Aerodynamic State Estimation from Sparse, Real-world Pressure Data in Highly Unsteady Flows Using Multilayer Perceptron

Abstract

Accurate modeling of aerodynamic loads is an important guarantee for safe flight. At present, it is of great difficult to predict the aerodynamic load in the high Reynolds number gust environment. Due to the complex flow separation mechanism, the application of mathematical modeling is limited. The Multi-Layer Perceptron (MLP) provides a new method for modeling nonlinear systems. This paper explores the feasibility of MLP for aerodynamic load prediction in gust environment, and uses real-time noisy sparse surface pressure measurements as input. The dynamic response of the long pressure tubes is studied and a Bessel filter is used to smooth such oscillation. The prediction is analyzed, and the performance and limitations of the MLP model are discussed. The results demonstrated that the MLP model can accurately predict the loads in the middle of the gust envelope, but with poor prediction of the conditions on the boundary of the gust envelope.

1 Introduction

Flow separations at high angles of attack during gusty wind conditions create some of the most challenging flight environments for lightweight aerial vehicles. For example, the leading edge vortex (LEV) that forms due to dynamic flow separation leads to a rapid increase in lift followed by a sudden drop once the LEV sheds [1–3]. Therefor, an accurate aerodynamic-load prediction, and would allow for active flow control to achieve stable flight even in highly unsteady and complex flow environments.

Traditional aerodynamic load estimation research is primarily based on classical theories such as potential flow theory, but the oversimplification of assumptions and the empirical corrections severely limited their practical applications. Some research focused on modeling steady-state 3D flows using vortex lift theory combined with the potential flow theory [4, 5]. Hemati et al. [6] constructed a low-order model and successfully predicted the aerodynamic force of a two-dimensional airfoil, but these works have mainly focused on two-dimensional, low Reynolds number (Re) flows. Compared with two-dimensional, steady flow, three-dimensional, unsteady flow is more universal, and its modeling is more realistic. Unlike steady flow, unsteady-state three-dimensional flow is accompanied by strong non-linearity created dynamic flow separation.

Animals make full use of their own limited sensors (such as the lateral line of fish and wing tip features, see Fig. 1)to improve the efficiency of predation or avoid natural enemies, and to collect feedback from the environment , combine with experience, estimate and control the current hydrodynamic state in time to improve survival probability. These insights demonstrate the possibility to reconstruct fluid loads with only a limited number of sensors.

Data-driven approach has proven to be an effective option and is being continuously improved, such as complex networks [7–9], transition networks [10–13], and Machine Learning (ML). Due to its great advantages in dealing with nonlinear and chaotic problems [14], ML methods are increasingly being used in fluid problems, including supervised learning aimed at fitting predictions (such as flow field prediction [15]) , and classification-based unsupervised learning (e.g. distinguishing turbulent from non-turbulent flow [16]). Among many machine learning models, MLP [17], Random Forest (RF) [18], Support Vector Regression (SVR) [19] and Convolutional neural network (CNN) [20], are among the most successfully applied models in fluid mechanics. Fukami et al. [14] evaluated these machine learning

algorithms for representative regression problems. Their work demonstrates that machine learning-based regression of aerodynamic loads is feasible with limited sensor data collection.

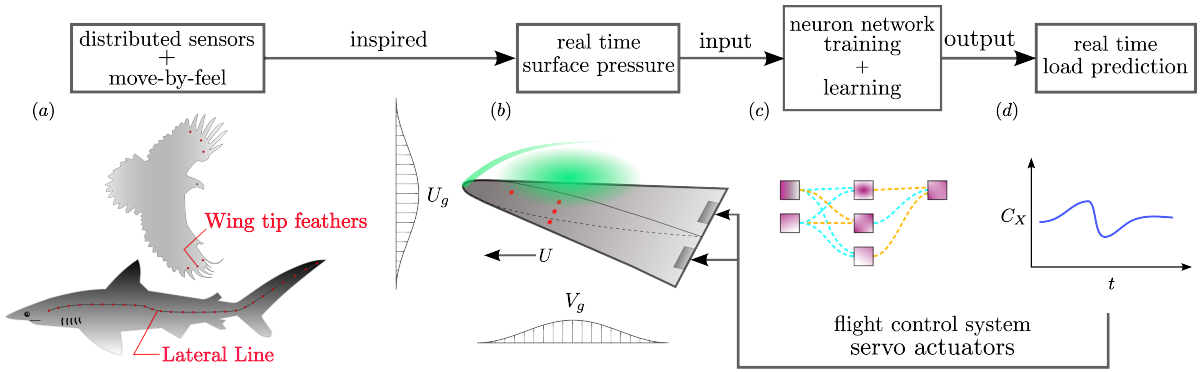


Figure 1: Schematic of sensors on bird and fish.

MLP, as an analogy to biological neural networks, has been successfully applied in fluid mechanics. Dennis and Robert [21] explored MLP estimation for aerodynamic coefficients. Faller and Schreck [22, 23] used MLP to predict unsteady pressure over a pitching wing, and explored the applications of MLP in aeronautics. Ling et al. [24] trained deep neural networks for turbulence modelling. Lui and Wolf [25] constructed a reduced order model using deep neural networks. Yu and Hesthaven [15] successfully reconstructed the flow field using a neural network. These successful applications demonstrate the ability of MLPs to capture flow field information. However, current works mainly focus on steady-state flows with low Reynolds numbers, and the data required for model training mainly comes from computational fluid dynamics (CFD) simulations. Numerical studies on unsteady flows with high Reynolds numbers mainly rely on the Reynolds averaged Navier-Stokes (RANS) model. Nevertheless, due to the introduction of time averaging in RANS, some transient information will be lost. For unsteady flows with high Reynolds numbers from the real world, the performance of MLPs has rarely been reported. Therefore, in this paper, we constructed a MLP model and analyzed its applicability to high Re unsteady flows using data collected from the real world.

Feature selection is a critical first step towards accurate aerodynamic load prediction ML model. Since the aerodynamic force depends on the motion history [26], it can be expressed as a nonlinear function of the angle of attack, side slip angle, etc. Chen et al. [26] took angle of attack and angle of sideslip et al. as the input of the ML mode. Wang et al. [27] took angle of attack and its first and second derivatives, reduced frequency, sideslip angle, and elevator deflection angle as the input variables, in which k is the reduced frequency; δ_e is elevator deflection angle. However,a major shorting of such features is the fact that it is difficult to measure in a realistic setting and impossible to be known in advance during a gust. In gust environment where the angle of attack changes dominated by aerodynamic forces, there is an inherent delay due to inertia of the system [28, 29]; the angle of attack cannot respond to the impact of the gust in time, and even after the gust passes, the aircraft may still keep its nose up for some time [30]; if it encounters head-on gust , the angle of attack remains constant [31]. The existence of this delay can adversely affect the flight control system. If not corrected, predicting aerodynamic forces with structural parameters as input variables will increase the error. Therefore, in the gust environment, the angle is not suitable as an input feature for machine learning. Using a distributed sensor array to acquire surface flow field data in real time can significantly reduce delays [32]. The pressure sensor is one of the more commonly used [33–35]. Others such as hot film [36, 37] and artificial hair sensors have also been successfully applied [32]. Most of these works are performed in the context of low Reynolds number flows. Using only surface pressure measurements as input features, Hou, et al. [38] accurately estimated the leading-edge suction parameters of a pitching plate by machine learning model. Wood et al. [39] used pressure data to successfully predict angle of attach, and so on. These indicate that surface pressure can reflect the motion state of the aircraft. Burelle et al. [40] built a linear regression model to approximate aerodynamic loads of a non-slender delta wing during axial and vertical gusts, by only surface pressure measurements; Iacobello et al. [13] built a transition network to predict aerodynamic loads around an accelerating elliptical plate at various steady states using only two surface pressure sensors.

