasks in aerodynamics. It is generally achieved by using traditional Computational Fluid Dynamics (CFD) methods, which are often known for being computational expensive. We propose a neural network based model: Element Spatial Convolution Neural Network (ESCNN), to efficiently predict the airfoil lift coefficient[29].

ESCNN is an end-to-end neural network that takes the airfoil coordinates x_j, y_j and angle of attack α as input, and gives the lift coefficient C_l as output. The ESCNN architecture is straightforward. As shown in Fig. 1, it consists of two standard convolution layers followed by non-linear activation function, and a fully connected layer before the prediction[29]. The two convolution layers use 200 convolution filters, where the filter size of first layer is 5×1 and the second layer is 1×1 respectively. The last layer, fully connected layer, contains 159 neurons. Note that since the ESCNN takes sequential airfoil coordinates as input, and only convolutions and activations are involved in the architecture, therefore all the hidden layers are not permutation invariant.

The airfoil data samples are taken from the database of UIUC Applied Aerodynamic Group[30], which covers a wide range of airfoil types from real-world designs. Each airfoil is represented by 160 points in x, y format. For the flow conditions, the Mach number *Mach* and Reynolds number *Re* are fixed to 0.3 and 1.3×10^7 respectively, and the angle of attack α varies from -2 to $10°$ to avoid flow separation (laminar flow). The ground truth of C_l is computed with the CFD solver Xfoil[31], for each angle of attack at the specified flow condition for each airfoil geometry. In total 15678 samples are generated to create the final dataset. The whole dataset is divided into training set and validation set at the ratio of 8:2. Note that in laminar flow where the dataset is generated, the lift coefficient C_l varies linearly with the angle of attack α, however the relationship between C_l and the airfoil coordinates is highly-nonlinear[32].

Table 1 shows the performance benchmarks of different models, where the regression models are provided as baselines, and three types of NNs are compared. Experiments are implemented on the Pytorch and MindSpore open-source deep learning frameworks. The Multilayer Perceptron (MLP) network model has three hidden layers, where each layer contains 256,128,128 neurons respectively. The AeroCNN is a recent NN model that achieves state-of-the-art prediction accuracy[33]. In the AeroCNN framework, the airfoils are processed as images, such that AeroCNN can adopt the typical convolution neural network architecture for image recognition[33]. The error ε is the relative error defined by Eq. (1), where $\hat{C}_l$ denotes the predicted lift coefficient and C_l denotes the ground truth lift coefficient.

$$\varepsilon = \frac{\|\hat{C}_l - C_l\|_2}{\|C_l\|_2}. \tag{1}$$

It is noted that even the baseline regression models can bring the error down below 10%. However, further reducing the prediction error is a difficult task: the model's learning capacity must be large enough to accurately approximate the non-linear relationship between C_l and airfoil coordinates. Generally, the error can be further

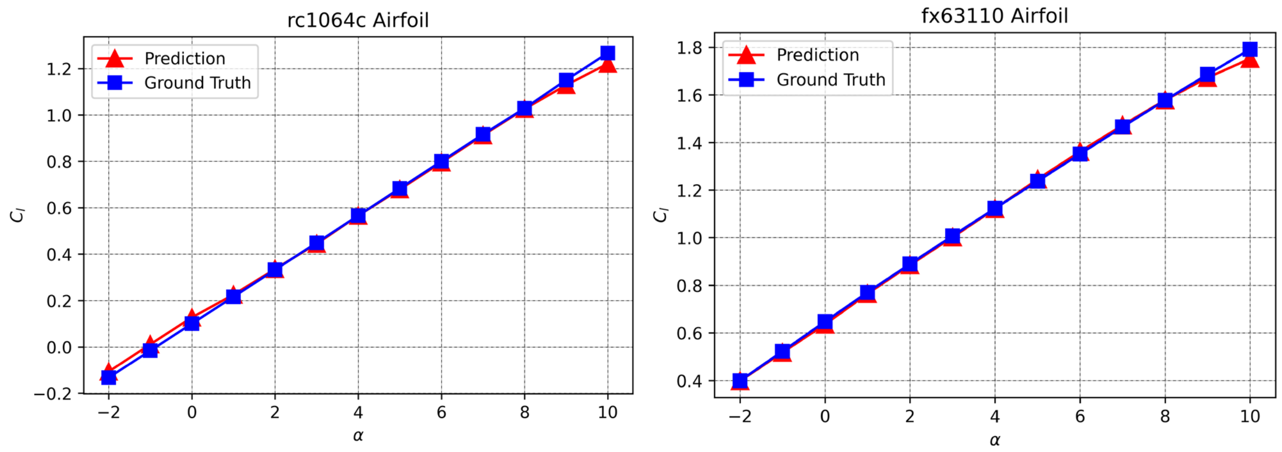

Figure 2. Prediction performance on validation airfoils.

