Towards Machine Learning for Acoustic Resonance Technology

Elisabetta Merico[1], Antonio Jimenez-Garcia[2]
1. Graduate Multiphysics Engineer, FT Technologies Ltd, Sunbury House, Sunbury-on-Thames, TW16 7DX, UK
2. Principal Research Engineer, FT Technologies Ltd, Sunbury House, Sunbury-on-Thames, TW16 7DX, UK

Abstract
FT wind sensors measure wind speed and direction using an acoustic field superimposed on a flow field in an acoustic resonator equipped with three piezoelectric transducers. The study proposed here focuses on the acoustic resonance technology of a simplified wind sensor geometry, and how it is affected by geometrical tolerances. In particular, it will explore the advantages of using machine learning algorithms to save computational costs without compromising the accuracy and reliability of the results. The outcome of this analysis is to optimise the number of points needed in the Design of Experiments (DoE) table, the structure of the Deep Neural Network (DNN), the learning rate used, and the duration of the DNN training. The results will focus on the comparison between the surrogate modelling and FE simulations in terms of accuracy, reliability, and computational time.

Keywords: Machine Learning, acoustic resonance technology, DNN hyperparameters optimisation, finite element method

Introduction
Machine learning techniques are rapidly advancing, enabling the modelling and solving of complex problems even without a complete understanding of the underlying physics or mathematics. Compared to traditional algorithms where the user must provide inputs and the rules to get an output, Artificial Intelligence (AI) algorithms exploit large datasets to identify the laws that relate the inputs to the outputs. In particular, a Deep Neural Network (DNN) is a type of AI model composed of multiple layers of neurons (or nodes) connected by weights that adjust during training. However, the performance of a DNN is highly sensitive to the choice of its parameters, such as learning rate, number of layers, number of neurons per layer, and duration of the training. The process of tuning these parameters, often referred to as hyperparameter optimisation, is crucial for maximizing the network's effectiveness.
This paper explains the process of building a DNN starting from a Finite Elements COMSOL Multiphysics® model of a simplified wind sensor that exploits an acoustic field to compute wind speed and direction. The first step is the setup of the Multiphysics model, followed by its validation. After that, a Design of Experiment (DoE) is conducted by exploiting the new functionality introduced with version 6.2 of COMSOL Multiphysics®. Finally, the table obtained is used to train a DNN, testing different settings to optimise the hyperparameters.

Computational Set Up
FT wind sensors work with an acoustic field superimposed on the flow field to compute wind speed and direction. To fully model these devices, the interactions between the acoustic and flow fields should be studied and understood, therefore accurate simulation models can take a long time to be set up and run. However, to simplify the model, only the acoustic field in the absence of flow will be considered for this test case. Even for a simple case as the one proposed in this study, one simulation across the frequency range of interest takes on average 4 minutes. If the influence of the tolerances on 10 geometrical features wants to be studied, choosing only 3 values for each of the parameters would require $3^{10} = 59{,}049$ simulations to test all the possible combinations. Given that each simulation takes 4 minutes, this would require 236,196 minutes which corresponds to half a year. Therefore the need to explore machine learning algorithms arises, to evaluate how they can cope with acoustic resonance problems to reduce the computational time without compromising the accuracy of the results.
The primary tool used for this study is COMSOL Multiphysics®, alongside the LiveLink for SolidWorks®. The simulation process involves several key steps:

- **Configuring FE Simulations:** establishing simulations with appropriate boundary conditions in the frequency domain
- **Validating the FE Model:** ensuring the accuracy and reliability of the FE model, through the comparison with experimental data
- **Creating the DoE Table:** exploiting COMSOL® v6.2 new functionality to create a large dataset to be used to train a DNN
- **Training the DNN:** using the data from the simulations to train the DNN
- **Testing the DNN:** evaluating the performance of the trained DNN to assess its effectiveness, and fine-tuning the hyperparameters

FE Model Setup

The simplified geometry of the wind sensor, illustrated in Figure 1, is imported in COMSOL Multiphysics® with the LiveLink for SolidWorks®. The computational domain is set up following an optimisation study outlined in reference [1], and it consists of a block that encompasses half of the resonator area and expands half wavelength outside of it, as shown in Figure 2.

