# Fully Data-Driven Solving of Aerofoil-Wind Interactions in a Single Training of a Single Multi-Output DeepONet

**Sebastien Andre-Sloan**[1], **Dibyakanti Kumar**[2], **Anirbit Mukherjee**[1]

[1] *Department of Computer Science, The University of Manchester, UK*
[2] *Barclays, India*

## 1 Introduction

In recent years, deep learning has emerged as a competitive method for solving PDEs numerically. It has gained significant momentum, and 'AI for Science' Karniadakis et al. (2021) has emerged as a distinctive research direction. Data-driven methods for solving P.D.Es can broadly be classified into two kinds, **(1)** those which train a single neural net to solve a specific P.D.E and **(2)** operator methods – which train multiple nets in tandem to be able to solve in "one shot" a family of differential equations, with a fixed differential operator and different "source / forcing" functions. The operator net was initiated in Chen & Chen (1995) and its most popular deep neural version, the DeepONet, was introduced in Lu et al. (2021). An unsupervised form of this idea was introduced in Wang et al. (2021). We note that the operator methods are particularly useful in the supervised setup when the underlying physics is not known – as is the setup in this work – and state-of-the-art approaches of this type, can be seen in works like Raonić et al. (2023).

In this work we demonstrate that on a realistic task of predicting fluid dynamics, operator networks can get accuracies within one order of magnitude of other baselines, while only using 15% of the data used by the baselines.

**The AirfRANS Dataset (Bonnet et al., 2023)** The dataset consists of 1000 simulations created by their finite volume solver, OpenFOAM, solving the 2-dimensional incompressible Reynolds-Averaged Navier-Stokes Equations, under the assumption of steady-state flow, where the temperature $T$ and kinematic viscosity $\nu$ are kept constant. The components being solved for are $(u, v, p, \nu_T)$, where $(u, v)$ is the velocity, $p$ is the reduced pressure, $\nu_T$ is the kinematic turbulent viscosity.

Each simulation has a different airfoil geometry and inlet velocity. Each simulation is a point cloud (a list of points) of ~ 180000 points, each with 12 values, $(x,\ y,\ u_{in},\ v_{in},\ d,\ n_x,\ n_y,\ u,\ v,\ p,\ \nu_T, b)$ where **(i)** $(x, y)$ is the coordinate position of the point **(ii)** $(u_{in}, v_{in})$ is the inlet velocity, which is the same for every point in the cloud **(iii)** $d$ is the distance from the airfoil **(iv)** $(u, v)$ is OpenFOAM's predicted velocity **(v)** $p$ is OpenFOAM's predicted pressure divided by the specific mass and **(vi)** $\nu_T$ is OpenFOAM's predicted turbulent kinematic viscosity The other values, $(n_x, n_y, \beta)$ is the normal to the airfoil surface with a boolean attached that represents whether it is on the airfoil's surface or not, but we don't use these values.

Our DeepONet would be trained on 100 $(N)$ of these airfoils given in the "scarce training" task of this dataset.

**Related Works** Similar to Shukla et al. (2023), we choose 100 points on the surface of the airfoil (half of the points Shukla et al. (2023) used) as a representation of the airfoil geometry. But unlike Shukla et al. (2023), the angle of attack, which is given as the inlet velocity, varies for us. So, we append the inlet velocity to the 100 surface points and thus each input to our branch net is of the dimension 202, denoted by $\boldsymbol{b}_i$ for $1 \leq i \leq N$. We also train only one DeepONet for all the components of the labels, unlike in Shukla et al. (2023) where each component was predicted by a separate DeepONet. In Bonnet et al. (2023), the baseline MLP used had a training dataset of size $3.96 \times 10^8$ and trained in ~2h30min whereas our dataset had size $6.12 \times 10^7$, and trained in ~1h10min.