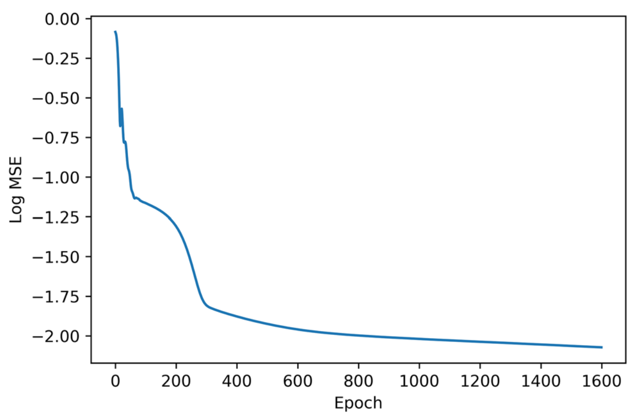

Figure 3. ESCNN training learning curve.

reduced at the cost of adding more parameters to increase the model's complexity[34], provided sufficient training data. However, the cost becomes more significant as the error gets smaller: the parameters are doubled to reduce the error from 5.58% (Multilayer Perceptron) to 3.46% (AeroCNN).

In contrast, the ESCNN achieves 0.97% error with two orders of parameters less than AeroCNN. Figure 2 shows examples of the ESCNN prediction performance on the validation set, where the predictions are accurate and the $C_l - \alpha$ linearity is well captured. With such few parameters and limited capacity, how can ESCNN perform so well in a high-dimensional and highly-nonlinear aerodynamic system?

Learning the Kutta condition

While investigating how ESCNN learns to make prediction, the fully connected layer, as shown in Fig. 1, draws our attention, since it is the last hidden layer before prediction.

We track the neuron values of the fully connected layer with a test airfoil at 3° angle of attack during the training. The test airfoil NACA 2412 contains 160 coordinates. Variations of the neuron values $[f_1, f_2, \ldots, f_n]$ at the end of different training stages are shown in Fig. 4, where the x-axis denote the sequential index number $[1, 2, \ldots, n]$ of the neurons, and the y-axis denote corresponding neuron values $[f_1, f_2, \ldots, f_n]$. Since the ESCNN model takes airfoil coordinates that are sequentially formatted as input, the order of coordinates is reflected in the neurons of learned hidden layers. The epoch numbers represent different training iterations, the network learns and evolves with the increasing of training epoch numbers. From the figure, it is obvious that as the training progresses, the fully connected layer is converging to a sine-shaped pattern, and this pattern begins to stabilize at around 400 epochs as shown in Fig. 4d. This result is also consistent with the learning curve that is shown in Fig. 3, as in this figure, after 400 epochs, the fitted Mean Squarer Error (MSE) converges towards a constant.

We notice an interesting phenomena, as shown in Fig. 4, the first and the last neurons values are always the same ($f_1 = f_n$) even at the very beginning of training. It seems like the ESCNN has learned, early in the training process, of the condition $f_1 = f_n$, and followed this rule strictly during the training progress. This $f_1 = f_n$ pattern reminds us the fundamental principle in aerodynamics: Kutta condition.

Recall the Kutta condition[32]: in fluid flow around a body with a sharp corner, the flow pattern in which fluid approaches the corner from both directions meets at the corner before flows smoothly go away from the body,