Recently, He et al. [41] constructed a nonlinear dynamics sparse identification algorithm, using spatially distributed pressure sensor measurements as input, to estimate the aerodynamic loads on a delta wing under tranverse gusts in real time. Therefore, in this study, we use surface pressure from real world to predict aerodynamic loads in highly separated and unsteady flow at a high Reynolds number in gust environment, by constructing machine learning models.

This paper aims to evaluate the learning ability of MLP and its robustness to highly separated and unstable gusts at high Reynolds numbers. We designed a series of experiments to obtain real-world, sparse surface pressure responses of a non-slender delta wing during challenging gust scenario to evaluate the performance of MLP based load estimation model.

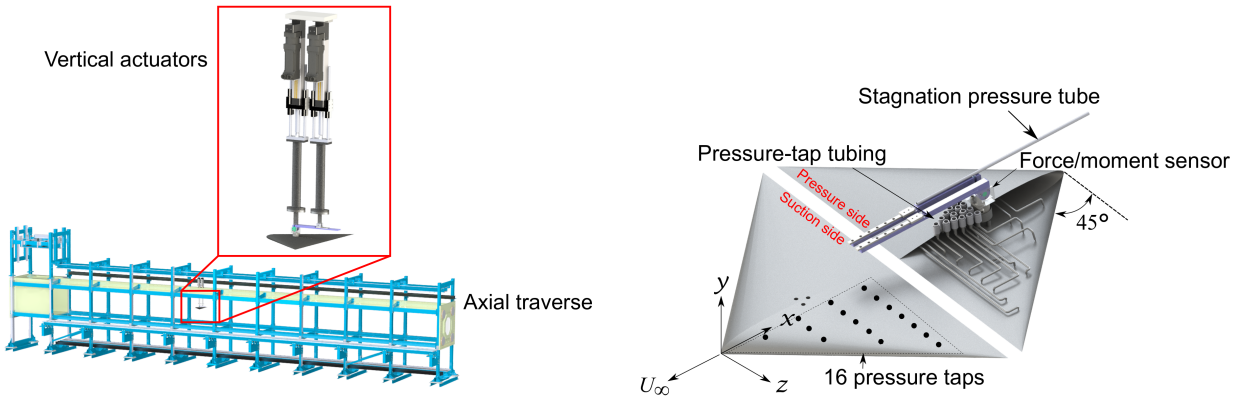
Experimental setups and flow conditions are described in Sec. 2. In Section 3, the MLP regression model and the data-driven procedure for training and prediction are introduced. Section 4 presents the prediction results of the MLP regression model for aerodynamic loads. Section 5 concludes the remarks.

2 Experimental Methodology

Large separation flows around a delta wing under various gust conditions are conducted in this research. Two types of gusts are combined in synchrony: one is axial gust and the other is vertical gust. The axial gusts are realized by a towing tank, and two identical linear actuators work together to achieve pitch motion to simulate vertical gusts. In both experiments, the surface sparse pressure was collected by the pressure sensor as input for machine learning; the aerodynamic loads would be reconstructed as output.

The model used in this research was a non-slender delta-wing model with a NACA 0012 cross-section. The sweep angle of the wing is 45° and the center span chord is $c = 0.3m$, as depicted in Fig. 2b . The model was selective laser sintering (SLS) printed in nylon plastic material, allowing internal piping to be connected to pressure fittings. The model's exterior was treated with cyanoacrylate to seal the porous material.

All the experiments were conducted in the $15m$ long optical towing tank facility at Queen's University as shown in Fig. 2a . The tank with a cross-section of $1 \times 1m^2$ has a ceiling to minimize the impact of the free surface, allowing only one opening large enough for the sting of the motorized traverse. The speed of the steady state (i.e., the speed before gusts) was $U_\infty = 1m/s$, resulting a centerline chord based $Re = 300000$. Two linear Parker actuators are combined together to control the pitching angle of the model, as shown in the partial enlarged view in Fig. 2a, with in-line accuracy $\pm 0.03mm$. The model was mounted on the left actuator (the first one in the moving direction) through the force sensor, while a linear bearing and rail were connected to the other actuator, which can slide to perform pitching motion. These two actuators can move simultaneously for plunging, or one can be fixed while the other moves up and down for pitching, reaching a pitching range $\pm 40^\circ$ if the initial angle is 0°. In this research, only pitching type and $\pm 30^\circ$ were used.



(a) Schematic of the tank with the delta wing model and the linear actuator system.

(b) Delta wing model with force sensor and pressure taps.

Figure 2: Experiment facilities.

2.1 Gust kinematics

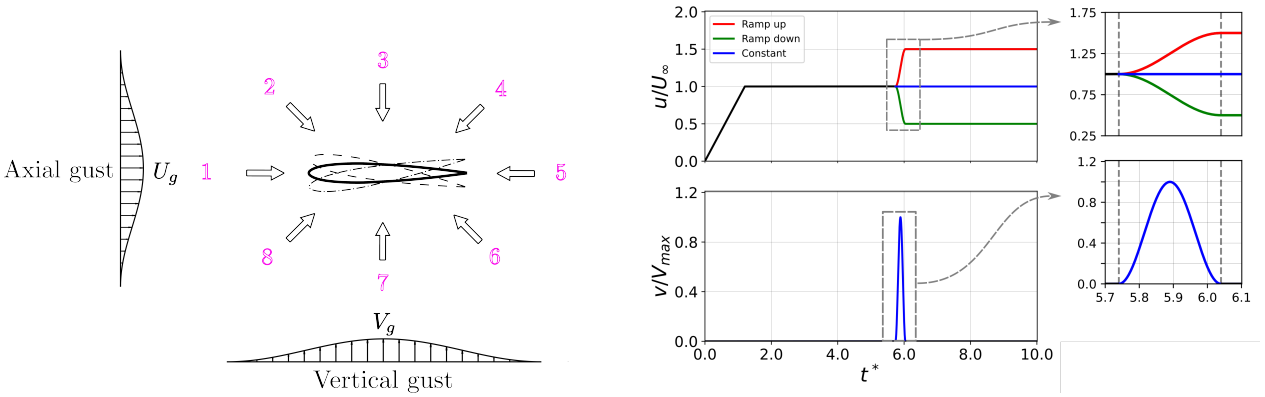
Gusts in 8 directions were conducted to simulate nature wind. To achieve this goal, the model was accelerated in both axial and vertical directions. Figure. 3a shows the schematic of the gusts and the directions, while Fig. 3b depicts the kinematics in detail. Before the experiment starts, the delta wing is set to various initial angles of attacks (AoA): $\alpha_{start} = \{10°, 20°, 30°\}$ by the two actuators. In the axial direction, the model is accelerated by the towing motor from standstill to $U_\infty = 1m/s$ over 2 chord lengths and then keeps over 15 chord lengths to achieve a steady state before gust occurs, while during this period, the actuators stay still. Then 3 types of motion occur in axial direction: ramp up, constant, and ramp down, while the actuator pitches simultaneously, activated by a photoelectric sensor when the towing motor passes. Equation 1