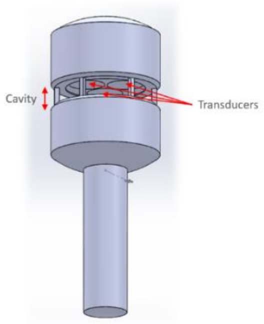


Figure 1: Wind sensor's simplified geometry

The entire computational domain is composed exclusively of air, with the sensor's walls treated as boundary conditions with an optimised absorption coefficient. The pressure acoustic physics is applied to the whole domain as this study will focus on the acoustic response of the resonator in the absence of flow in the frequency domain. The sensor's behaviour is replicated by treating the transmitting transducer (B following the nomenclature in Figure 2) as a piston oscillating with constant speed amplitude across the frequencies of interest and measuring the signal on transducer A.

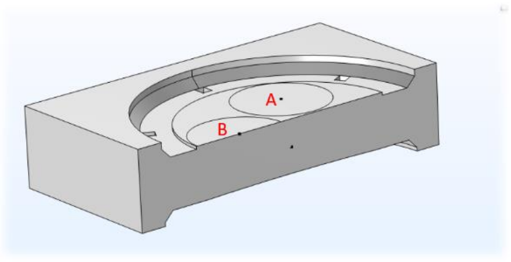
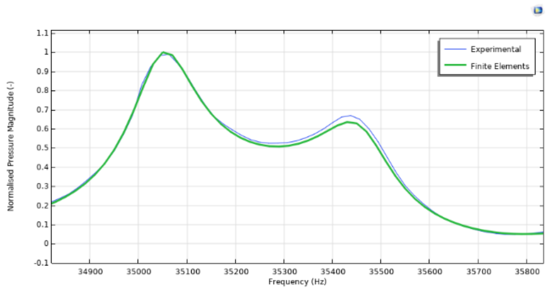


Figure 2: Computational domain encompassing half the resonator area and extending half wavelength outside of it

Validation of FE Model

The Finite Element model presented in the section above was validated against experimental data. In particular, the normalised pressure magnitude on the receiving transducer (A) was compared with the voltage signal received by the real transducer while testing, and the results are reported in The comparison was obtained by measuring the dimensions of a wind sensor on a coordinate measuring machine (CMM), and using the measured values to create an accurate model in COMSOL Multiphysics®.

Figure 3.



The comparison was obtained by measuring the dimensions of a wind sensor on a coordinate measuring machine (CMM), and using the measured values to create an accurate model in COMSOL Multiphysics®.

Figure 3: Normalised pressure response of the transducer A: comparison of Finite Element simulation with experimental results

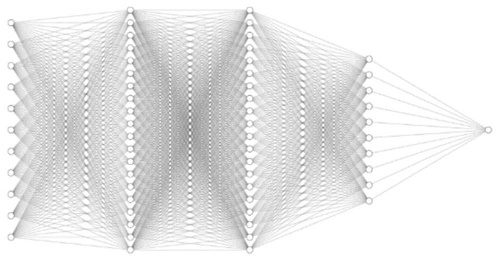
Design of Experiments (DoE)

After the model is validated, it is possible to proceed with building the Design of Experiments exploiting the surrogate model feature, introduced by COMSOL® v6.2 in November 2023. In order to do so, the geometry is parametrised in SolidWorks®, choosing 10 geometrical parameters, which are varied in their tolerance limits with a Gaussian distribution to build the DoE table. In addition to the geometrical parameters, also the frequency will be varied, but following a uniform distribution in the band of interest (33-38kHz). The 11 parameters are then varied following a Latin Hypercube Sampling (LHS) strategy, and the magnitude of the acoustic pressure on the receiving transducer, the output of the study, is computed for each set of input parameters. For this step, it is suggested to use 1000 points in the DoE for each variable considered, therefore in this case, at least 11000 points should be needed, but this number will be optimised to minimise the computational cost.

Training of the DNN

Once the Design of Experiments (DoE) table is created, it can be used to train a Deep Neural Network (DNN). This is done by adding a function to the global definition node and selecting the DNN option. This process enables the setup of the neural network structure, including the number of layers, the number of nodes per layer, and other hyperparameters.