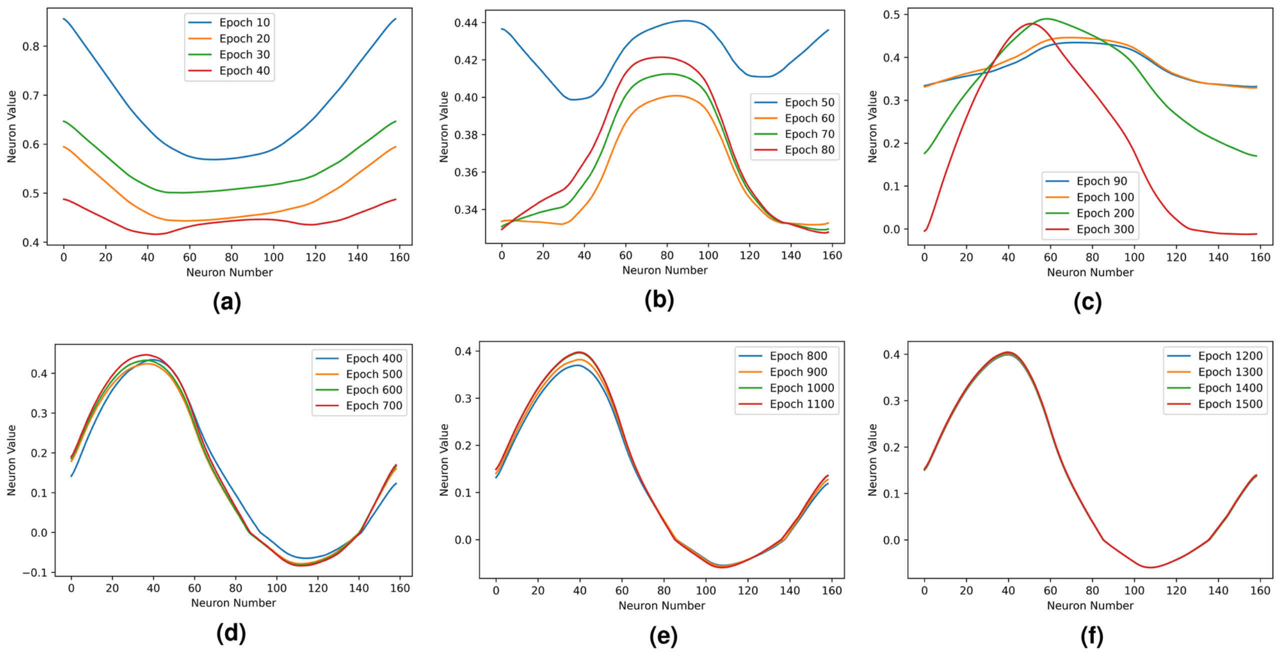

Figure 4. Fully connected layer evolution in training.

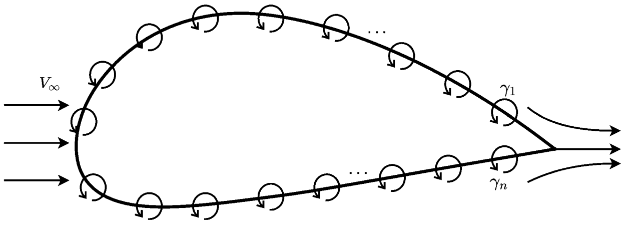

Figure 5. Airfoil vortex panels in free stream.

as shown in Fig. 5. Equation (2) describes the Kutta condition mathematically, where γ_1 and γ_n represents the vortex strength at the upper and lower surface of the airfoil trailing edge respectively.

$$\gamma_1 = \gamma_n. \tag{2}$$

Apparently, ESCNN manages to figure out the importance of the Kutta condition by itself—it prioritizes to keep the value of the first neuron and the last neuron the same during the entire learning process.

Learning the vortex strength distribution pattern

To further investigate the meaning of the sine-shaped pattern, we compute the vortex strength distribution over airfoil using Vortex Panel Method (VPM). VPM is an engineering numerical method to compute the vortex strength distribution over airfoil. It replaces the airfoil surface with a series of vortex panels.

Figure 6 shows the scaled vortex strength distribution calculated by VPM over the test case NACA 2412 airfoil at 3° angle of attack. As shown in this figure, when fewer panels are used in the calculation, the distributions of vortex strength , γ, are not smooth. The oscillations from one panel to another is a well-known flaw of VPM which is triggered by the numerical inaccuracy[32]. With the larger number of panels used during the calculation, the oscillations fades away, the results get more accurate, and the sine-shaped pattern of vortex distribution gradually emerges.

Figure 7 compares the computed 160 panels vortex strength distribution with the converged layer pattern at the same scale, both curves show a similar sine-shaped pattern, except they are symmetric in relation to the horizontal axis. Furthermore, if we down-sample the number of input airfoil coordinates to fewer dimensions, the converged sine-shaped pattern are still kept by the trained network, as shown in Fig. 8, not subject to the numerical inaccuracy of VPM.

Both the converged layer pattern and the computed VPM vortex strength distribution exhibit a similar sine-shaped wave. One possible hypothesis is that the network has learned another physical concept that is a function of the vortex strength. Although there is no aerodynamics law at the moment that theoretically addresses this

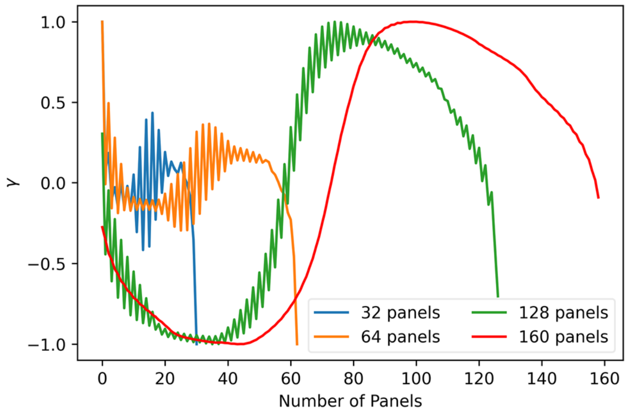

Figure 6. Computed vortex distribution as a function of the number of panels.