$$u_g = \begin{cases} \frac{\omega_0^u}{2}(1 - cos(\frac{2\pi t^*}{T^u}) + 1) & ramp\ up \\ 1 & constant \\ -[\frac{\omega_0^u}{2}(1 - cos(\frac{2\pi t^*}{T^u}) + 1] & ramp\ down \end{cases} \tag{1}$$

and Equ. 4

$$v_g = \begin{cases} 0 & t^* \leq 0 \\ \frac{\omega_0^v}{2}(1 - cos(\frac{2\pi t^*}{T^v}) & 0 < t^* \leq T^v \\ 0 & t^* > T^v \end{cases} \tag{2}$$

depict such motions, where ω_0^u and ω_0^v are the amplitude of the axial and vertical gusts respectively, and T^u, T^v is the period of the gusts, whereas $T^v = T^u/2 = 0.3s$. t^* is the dimensionless time, $t^* = t/T^v$, where $t^* = 0$ represents the start of the gusts. There are various AoAs after the gust: $\alpha_{end} = \{-10°, 0°, 10°, 20°, 30°\}$, and axial velocity $u_{end} = \{0.5m/s, 1m/s, 1.5m/s\}$, yielding a range of $Re = \{150000, 300000, 450000\}$. The whole test matrix contains 32 cases which can be found in Table. 1.



(a) Schematic diagram of gust in 8 directions in space, the directions are indicated by red numbers, V_g represents the gust in the vertical direction, U_∞ represents the incoming flow velocity in the far front.

(b) The gust velocity profiles of axial gust (upper) and vertical gust (lower). For the axial motion, there are three velocity profiles, one for ramp up motion (in red), one for ramp down (in green), and one for constant (in blue). For vertical motion, there is one velocity profile, but with different amplitude.

Figure 3: Gust kinematics in detail.

Table 1: Test matrix in detail

Case	α_{start}	α_{end}	$u_{end}(m/s)$	Gust direction	Case	α_{start}	α_{end}	$u_{end}(m/s)$	Gust direction
1	10	0	1	7	17	20	30	1.5	2
2	10	-10	1	7	18	30	30	1.5	1
3	10	20	1	3	19	30	20	1.5	8
4	10	30	1	3	20	30	10	1.5	8
5	20	10	1	7	21	10	10	0.5	5
6	20	0	1	7	22	10	0	0.5	6
7	20	30	1	3	23	10	-10	0.5	6
8	30	20	1	7	24	10	20	0.5	4
9	30	10	1	7	25	10	30	0.5	4
10	10	10	1.5	1	26	20	20	0.5	5
11	10	0	1.5	8	27	20	10	0.5	6
12	10	-10	1.5	8	28	20	0	0.5	6
13	10	20	1.5	2	29	20	30	0.5	4
14	20	20	1.5	1	30	30	30	0.5	5
15	20	10	1.5	8	31	30	20	0.5	6
16	20	0	1.5	8	32	30	10	0.5	6

2.2 Pressure and Force Sensors

An ATI Nano 25 six-axis force/moment sensor was used to measure the aerodynamic loads, which was mounted at the center chord, along the center span of the model. The load sensor had peak resolutions of $0.125N$ and $0.0015Nm$ for force and torque components, respectively.

The sparse pressure measurements over the suction side of the delta wing are conducted by 16 pressure taps. There are sixteen 3.175 mm ID passages that are printed directly into the wing, and each passage has a tap with $d = 1.558mm$ and $3.175mm$ in depth. The tap positions can be found in Fig. 4. Note that, there is one tap between $p10$ and $p11$ that did not use, instead, there was one tap $p0$ directly ahead of the model, which was used to gather stagnation point

pressure instead of all pressure data coming from the model surface. The pressure was collected using 8 Omega PX419 absolute vacuum transducers which have response times of about 0.001 s, at accuracy levels of up to 0.08%.

The aerodynamic loads and pressure were sampled over 1000Hz. The time-resolved measurements were averaged over 10 runs with 8 minus settling time between each run. No filter was applied to the loads, while a Bessel filter was used to mitigate the dynamic response of the pressure tubes (which will analyze in the next subsection), but pressure without filtering was still used for comparison to analyze the ability of MLP to deal with noise and tube dynamic response.

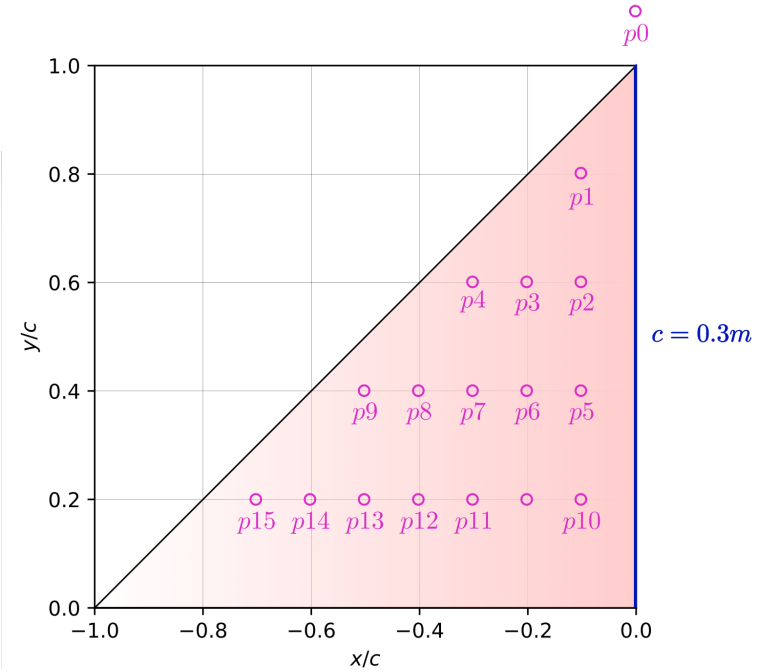


Figure 4: The taps location on the delta wing. The chord (marked in blue) length $c = 0.3m$.