Most of the different options that can be chosen in this node will be thoroughly explored to find the best combination of the hyperparameters that define the



neural network. To start with, a simple structure of the DNN was chosen with 3 hidden layers, as per the schematics below:

Figure 4: DNN starting structure with three hidden dense layers of 20, 20, and 10 nodes each

The other hyperparameters were only slightly changed from the default combination, leaving the Adam method, with α=1e-3 as the learning rate and no weight decay option, 512 batch size, the root-mean-squared error as the loss function, and the hyperbolic tangent as the activation function, but increasing the number of epochs to 4000 and the validation data fraction to 0.2. The random seed was left fixed at 0 to ensure the repeatability of the results.

Simulation Results

This section presents the main findings on how the hyperparameters affect the accuracy and reliability of DNN predictions. To evaluate the DNN, the Finite Element (FE) results from the validated model were compared with the predictions made by the DNN for the nominal values of the geometrical features and the ones obtained with the CMM scan.

The first factor to consider is the number of points needed to build the DoE table. This step is the longest of the entire process, so it is crucial to balance its computational time and the accuracy of the final results. In this case, the error is measured using the MSE (Mean Squared Error) between the DNN prediction and the FE simulation results. The comparison is led both for the nominal values of the acoustic resonance wind sensor and for the ones obtained with the CMM scan.

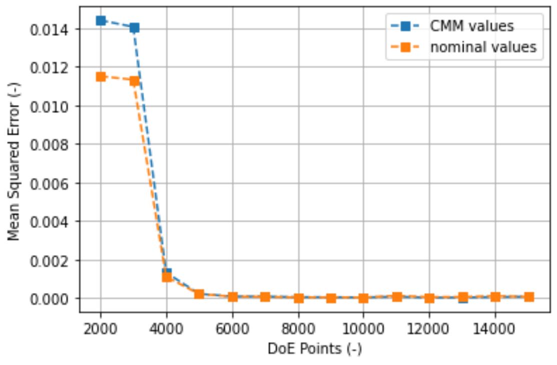


Figure 5: MSE comparing DNN predictions with FE simulation results for nominal values geometry and CMM scanned ones

The plot above shows that increasing the number of points in the DoE table over 6000 does not bring much value to the relative error between the Finite elements simulation results and the DNN prediction, both for nominal values parameters and CMM scanned values.

On the other hand, comparing the computational time needed to build the DoE tables, it takes on average 14.5s to compute each line of the table, therefore, some estimates of the time needed for different numbers of points are provided in the table below:

# points in DoE	Computational Time
2000	8hrs 3mins 20s
6000	24hrs 10min
11000	44hrs 18min 20s
15000	60hrs 25min

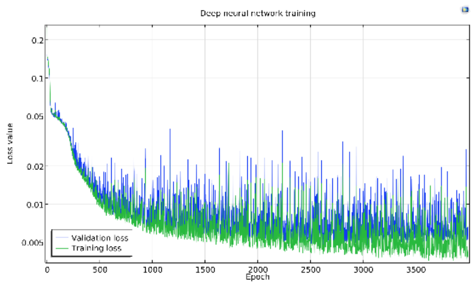
Table 1: Computational time related to different amounts of points in the DoE table, using an Intel® Core™ i9 CPU @ 3.60 GHz, with 8 cores and 128 GB of RAM

The second hyperparameter to fix is the learning rate. The learning rate controls the speed of training by determining the size of the steps taken during the gradient descent optimisation of the loss function. In gradient descent, the model parameters are updated iteratively to minimize the loss function $J(\theta)$, according to the following rule:

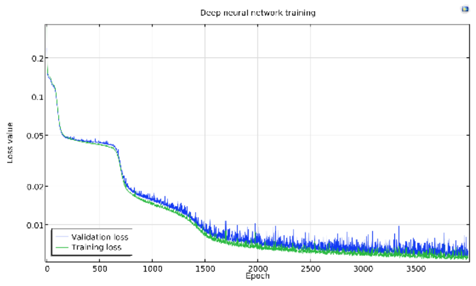
$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \alpha \boldsymbol{\nabla}_{\boldsymbol{\theta}} \boldsymbol{J}_n$$

A high learning rate can cause the loss function to fluctuate, potentially leading to divergence, whereas a too-small learning rate may result in the model getting stuck in a local minimum. Therefore, for each specific case, it is vital to find a good compromise between these two corner cases. Trying different options for the case study proposed, the following convergence plots are obtained:

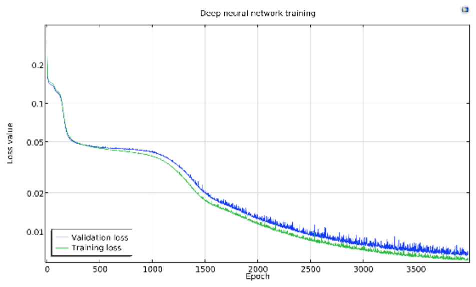
(a)



(b)



(c)



(d)

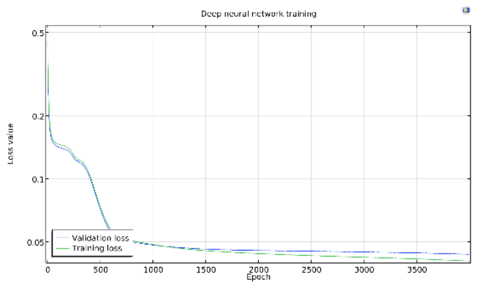


Figure 6: Convergence plots varying the learning rate α, being (a) 5x10^{-3}, (b) 1x10^{-3}, (c) 5x10^{-4}, and (d) 1x10^{-4}

The plots in Figure 6 clearly show that if α is set to 5*x*10^{-3}, the loss function exhibits multiple spikes which is highly undesirable. Decreasing it to 1*x*10^{-3} improves the situation, even though the descent is still not smooth. Lowering it further to 1*x*10^{-4} completely removes the oscillations but reveals that the loss function is stuck in a local minimum as the error plateaus over many epochs.
Setting α to 5*x*10^{-4} proved to be a good compromise as it almost eliminates the oscillations, avoiding the divergence of the solution. However, it slowed down the training, as at the end of the 4000 epochs, the loss function is still decreasing, indicating that further improvement is possible. This leads to the next hyperparameter to be optimised which is the number of epochs needed for the DNN training.
The number of epochs does not greatly affect the training time of the neural network, as in this simple case it is in the order of a minute, which is negligible compared to the time needed for the DoE; therefore, this analysis will be based only on the effects that the number of epochs has on the accuracy of the DNN predictions, without focusing on the time required for the training.
For this step, the learning rate was set to 5*x*10^{-4} and the full dataset with 15000 points in the DoE was selected, as having more points allows to better understand the other parameters' influence. With these settings, the following convergence plot is obtained for the training of the DNN:

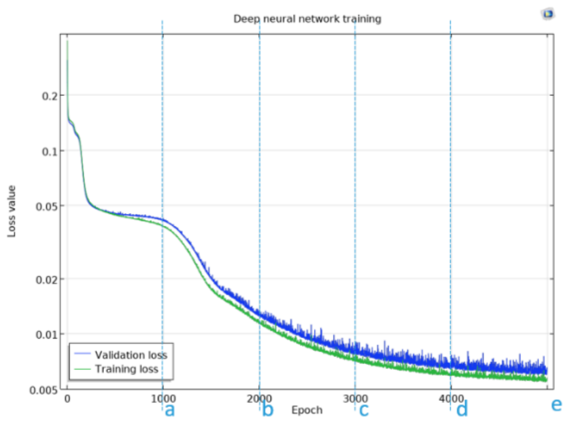


Figure 7: Convergence plot for the training of a DNN using α=5e-4, 15000 points in the DoE and the layer structure in Figure 4

Stopping the training at the points (*a*, *b*, *c*, *d*, *e*) indicated in Figure 7, the predictions in the following are obtained when the CMM scan values are used to evaluate the DNN function:

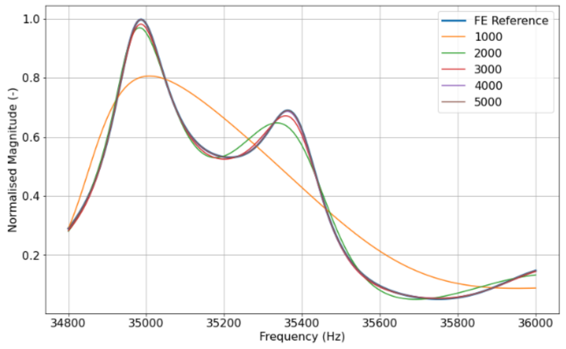


Figure 8: Comparison of the DNN prediction when it is trained on the same dataset, but for a different number of epochs