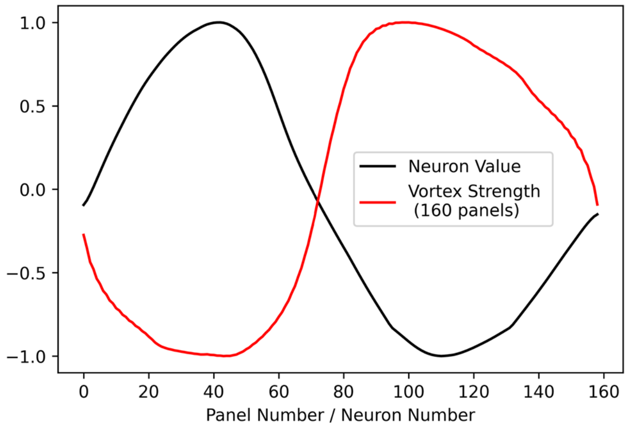

Figure 7. Comparison of the converged layer pattern with 160 panels vortex strength distribution.

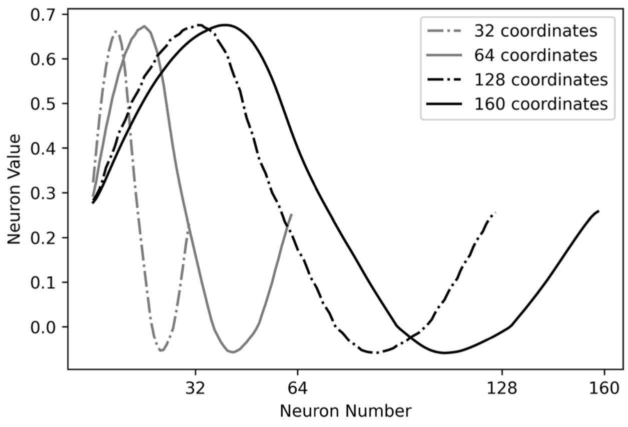

Figure 8. Neuron values of the fully connected layer with different number of airfoil coordinates.

sine-shaped pattern, we surmise that the distribution of vortex strength over airfoil follows a certain underlying pattern.

Learning with ReLU activation

The initial activation function we use to train the ESCNN model is the LeakyReLU, a common choice for neural networks[35]. More importantly, the output domain of LeakyReLU activation can be both positive and negative values, so is the value of vortex strength. This feature enables ESCNN to learn the vortex related physical quantity.

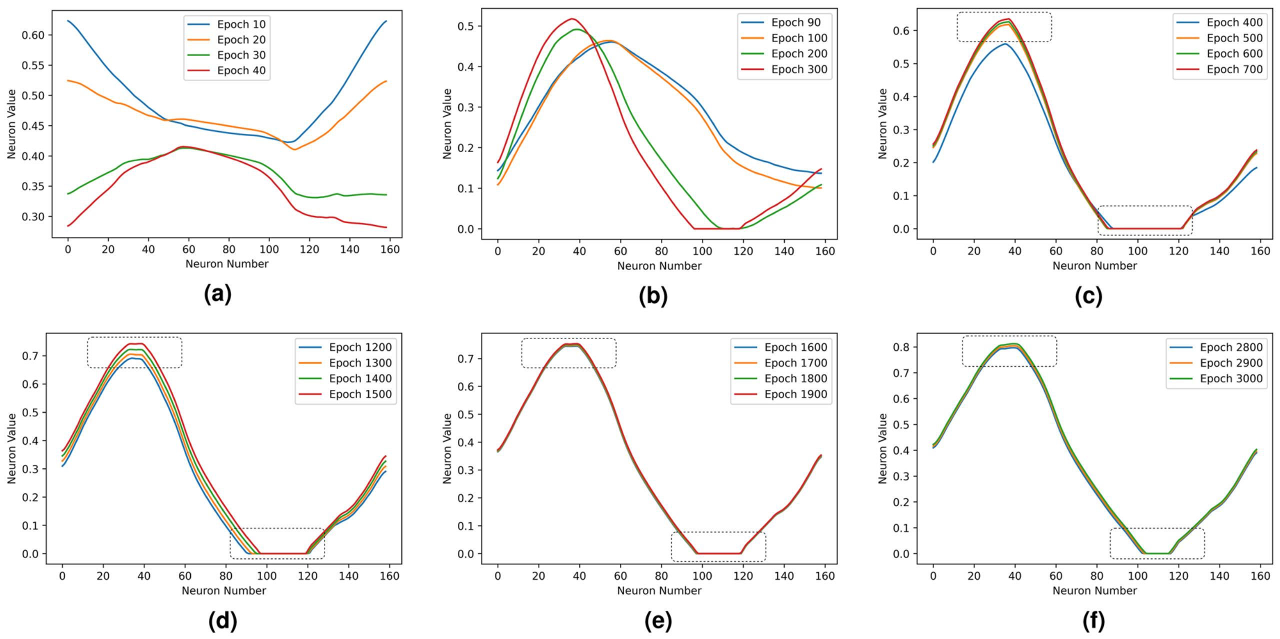

Figure 9. Fully connected layer under constrained activation.