Dynamic pressure ($q_\infty = 1/2\rho U_\infty^2$) was used to normalize the pressure (P) and loads (lift L, drag D) of the gust response, which was:

$$C_N = \frac{N}{q_\infty} \tag{3}$$

$$C_P = \frac{P - P_\infty}{q_\infty} \tag{4}$$

where N represents L and D and P_∞ is the static pressure in freestream.

2.3 Preliminary Results

All 32 cases were conducted in the towing tank, as described in the previous subsections. Force and pressure data were collected at the beginning of the motion, while the flow around the model kept steady most of the time before gusts occurred, which was not what we are interested in. So the data was trimmed and only the data around the gust was kept. As described in the previous subsection, $t^* = 0$ represents the start of the gust. We intercepted 500 sampling points forward from the gust, 2000 sampling points backward, a total of 2500 sampling points, 2.5s, or 8.3 ($2.5s/0.3s = 8.3$, 0.3s is the gust period) cycles of data, to analyze and use as the input feature of the MLP model.

Here, 4 cases results are listed to show the response of gusts, where the data are just ensemble-averaged, no filter is used for both force and pressure, and they are $case = \{13, 20, 25, 28\}$, which are actually the final test cases. Section 3 will illustrate the train and test cases in detail. Figure 5 shows the gust loads (C_L and C_D) and the pressure maps of these 4 cases. A strong correlation between C_L and C_D is observed in the gust load plots since the period of the gust is small, but there is a small, but clear phase lag for *case*_13, with an $\alpha_{start} = 10^\circ$ and $\alpha_{end} = 10^\circ$, causing a $\Delta\alpha = 10^\circ$. This phase lag disappears in all other 3 cases which have the same $\Delta\alpha = 20^\circ$, indicating that the gusts are much stronger than *case*_13. A big vibration is observed in both *case*_25 and *case*_28 after the gust in $3 < t^* < 4$, which is caused by the mechanical vibration of the towing motor. We do not use filters to smooth this vibration as we want to examine the handling of outliers by MLP models. This will discuss in Section 4.

The pressure maps are obtained through interpolation of the 15 pressure taps. In gust load plots, 4 times are shown in dashed lines: $t_1^* = -0.6$ before gust occurs, and the first pressure map (the top-left of each case) shows that the pressure is nearly constant at this time stamp, indicating a large-region of separation; $t_2^* = 1$ is the end of the gust, and in the second pressure map (the top-right of each case), the separation is observed in only *case*_13 with a clear reattached line, while in other cases, weak pressure gradient dominates. $t_3^* = 1.5$ is a short time after the gust. A stronger detached flow occurs in case_13, while the other cases are still with weak pressure gradients except case_20. In the last $t_4^* = 6$ is far from the gust. and the pressure of all cases gets recovered, in which *case*_13 and *case*_20 have flow reattached.

It must be mentioned that these 4 cases, as well as the training cases in the next section, are all randomly selected. Through the above analysis, it can be seen that these 4 cases contain rich flow phenomena, so they are suitable as final test cases.

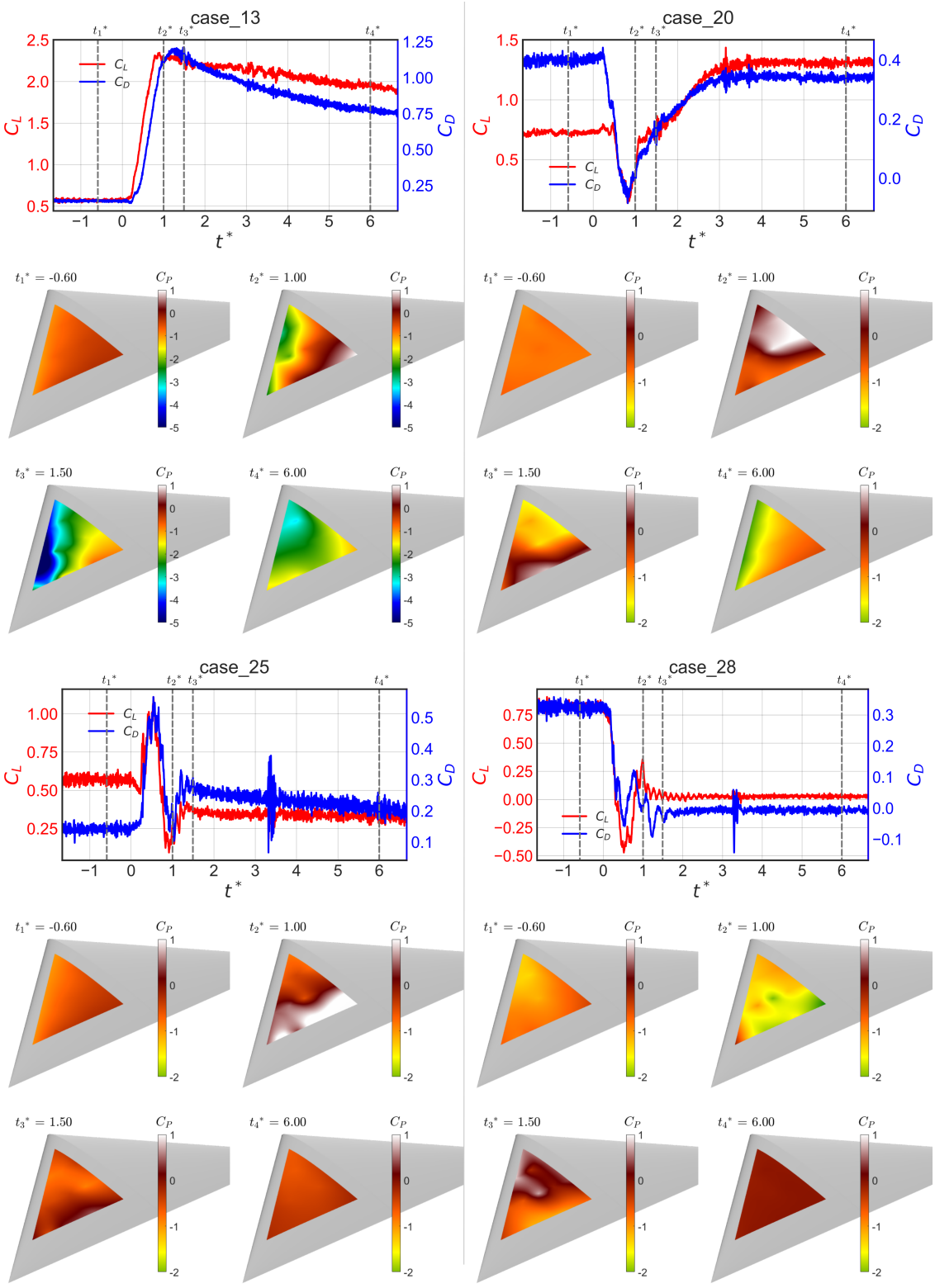


Figure 5: Gust loads and surface pressure maps of the 4 cases. The 4 pressure maps under loads of each case correspond to the 4 times shown in the load figures.