The plot in Figure 8 proves that training the DNN for 1000 epochs is definitely not enough to reach a good accuracy of the results, as the DNN is able to pick up the first peak resonance but the shape is completely different. Keeping training for an additional 1000 epochs improves the situation leading the DNN to be able to predict also the second peak, following the finite elements simulation results. Prolonging the training brings other smaller improvements which can be only qualitatively assessed from the comparison with the curve obtained with FE simulation.
As previously, to quantitatively judge the accuracy of the DNN predictions in comparison with the finite elements simulation results, the mean squared error is used as a metric:

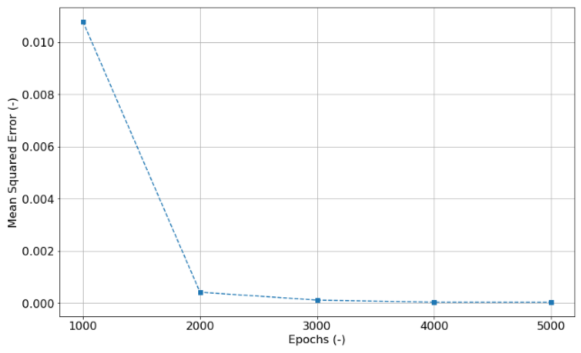


Figure 9: Mean squared error decreases with the number of epochs of the training

Figure 9 confirms the observation obtained from Figure 8; the error significantly decreases when the DNN is trained for 2000 epochs instead of 1000, and then slowly decays, with only marginal improvements after 4000 epochs. Although Figure 6 and Figure 7 might suggest that the training is not completed in 4000 epochs, the last two plots show that it is not necessary to keep training to decrease the training and validation errors after 4000 epochs, as it is already sufficiently small for this application.

The layer structure chosen at the beginning proved to be a good one, as a negligible error is recorded when the DNN predictions are compared with the finite elements simulation results. However, it is interesting to study how the shape of the network can influence the accuracy of the predictions and the computational cost of the training.
For this step of the study, the learning rate and the number of epochs to train for were set as the optimums just found, $5x10^{-4}$ and 4000 respectively, and only the structure of the layers was modified. As there are infinite possible combinations of the layers, the research was narrowed down to the 3 cases:

- 5 neurons per layer
- 10 neurons per layer
- 20 neurons per layer

For each of these cases, the number of hidden layers was gradually increased, starting from a single hidden layer.

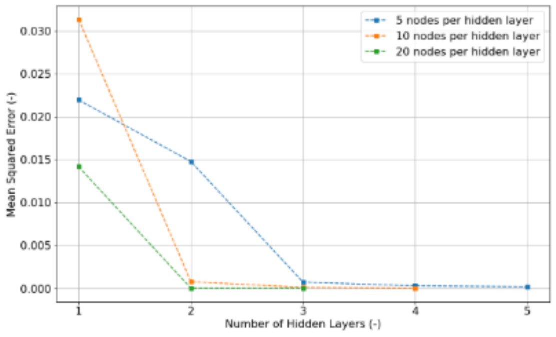


Figure 10: MSE comparing the DNN predictions to the FE results with different layers configurations

In Figure 10, MSE was used to determine the accuracy of the DNN predictions and compare them against the FE simulations using geometrical values obtained from the CMM scans. The plot clearly shows, as expected, that a higher number of nodes per layer and a higher number of layers lead to better predictions, as it allows the network to account for more interactions between the inputs and more non-linearities in the relationship inputs-output. Comparing the acoustic pressure shape in the frequency domain predicted in the three cases, for the highest number of layers tested, the plot in Figure 11 is obtained.

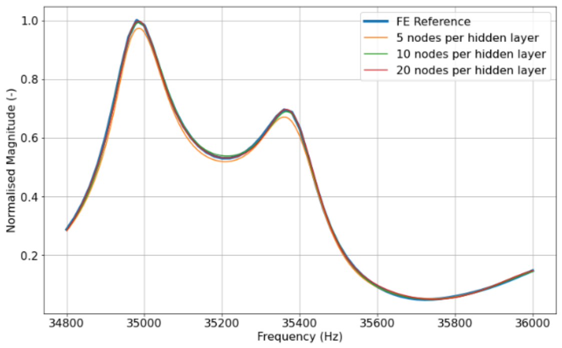


Figure 11: Comparison of DNN predictions when it is built with different numbers of nodes per layer