However, what if the network is trained with a ReLU activation function, can it still learn the vortex related physical quantity? In this section, we intentionally limit the value range of the fully connected layer by implementing another activation—ReLU function[35]. Equations (3) and (4) describes the LeakyReLU and ReLU activation function respectively, where $\mathbf{k} = 0.5$ is the negative slope coefficient. For any real-valued input, the LeakyReLU outputs both positive and negative values, whereas ReLU outputs non-negative values only[35].

$$\text{LeakyReLU }(x) = \max(0, x) + \mathbf{k} * \min(0, x). \tag{3}$$

$$\text{ReLU}(x) = (x)^+ = \max(0, x). \tag{4}$$

Figure 9 shows that the ESCNN still learns the Kutta condition and vortex distribution pattern, although the pattern is constrained within the non-negative range. More interestingly, the fully connected layer is forcing itself to evolve into a symmetric pattern as the training goes on, as marked in the dotted rectangular in Fig. 9. Despite that there is no constraint placed in the positive domain, the evolution to such symmetric pattern is quite clear in both Figs. 4 and 9.

Learning the lift coefficient equation

The ESCNN contains standard convolution layers, how can it outperform existing neural networks with significantly less parameters? The key reason of success is that the ESCNN makes prediction by learning the lift coefficient equation of aerodynamics at the last layer. For an airfoil at a fixed angle of attack, the lift coefficient C_l is given by Eq. (5), where c is a constant[32], γ_i is the vortex strength at the panel i, and l_i is the length of the panel i. Meanwhile, the prediction target $\hat{C}_l$ is obtained at the last layer of ESCNN (fully connected layer), as described by Eq. (6), where f_i represents the neuron values and w_i represents the learned weight parameters.

$$C_l = c\sum_{i=1}^{n} \gamma_i l_i. \tag{5}$$

$$\hat{C}_l = \sum_{i=1}^{n} f_i w_i. \tag{6}$$

We show that the ESCNN model learns Eq. (5) by matching the neuron values $[f_1, f_2, \ldots, f_n]$ with the vortex strength over airfoil $[\gamma_1, \gamma_2, \ldots, \gamma_n]$, and matching the learned weights $[w_1, w_2, \ldots, w_n]$ with the length of each panels $[l_1, l_2, \ldots, l_n]$.

We calculate the correlation coefficient between the neuron values $[f_1, f_2, \ldots, f_n]$ and the vortex strength over airfoil $[\gamma_1, \gamma_2, \ldots, \gamma_n]$, for each sample in the testing dataset. The correlation coefficient cov is defined by Eq. (7), where $\mathbf{x}$ and $\mathbf{y}$ are both vectors. Results show that the neuron values and vortex strength of all testing samples are highly correlated: the average of correlation coefficient is -0.975, with a standard deviation of 6.4×10^{-3}, an example is shown in Fig. 7.

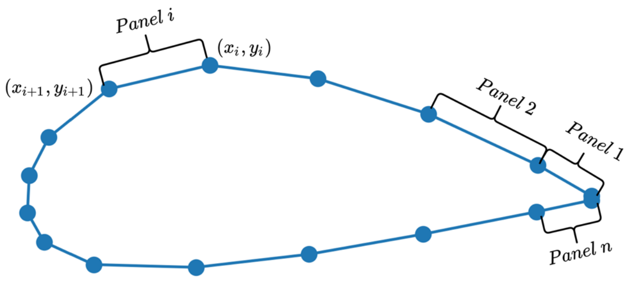

Figure 10. Panel distribution over the airfoil surface.

$$cov = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}. \tag{7}$$

We also calculate the correlation coefficient between the length of each panels $[l_1, l_2, \ldots, l_n]$ and the learned weights $[w_1, w_2, \ldots, w_n]$. Note that the learned weights are fixed after training, however, the length of each panels $[l_1, l_2, \ldots, l_n]$ varies across different airfoils. The panels of an airfoil refer to the segments between two sequential points, as shown in Fig. 10. The panels are serially numbered from 1 to n according to their locations on the airfoil surface (starting from the trailing edge, along the upper surface to the leading edge and back around the lower surface to trailing edge). The length of each panel l_i is the spatial distance between two sequential coordinates as defined by Eq. (8). Figure 11 shows the randomly sampled airfoils, and their corresponding panel length distribution.