3 Multilayer Perceptron Regression Model

The aerodynamic loads of the vehicle are the integral of surface stresses (surface pressure and viscous shear stress). In Sec. 1, it is learned analytically that, in separated flows, it is feasible to predict aerodynamic forces only from surface sparse pressure measurements, neglecting viscous shear stress. So, only pressure measurements are considered in this article. Therefore, the aerodynamic loads C_X can be expressed as a nonlinear function of the pressure P_N:

$$C_X(t) = C_X[C_{P_1}(t), C_{P_2}(t), ..., C_{P_N}(t))] \tag{5}$$

Where C_X represents the lift (C_L), drag (C_D), and moment (C_M) coefficient, and C_{P_N} is the pressure coefficient of the N_{th} surface pressure tap.

Note that there are many other candidate input features that can be used for unsteady aerodynamic modeling, for example, Wang et al. [27] use $[\alpha(\tau),\ \alpha(\tau-8),\ \alpha(\tau-16),\ \alpha(\tau-24),\ \alpha(\tau-32),\ \bar{q}(\tau)]$ as inputs where τ is the current time and $\bar{q}$ is the nondimensional pitching rate; He et al. [42] use $[C_P, \dot{C}_P, C_P \circ \dot{C}_P]$ as input variables for loads identification in transverse gust encounters, where $\dot{C}_P$ is the first derivative of C_P, $C_P \circ \dot{C}_P$ is the product of C_P and $\dot{C}_P$. However, it is hard to determine how long data to use before the current state if history information is included, and solve extra partial differential equations if $\dot{C}_P$ is included. In this research, we only use C_P itself as input features, no history information, and no derivatives to ignore such problems. C_L and C_D are the output target, but they are predicted separately to reduce the propensity learning of the model due to magnitude differences. Hence, there are 16 input features and 1 target for our MLP models.

3.1 The MLP regression

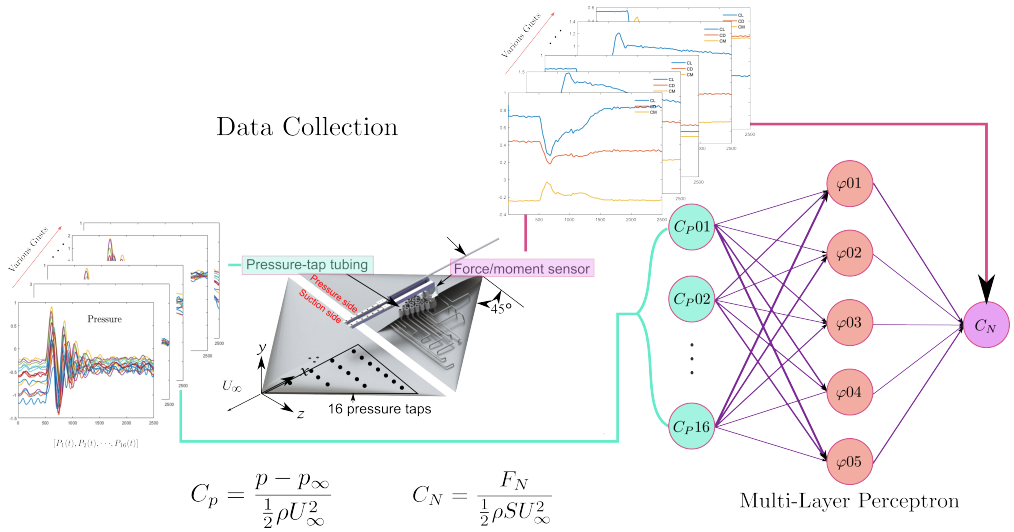


Figure 6: Schematic of the MLP regression process.

Multilayer Perceptron (MLP), developed by Rumelhart et al. [17] is one of the basic types of neuron network (NN), which consists of neurons and hidden layers while weights and biases are stored in neurons. MLP is considered a densely connected feed-forward NN, which means that there is no loops in the connection between the neurons, and every neuron is linked to all of the neurons in the next layer. The schematic of an MLP can be found in Fig. **??**. As described previously, surface pressure and load are collected when encountering gusts, then pressure is fed to the MLP model as input features (C_{P_N}), while the gust loads are fed as target (C_X). Then MLP model will give an output ($\hat{C}_X$) with the given input. The goal of the training is to find the weights and biases of the MLP model to minimize the error between $\hat{C}_X$ and C_X. Usually, this error is given by a loss function (also called error function, objective function):

$$L = \frac{1}{2}\Sigma(\hat{C_X} - C_X)^2 \tag{6}$$

Here L is the loss function with Mean Square Error (MSE), which is used to evaluate the performance of most regression algorithms. The coefficient $1/2$ is added for the convenience of solving the derivative.

The renowned non-linear fitting capabilities of MLP is a blessing for the modeling of complex flow problems but also a limitation due to its over-fitting tendency [43, 44]. The goal of MLP regression is not to get a good description of training data, but to be able to predict what is unseen in the training process. Several methods are developed by the ML community to mitigate or prevent over-fitting, such as dropout [45], $L1$ (Lasso), $L2$ (Ridge) regularization [46], and application of noise [47]. Dropout is to randomly ignore some neurons and hence no contribution to the activation of the forward neurons. This technique is widely used for large models such as convolution neuron networks which usually have 1000 features at least. So it is not suitable for our problem since we only have 16 features. $L1$ regularization is often used in linear regression and feature selection because of its ability to identify a sparse solution. $L2$ regularization penalizes the sum of squared weights, which is more suitable for complex patterns. The loss function with $L2$ regularization:

$$L = \frac{1}{2}\sum(\hat{C_X} - C_X)^2 + \frac{\lambda}{2}\sum(\mid \omega_j \mid)^2 \tag{7}$$

where $\hat{C_X}$ is the predicted aerodynamic loads, and ω_j is the weight of the neurons in the MLP model, and λ is the regularization coefficient. $L1$ and $L2$ regularization both can address over-fitting, but this technique did not improve the prediction of our problem as we did many tests, while it will cause more calculation, so no $L1$ or $L2$ regularization was used in this research. As mentioned in Section 2, there is no filter smoothing for gust loads, but just ensemble-averaged. Therefore, much noise still exists in the loads, as shown in Fig. 5, which will also help mitigate over-fitting. In Section 4, we will see that MLP can even ignore the outliers due to the mechanical vibration of the towing motor.