**Data Pre-processing** The training points for the trunk net, denoted by $\boldsymbol{t}_{i,j}$ for $1 \leq i \leq N$ and $1 \leq j \leq P$, are chosen by ensuring 70% of the points are at most 0.3 units away from the chord of the airfoil, as there

needs to be a focus on the region close to airfoil due to the rapid changes in that area of the domain. The last 30% points are found anywhere else in the domain. Our setup has 3000 total points ($P$) per each of the 100 ($N$) airfoils. The labels are denoted as $(u_{i,j}, v_{i,j}, p_{i,j}, (\nu_T)_{i,j})$ for $1 \le i \le N$ and $1 \le j \le P$. To balance the variation in magnitude across the different label components, we scale the labels and train the DeepONet on the scaled data. Let the mean and standard deviations of the labels over the whole training dataset be $(\phi_p, \phi_u, \phi_v, \phi_{\nu_T})$ and $(\sigma_p, \sigma_u, \sigma_v, \sigma_{\nu_T})$ respectively. Then the normalized labels used are $\bar{a}_{i,j} = \frac{a_{i,j} - \phi_a}{\sigma_a}$ for $a \in \{p, u, v, \nu_T\}$.

Each inputs-label pair is created with **(i)** Input : The 200 coordinates from the airfoil boundary, the 2 components of inlet velocity for the branch network, $\boldsymbol{b}_i$, and 2 coordinates for the position where the prediction is needed is the trunk network input, $\boldsymbol{t}_{i,j}$. **(ii)** Output : The 4 scaled OpenFOAM predictions, namely the velocity $(\bar{u}_{i,j}, \bar{v}_{i,j})$, pressure $\bar{p}_{i,j}$, and turbulent kinematic viscosity $(\bar{\nu}_T)_{i,j}$.

This gives a total of $3 \times 10^5$ training points $((N = 10^2) \times (P = 3 \times 10^3))$, each with 204 values for the input to the DeepONet and 4 values for the labels.

## 2 The Results from a Multi-Output DeepONet

Let the branch net be $\mathcal{N}_B$ and the trunk net be $\mathcal{N}_T$ where, $\mathcal{N}_B : \mathbb{R}^{202} \to \mathbb{R}^{4q}$,$\mathcal{N}_T : \mathbb{R}^2 \to \mathbb{R}^{4q}$. Both networks have a depth of 3 and a width of 100, and we choose $q = 100$. The DeepONet, $\tilde{\mathcal{G}}$ is the map, $\mathbb{R}^{202\times 2} \ni (\boldsymbol{b}, \boldsymbol{t}) \mapsto \tilde{\mathcal{G}}(\boldsymbol{b}, \boldsymbol{t}) := (\tilde{\mathcal{G}}_1(\boldsymbol{b}, \boldsymbol{t}), \tilde{\mathcal{G}}_2(\boldsymbol{b}, \boldsymbol{t}), \tilde{\mathcal{G}}_3(\boldsymbol{b}, \boldsymbol{t}), \tilde{\mathcal{G}}_4(\boldsymbol{b}, \boldsymbol{t})) \in \mathbb{R}^4$, where, $\tilde{\mathcal{G}}_i(\boldsymbol{b}, \boldsymbol{t}) = \sum_{k=(m-1)q}^{mq} (\mathcal{N}_B(\boldsymbol{b}))_k \cdot (\mathcal{N}_T(\boldsymbol{t}))_k, \forall\ m = 1, 2, 3, 4$

### 2.1 Loss Function

The loss function used to train the DeepONet $(\tilde{\mathcal{G}})$ is the MSE Loss on the above scaled training dataset with the points representing the airfoils and inlet-velocity $\{\boldsymbol{b}_i\}_{i=1}^{N}$, the trunk coordinates as $\{\boldsymbol{t}_{i,j}\}_{i=1,j=1}^{N,P}$ and the labels as $\{(u_{i,j}, v_{i,j}, p_{i,j}, (\nu_T)_{i,j})\}_{i=1,j=1}^{N,P}$. Then the loss can be written as $\mathcal{L}_{\text{train}}(\theta) = \frac{1}{NP} \sum_{i=1}^{N} \sum_{j=1}^{P} \left( \left|\bar{u}_{i,j} - \tilde{\mathcal{G}}_1(\boldsymbol{b}_i, \boldsymbol{t}_{i,j})\right|^2 + \left|\bar{v}_{i,j} - \tilde{\mathcal{G}}_2(\boldsymbol{b}_i, \boldsymbol{t}_{i,j})\right|^2 + \left|\bar{p}_{i,j} - \tilde{\mathcal{G}}_3(\boldsymbol{b}_i, \boldsymbol{t}_{i,j})\right|^2 + \left|(\bar{\nu}_T)_{i,j} - \tilde{\mathcal{G}}_4(\boldsymbol{b}_i, \boldsymbol{t}_{i,j})\right|^2 \right)$.