Figure 12 compares the learned weights and the panel length distribution of example airfoils at the same scale. Despite that panel length distributions are different across airfoils, they are close to symmetric. This symmetric pattern explains why the fully connected layer evolves for symmetry during training as shown in Figs. 4 and 9. The learned weights are not as highly correlated with the panel length distribution, as compared with the correlation between neuron values and vortex strength. However, the symmetric pattern of panel length distribution has been captured by the learned weights. Moreover, we notice from Table 2 that the correlation between the panel length and learned weights affects the prediction performance: higher correlation leads to lower prediction error. Different airfoils have different panel length distributions, whereas the learned weights are fixed, failing to match the weights for corresponding airfoil leads to the increase of prediction error.

$$l_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}. \tag{8}$$

How does ESCNN manages to learn the physical quantity of vortex strength and the panel length? The reason is that ESCNN adopts a structured input that incorporates prior physical knowledge.

In the vortex panel method, the vortex strength γ_1 to γ_n are obtained by solving a linear system of n equations, and solving each equation requires the angle of attack α and the coordinates of corresponding panel[32]. Inspired by the vortex panel method, we combine the panel coordinates $x_i, y_i, x_{i+1}, y_{i+1}$ and the angle of attack α as an element unit, and then concatenate all the element units into a single vector, as shown in Fig. 1. The first layer, CONV1, performs $1D$ convolution over each element unit, and each element unit contains sufficient information to solve the vortex strength and panel length.

The physics-informed structured input allows the convolution layer to pick up the vortex strength γ_i and panel length l_i, and further allows the fully connected layer to learn the lift coefficient equation.

Conclusions

In this work, we propose a neural network model that outperforms existing state-of-the-art NNs in the airfoil lift coefficient prediction task, with two orders of less parameters. We further investigate how the ESCNN makes accurate predictions.

The ESCNN network learns to constrain the first and last neurons to be equal, during the entire learning process. This is the evidence that it self-learns the fundamental aerodynamics principle, the Kutta condition. Moreover, the fully connected layer converges to a sine-shaped wave pattern that is highly correlated to the vortex strength distribution over airfoil, which demonstrates that the network learns the vortex related physical quantity. In addition, we explore ESCNN's learning ability with constrained activation by replacing LeakyReLU to ReLU function. The results show that even with a limited value range of the neurons, ESCNN can still learn the critical physics of Kutta condition and the vortex distribution pattern. In the end, we show that the network learns lift coefficient equation at the last layer.