All the hyperparameters of MLP used in this research are summarized in Fig. 7. Among these, the neuron and layer are the ones that need to be optimized through cross-validation. *PReLU* (Parametric Rectified Linear Unit) is used as the activation function, as it allows negative values to contribute to the activation, which is an optimization of *ReLU* (Rectified Linear Unit), a widely used activation function in regression problems. *PReLU* and its derivative are shown in the top-right of Fig. 7, as can be seen, that the derivative of negative values is not 0. There is a coefficient in *PReLU* that also needs to train, but the increased computation is negligible.

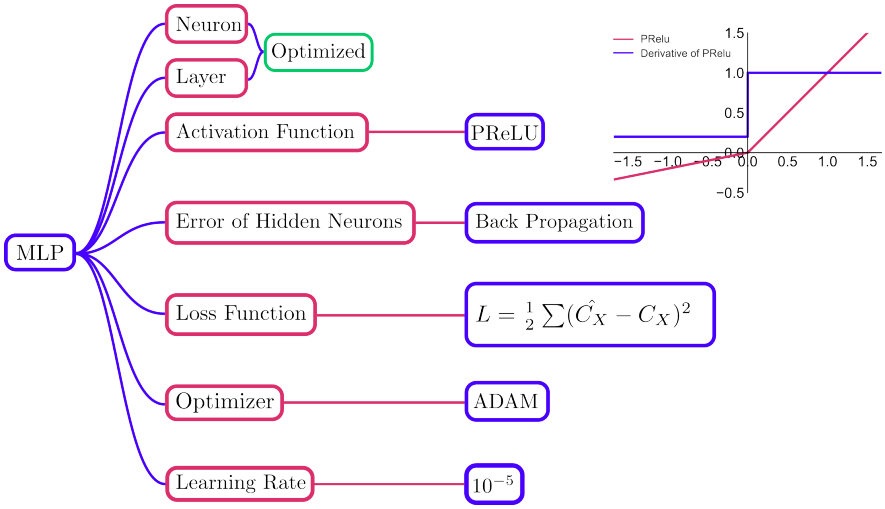


Figure 7: MLP hyperparameters used in this research.

3.2 Hyperparameter Optimization

Neurons and layers are chosen for optimization as they are the most vital elements in MLP. Cross-validation is a necessary practice to ensure the legitimacy of the model and can be used for hyperparameter optimization. In this research, k-fold cross-validation is used. By dividing the data into different subsets, the MLP model with different neural, and hidden layers are used to train on each subset, and the best hyperparameters are selected by comparing the averaged results. Figure 8 shows the $8-fold$ cross-validation optimization process. The entire data set is divided into 8 splits on average (each has 4 cases), one of which is randomly selected as the test set, and the remaining 7 splits are used for the cross-validation, by which the best neurons and layers can be selected. As depicted in this figure, 7 folds are used for cross-validation, while the last one, *Fold* 8 is for the final test: the model with best neurons and layers is trained with the training sets of *Fold* 8 , then test using the test sets of *Fold* 8; in this way, the final prediction is obtained. To eliminate the influence of the initial condition of the weight and bias of MLP, the final *Fold* 8 is repeated 30 times and ensemble-averaged to eliminate the effect of randomness.

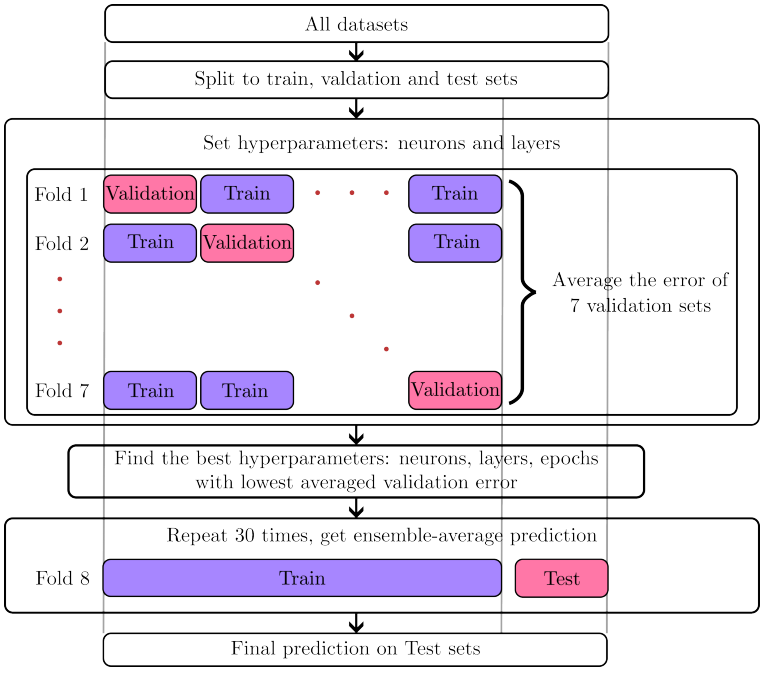


Figure 8: The $8-$*Fold* cross-validation used in this paper. Purple boxes represent the training data used to derive the machine-learning model. The red box represents the validation data used to validate the trained model. The average result of the 7 validation sets will be used to evaluate the performance of the corresponding hyperparameters. After the best hyperparameters are selected, they will be used on the training set of Fold 8, and then the final test will be done on the test set of Fold 8.

The details of the 8 folds for cross-validation can be found in Table 2. *Fold* 1 to *Fold* 7 have 24 training cases and 4 validation cases, while *Fold* 8 has 28 cases for training and 4 for the final test. Note that *Fold* 1 to *Fold* 7 as cross-validation cases do not include any cases in the test case of *Fold* 8, which ensures the independence of the final test.

Table 2: Details of the 8 folds for cross-validation

Fold	Train case	Validation case
1	01 02 03 05 06 07 08 09 10 11 12 14 15 16 17 18 21 22 23 24 27 29 31 32	04 19 26 30
2	01 02 03 04 06 07 08 09 10 11 12 15 16 17 18 19 21 22 23 24 26 27 29 30	05 14 31 32
3	02 03 04 05 06 07 08 09 10 11 12 14 15 16 17 19 21 23 24 26 27 30 31 31	01 18 22 29
4	01 02 03 04 05 07 08 09 11 12 14 15 18 19 21 22 23 24 26 27 29 30 31 32	01 18 22 29
5	01 03 04 05 06 07 09 10 12 14 15 16 17 18 19 21 22 23 26 27 29 30 31 32	02 08 11 24
6	01 02 04 05 06 08 09 10 11 12 14 16 17 18 19 21 22 23 24 26 29 30 31 32	03 07 15 27
7	01 02 03 04 05 06 07 08 10 11 14 15 16 17 18 19 22 24 26 27 29 30 31 32	09 12 21 23
Fold	Train case	Test case
8	01 02 03 04 05 06 07 08 09 10 11 12 14 15 16 17 18 19 21 22 23 24 26 27 29 30 31 32	13 20 25 28

4 Results and discussion

Burelle et al. [40] studied the linear mapping of sparse pressure data and the aerodynamic loads in various axial and vertical gusts by linear regression. Their study shows a robust pressure-load mapping with aggregate data, although with slightly large errors, but it successfully shows the relationship between surface pressure and aerodynamic loads and the possibility to model loads with only pressure data. Based on this, we further not only studied descriptive, but the predictive of data-driven method in various highly gusty environments by nonlinear MLP model.