It can be observed that although the error appears almost zero in Figure 10, using 5 neurons per layer with 5 layers is not enough to accurately represent the problem. In contrast, even with just two hidden layers of 20 neurons each, the predictions are significantly improved. This can be explained by the fact that when input parameters interact strongly, having multiple layers with fewer neurons can effectively capture these complex relationships. However, in this case, where the effects of the geometrical parameters on the final output are less correlated, a shallow, wider network provides a better representation.
On the other hand, increasing the number of neurons to create a wider network is usually disadvantageous, as it significantly increases the number of weights that need to be computed compared to simply adding more layers, as illustrated below:

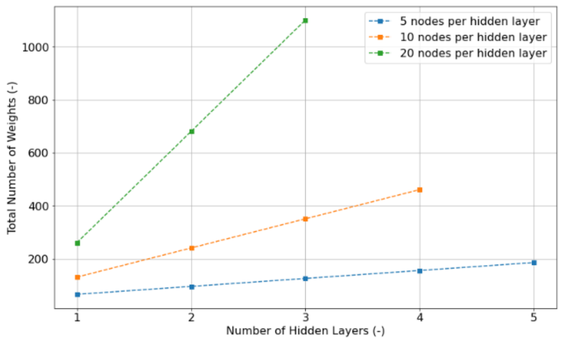


Figure 12: Total number of weights in the neural network for an increasing number of layers with different amounts of nodes per layer

In this case, it is evident that increasing the number of neurons from 5 to 20 per layer leads to a considerable rise in the number of weights as the number of layers increases. However, as mentioned above, the training of this neural network takes a relatively short time compared to the rest of the process therefore also the increase of the weights in the 20-nodes-per-layer case is not considerable in terms of computational time.
Lastly, adding the weight decay was tested to see if it could bring any improvement to the training. It consists of limiting the growth of the weights following the formula:

$$J'(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda \sum_i \boldsymbol{\theta}_i^2$$

This technique is called a regularisation technique and helps prevent the overfitting of the training dataset. However, this problem was not encountered in this case, therefore using the weight decay only proved to worsen the situation.
Finally, the only hyperparameters left, the validation fraction and the batch size, were not modified as they are greatly correlated with the number of points used for training, therefore it was chosen not to change too many interconnected parameters together, to be able to uncover the effect of each one of them.

Conclusions

This study investigated the impact of various hyperparameters on the accuracy and reliability of DNN predictions for modelling acoustic resonance wind sensors. As explained, to accurately model acoustic resonance wind sensors, the interactions between the acoustic and flow fields should be studied and understood, therefore exploiting ML algorithms can bring great advantages in terms of computational time.

The results demonstrated that by using DNNs, a tool can be developed to analyse the effect of tolerances on acoustic resonance wind sensor geometry in just over 24 hours, compared to the six months required to capture these interactions using FE simulations.

The study also examined the influence of individual hyperparameters on DNN predictions. It was determined that 6000 points in the Design of Experiments (DoE) table are sufficient to develop an accurate network that maps the 11 inputs to the output. Increasing the dataset size would lead to higher computational costs without substantial gains in prediction accuracy.

In terms of training configuration, the optimal learning rate was found to be $5x10^{-4}$, as it effectively balanced reducing error without causing the loss function to oscillate or diverge. The study showed that training for 2000 epochs provided a significant reduction in error, while extending training beyond 4000 epochs resulted in only marginal improvements. Hence, 4000 epochs were identified as sufficient for this application.

Finally, the structure of the DNN was also optimised. It was observed that a configuration with 20 neurons per layer and a shallow network architecture provided better predictions than using just 5 neurons per layer, with a greater network depth. Increasing the number of layers or neurons increases the computational complexity due to the larger number of weights, but a shallow, wider network was more effective for this problem, where the input parameters had less correlated effects on the output.

In conclusion, this study highlights the importance of carefully selecting hyperparameters, including learning rate, number of epochs, and network structure, to optimise the performance and efficiency of DNNs in modelling complex physical phenomena like those in acoustic resonance wind sensors. Unfortunately, hyperparameter tuning must be done on a case-by-case basis, as there is no one-size-fits-all combination that works for every problem. However, the impact of each hyperparameter highlighted in this study is likely applicable to various other problems, lending a level of generality to the conclusions drawn.

References

[1] A. Jimenez-Garcia and G. C. Diwan, "A comparison of acoustic solvers for FT ultrasonic wind," in *54th Spanish Congress on Acoustics*, Cuenca, 2023.