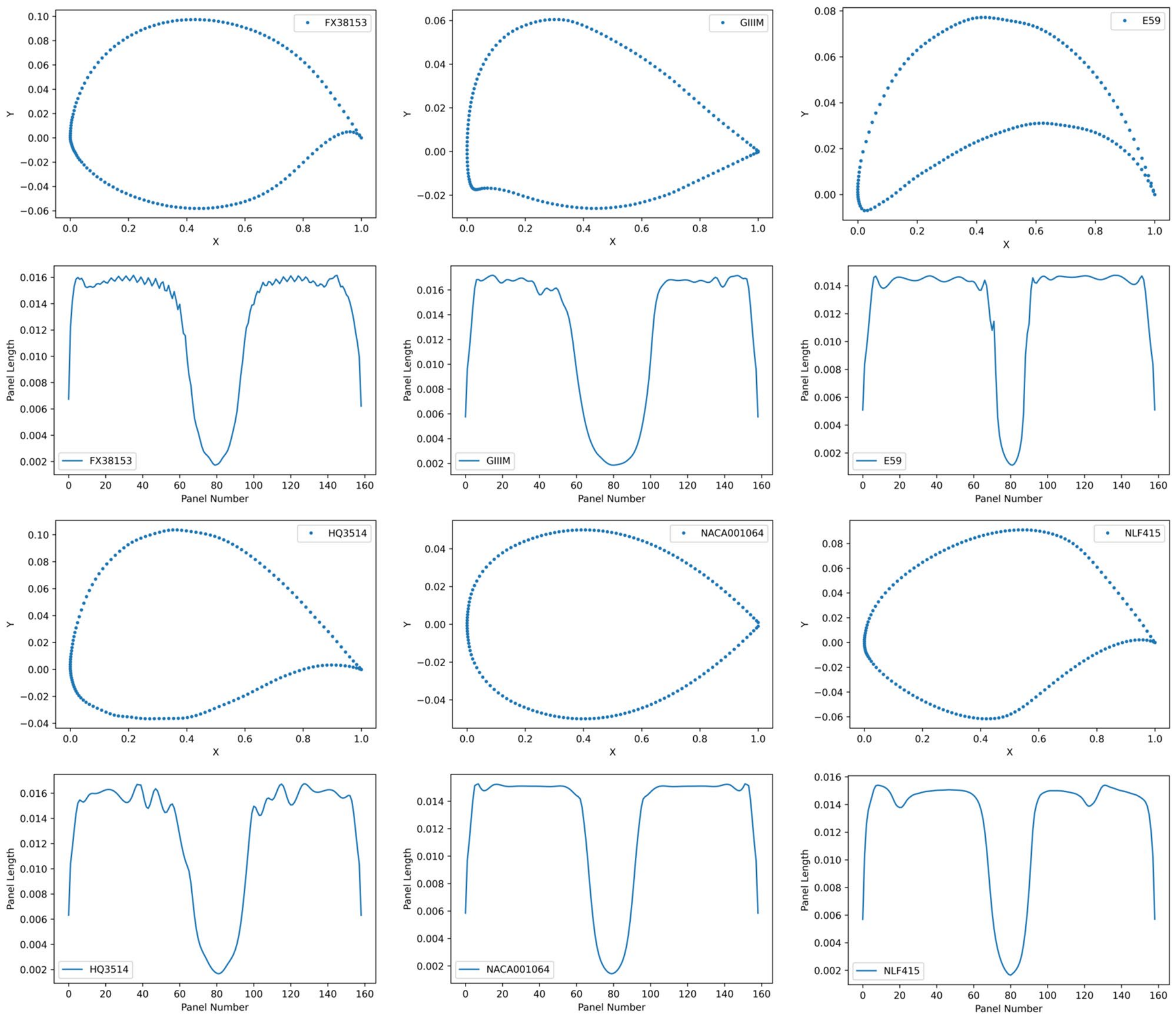

Figure 11. Randomly sampled airfoils and their panel length distribution.

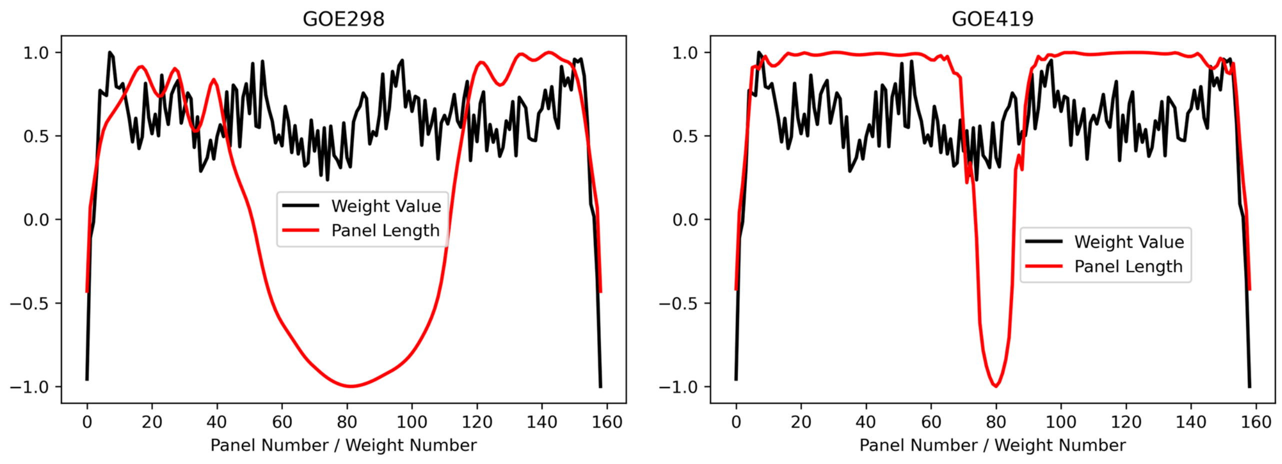

Figure 12. Comparison of learned weights and panel length distribution at the same scale.

Airfoil	Correlation	Error (%)
GOE298	0.16	1.22
GOE419	0.43	0.71
EH2012	0.32	0.93
HQ259B	0.30	0.97
NACA632615	0.27	1.03

Table 2. Correlation (between the learned weights and the panel length) of different airfoils and prediction error.

Received: 5 September 2021; Accepted: 31 March 2022
Published online: 26 April 2022