Tensorflow + keras open source library are used for building and training MLP model, which are libraries that provide highly powerful building blocks.

4.1 Lift Prediction

4.1.1 Cross-validation results

For the hyperparameter optimization, we made various MLP models with different neurons and layers for the $8-$*Fold* cross-validation as described in Section 2. For neurons, they are $\{16,24,32,40,48,64,72\}$, while for layers, they are $\{1,2,3,4,5,6,7,8\}$. The specific training process is: we first fix the number of layers, and then loop through all the number of layers; then fix the next number of layers and repeat the above step. So there are $7\times 8=56$ models in total. The pseudocode is illustrated in Table 1. Note there is a variable "*predictions*". After each epoch training, we use the corresponding validation cases to verify and get a temporary fit. This variable is used to store the fitting results of each epoch. In this way, the validation error of each epoch is obtained, through which the over-fitting can be observed, and the training convergence can also be learned, so that we can stop the training early. The validation error is recorded instead of the training error because MLP is easily over-fitting to the gust datasets in this research and always gets a lower and lower training error.

Algorithm 1: Pseudocode of the cross-validation in MLP

```
Initialize:
    num_neuron ← [16,24,32,40,48,56,64,72]
    num_layer ← [1, 2, 3, 4, 5, 6, 7, 8]
    folds ← 7
for i in num_neuron do
    for j in num_layer do
        for k in 0 to folds-1 do
            predictions[k, i, j] ← []
            model ← initialize_sequential_model()
            neuron ← i
            add_input_layer_to_model(model)
            for jj in 0 to j-1 do
                add_hidden_layer_to_model(model, neuron)
                add_prelu_to_model(model)
            end for
            add_output_layer_to_model(model)
            compile_model(model)
            train_model(model, P_train, F_train)
            predictions[k, i, j] ← fit_model(P_validation)
        end for
    end for
end for
```

Before training, the surface pressure with and without smoothing is compared to check the influence on the dynamic response of the pressure tubes. Kawata et al. [48] studied the dynamic response of the fluctuating hydraulic pressure measurement under water using pressure tubes. They found that the length of the tubing clearly influences the step response, in both phase lag and amplitude of oscillation. The longer the tube, the more pronounced this effect. In our experiment, the model is $0.5m$ under water and the transducers are $0.5m$ above water, so the tube is more than $1m$ long since extra length is needed for pitching and fixing. Hence, it is necessary to consider reducing the influence of the dynamic response of tubing. Kawata et al. [48] proposed a model equation to calibrate the dynamic response, in which 3 coefficients need to be determined. Instead of using their equation to find the 3 coefficients, we use the Bessel filter, an analog linear filter, to smooth the dynamic response of the tubing. The Bessel filter is: $Bessel(8, 0.01)$, 8 is the order of the filter, 0.01 is the critical frequency, which yields a $Nyquist\ frequency = 5$ based on the $sampling\ frequency = 1000$.

The *case*_13 is chosen to show the effect of Bessel filtering. Figure 9 shows the comparison, in which 2 taps are selected, $p0$ and $p15$. $p0$ is at the stagnation point in front of the model, so the C_P is constant 1 before the gust, while $p15$ is the one outermost, and the C_P is still constant, indicating this tap is in separated region, which can be identified by the small pressure gradient around it (blue region) in Fig. 10. The C_P of both $p0$ and $p15$ has oscillation that lasts for a while but is smoothed by the Bessel filter.

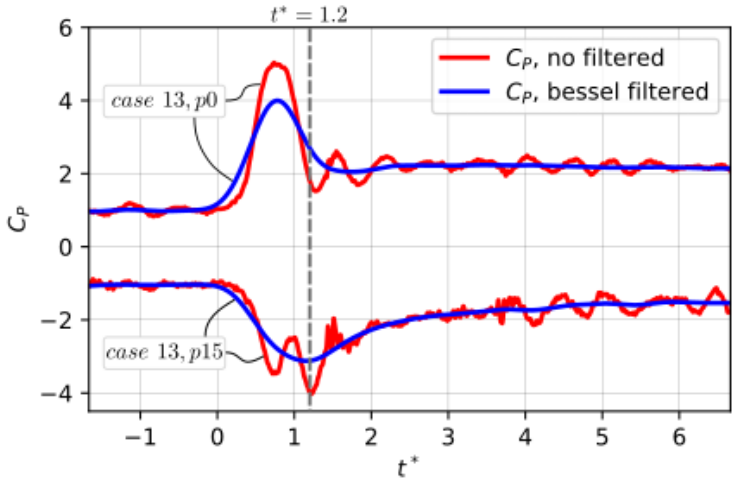
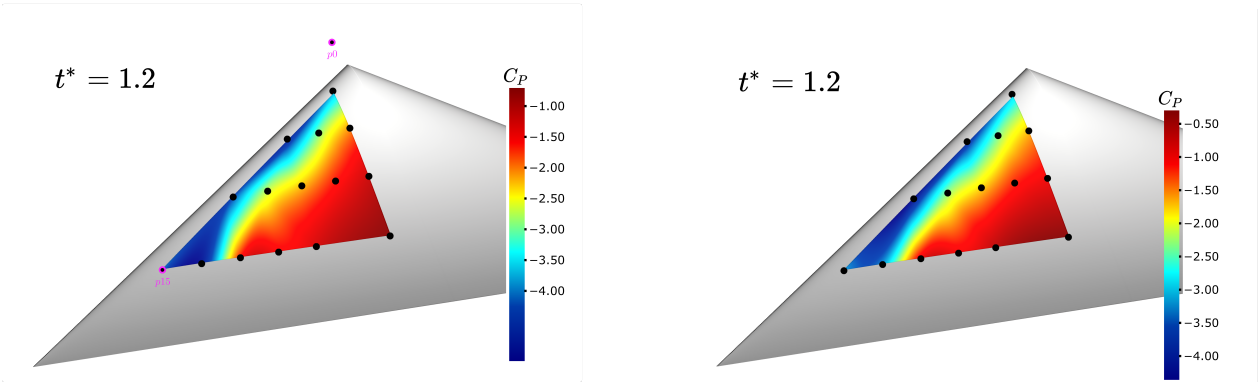


Figure 9: Pressure comparison with and without filtering for case13. These two pressure are from $p0$ (the stagnation point) and $p15$.

In this Fig. 9, a gray dashed line shows at $t^* = 1.2$, and the pressure at all taps is shown in Fig. 10 corresponding to this time point. The two pressure contour maps are almost indistinguishable except at the pressure tap before $p15$ on the leading edge, which indicates that the Bessel filter does not change the overall pressure distribution. Therefore, the $8-Fold$ cross-validation is used Bessel-filtered pressure, while the gust loads keep unchanged.