### 2.2 Results

The $\ell_2$-error averaged over scaled test data is shown in Table 1, and the plots of the prediction on an airfoil from the test data is given in Figure 1.

| | $\bar{u}$ | $\bar{v}$ | $\bar{p}$ | $\bar{\nu_T}$ |
|---|---|---|---|---|
| $\ell_2$-error | 0.2812 | 0.1423 | 0.1727 | 0.2452 |

Table 1: The above table shows the $\ell 2$-error between the DeepONet and the OpenFOAM predictions.

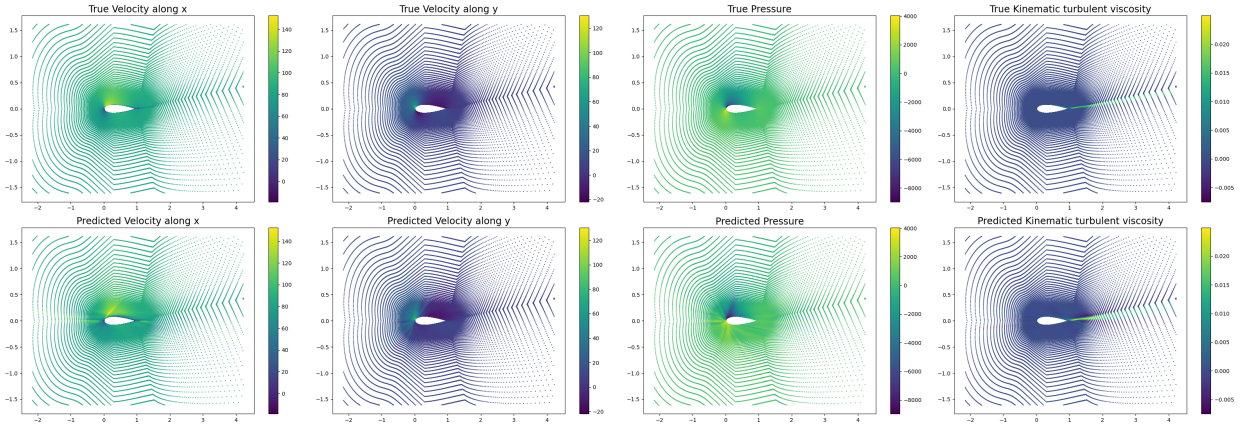


Figure 1: The top row of plots are the OpenFOAM simulations and the bottom row is the DeepONet predicted solutions.

## References


Florent Bonnet, Ahmed Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier-stokes solutions, 2023.

Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995. doi: 10.1109/72.392253.

Karniadakis, G.E., and Kevrekidis. Physics-informed machine learning, 2021.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, mar 2021. doi: 10.1038/s42256-021-00302-5.

Bogdan Raonić, Roberto Molinaro, Tobias Rohner, Siddhartha Mishra, and Emmanuel de Bezenac. Convolutional neural operators. *arXiv preprint arXiv:2302.01178*, 2023.

Khemraj Shukla, Vivek Oommen, Ahmad Peyvan, Michael Penwarden, Luis Bravo, Anindya Ghoshal, Robert M. Kirby, and George Em Karniadakis. Deep neural operators can serve as accurate surrogates for shape optimization: A case study for airfoils, 2023.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40):eabi8605, 2021. doi: 10.1126/sciadv.abi8605.