References

1. Hermann, J., Schätzle, Z. & Noé, F. Deep-neural-network solution of the electronic Schrödinger equation. *Nat. Chem.* **12**, 891–897 (2020).
2. Senior, A. W. *et al.* Improved protein structure prediction using potentials from deep learning. *Nature* **577**, 706–710 (2020).
3. Carleo, G. *et al.* Machine learning and the physical sciences. *Rev. Modern Phys.* **91**, 045002 (2019).
4. Brunton, S. L., Noack, B. R. & Koumoutsakos, P. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **52**, 477–508 (2020).
5. Duraisamy, K., Iaccarino, G. & Xiao, H. Turbulence modeling in the age of data. *Annu. Rev. Fluid Mech.* **51**, 357–377 (2019).
6. Jin, X., Cai, S., Li, H. & Karniadakis, G. E. Nsfnets (Navier–Stokes flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations. *J. Comput. Phys.* **426**, 109951 (2021).
7. Kochkov, D. *et al.* Machine learning-accelerated computational fluid dynamics. *Proc. Natl. Acad. Sci.*. **118**, e2101784118 (2021).
8. Li, Z. *et al.* Fourier neural operator for parametric partial differential equations. *arXiv preprint*. arXiv:2010.08895 (2020).
9. Li, Z. *et al.* Neural operator: Graph kernel network for partial differential equations. *arXiv preprint*. arXiv:2003.03485 (2020).
10. Erichson, N. B., Muehlebach, M. & Mahoney, M. W. Physics-informed autoencoders for lyapunov-stable fluid flow prediction. *arXiv preprint*. arXiv:1905.10866 (2019).
11. Wang, R., Kashinath, K., Mustafa, M., Albert, A. & Yu, R. Towards physics-informed deep learning for turbulent flow prediction. in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1457–1466 (2020).
12. Lusch, B., Kutz, J. N. & Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nat. Commun.* **9**, 1–10 (2018).
13. Sirignano, J. & Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **375**, 1339–1364 (2018).
14. Sun, Y., Zhang, L. & Schaeffer, H. Neupde: Neural network based ordinary and partial differential equations for modeling time-dependent data. in *Mathematical and Scientific Machine Learning*, 352–372 (PMLR, 2020).
15. Tang, H. *et al.* An exploratory study on machine learning to couple numerical solutions of partial differential equations. *Commun. Nonlinear Sci. Numer. Simulat.* **97**, 105729 (2021).
16. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Computat. Phys.* **378**, 686–707 (2019).
17. Pang, G., Lu, L. & Karniadakis, G. E. fpinns: Fractional physics-informed neural networks. *SIAM J. Sci. Comput.* **41**, A2603–A2626 (2019).
18. Yang, X., Zafar, S., Wang, J.-X. & Xiao, H. Predictive large-eddy-simulation wall modeling via physics-informed neural networks. *Phys. Rev. Fluids* **4**, 034602 (2019).
19. Wang, S., Teng, Y. & Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.* **43**, A3055–A3081 (2021).
20. Mao, Z., Jagtap, A. D. & Karniadakis, G. E. Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.* **360**, 112789 (2020).
21. Ding, M., Chen, Z., Du, T., Luo, P., Tenenbaum, J. & Gan, C. Dynamic visual reasoning by learning differentiable physics models from video and language. *Adv. Neural Inf. Process. Syst.* **34**, (2021).
22. Zhang, Y., Tiňo, P., Leonardis, A. & Tang, K. A survey on neural network interpretability. in *IEEE Transactions on Emerging Topics in Computational Intelligence* (2021).
23. Zhang, Q. & Zhu, S.-C. Visual interpretability for deep learning: A survey. *arXiv preprint*arXiv:1802.00614 *(2018).*
24. Zhang, Q., Wu, Y. N. & Zhu, S.-C. Interpretable convolutional neural networks. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* **8827–8836**, (2018).
25. Chakraborty, S. *et al.* Interpretability of deep learning models: A survey of results. in *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI)*, 1–6 (IEEE, 2017).
26. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
27. Davies, A. *et al.* Advancing mathematics by guiding human intuition with AI. *Nature* **600**, 70–74 (2021).
28. McGrath, T. *et al.* Acquisition of chess knowledge in alphazero. *arXiv preprint*. arXiv:2111.09259 (2021).
29. Peng, W., Zhang, Y. & Desmarais, M. Spatial convolution neural network for efficient prediction of aerodynamic coefficients. in *AIAA Scitech 2021 Forum*, 0277 (2021).
30. Selig, M. *UIUC airfoil data site* (Department of Aeronautical and Astronautical Engineering University of Illinois at Urbana-Champaign, 1996).
31. Drela, M. Xfoil: An analysis and design system for low Reynolds number airfoils. in *Low Reynolds Number Aerodynamics*, (ed. Thomas J. Mueller.) 1–12 (Springer, 1989). https://link.springer.com/book/10.1007/978-3-642-84010-4#editorsandaffiliations
32. Anderson, J. D. Jr. *Fundamentals of Aerodynamics* (Tata McGraw-Hill Education, 2010).
33. Zhang, Y., Sung, W. J. & Mavris, D. N. Application of convolutional neural network to predict airfoil lift coefficient. in *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 1903 (2018).
34. Franklin, J. The elements of statistical learning: Data mining, inference and prediction. *Math. Intell.* **27**, 83–85 (2005).
35. Xu, B., Wang, N., Chen, T. & Li, M. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint*. arXiv:1505.00853 (2015).

Acknowledgements
We acknowledge the computing resources powered by Google Colab, and by the CAAI-Huawei MindSpore Open Fund.

Author contributions
W.P. developed the idea for this study, conceived the code, designed experiments, and analyzed the data. Y.Z. and M.D. contributed to proof-of the-concept and refining the ideas, and discussion of results. W.P., Y.Z., E.L., M.D. interpreted the results. Y.Z. made conclusions, and organized the paper. M.D. and E.L. reviewed and edited the manuscript. All authors contributed to manuscript revision. All authors commented and approved the final manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to W.P.