(a) Pressure map without filtering.

(b) Pressure map with Bessel filtering.

Figure 10: The pressure map comparison with and without filtering for case 13. The two ports, 0 and 15 are marked as pink color in the figure.

Figure 11 shows the mean square error (MSE, see Equation 8) of C_L and $\hat{C}_L$ distribution of the 7 cross-validation folds. We first fix the epoch of each model to 50 (later will show the validation error convergence history). The $x-axis$ represents the number of layers, while the $y-axis$ is the number of neurons. The error is the ensemble average of the 7 folds. For example, the MSE of Model with $\{Neuron,\ Layer\} = \{5,56\}$ is the ensemble average of the 7 folds, which can be found in the Table 1. This MSE contour map shows a pattern between MSE and neurons layers: the more neurons and layers, the lower MSE, as shown by the blue dashed line in the Fig. 11 (MSE from high to low, from the lower left corner to the upper right corner).

$$MSE(C_L, \hat{C}_L) = \frac{1}{n_{samples}} \sum (C_L - \hat{C}_L)^2 \tag{8}$$

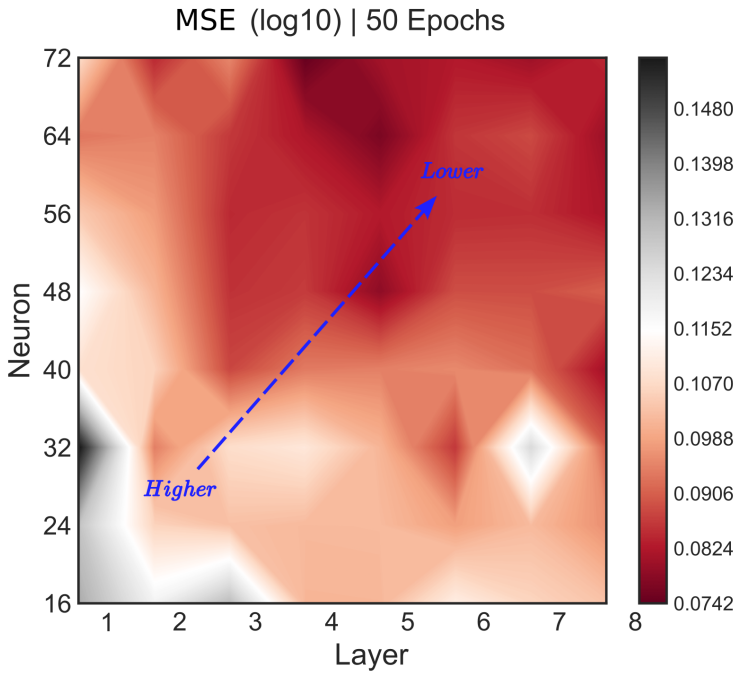


Figure 11: Mean square error contour at 50 epochs. The error is displayed as a base 10 logarithm.

Now, let's look at the MSE convergence history of the model with $\{Neuron,\ Layer\} = \{8, 72\}$. Here we choose two stop epochs, 50 and 200, to show the history, and compare the influence of epochs. Figure 12 shows the averaged MSE convergence history, in which 50 and 200 epochs are indicated by gray dashed lines. It shows that the MSE first reduces to a plateau at 50 epochs (that is why we show the MSE of 50 epochs in Fig. 11), and then keeps reducing until 200 epochs. For the situation like this, the MLP may get over-fitting at the first plateau, 50 epochs. So 50 and 200 epochs will both analyze to minimize over fitting. Figure 13 shows the same MSE contour map but with 200 training epochs. The pattern is much clearer in this figure with 200 epochs than the one with 50 epochs.

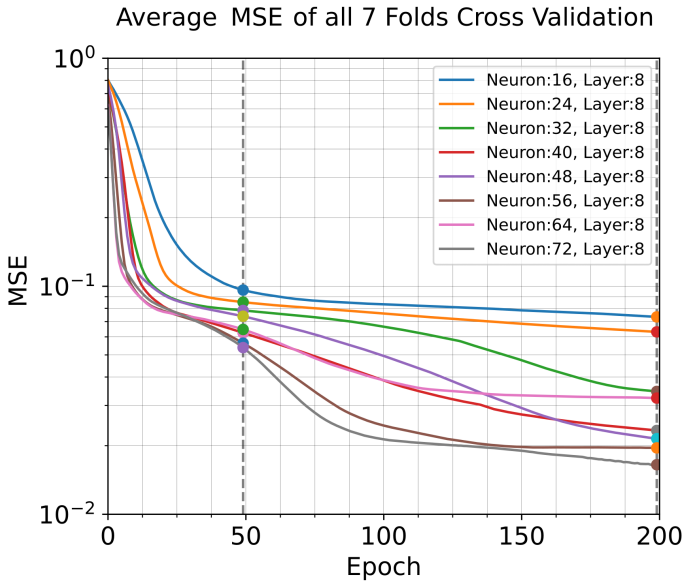


Figure 12: The average mean square error of all the 7 cross-validation folds. All the MLP models in this figure are with 8 layers.

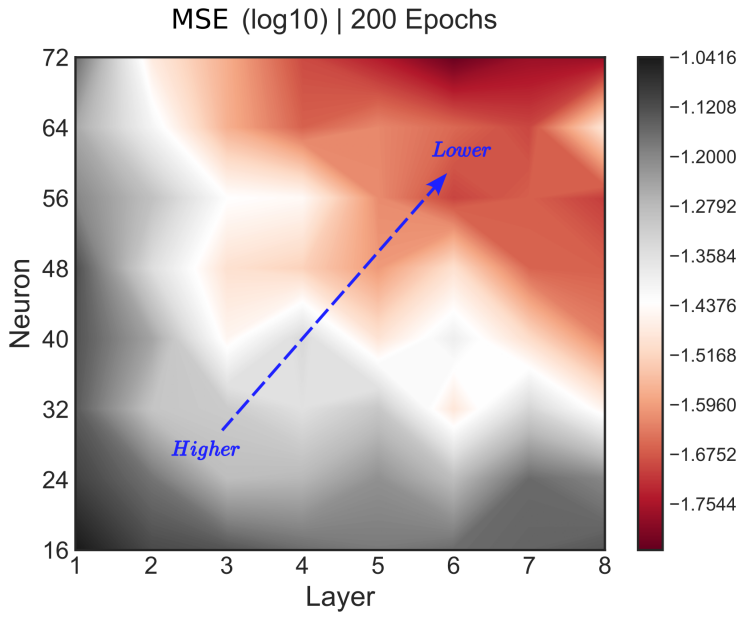


Figure 13: Mean square error contour at 200 epochs.

Both Fig. 11 and Fig. 13 show the pattern that the MSE is positively correlated with the number of neurons and layers. This does not mean that more neurons and hidden layers should be used, because the calculation will increase exponentially, while the MSE does not decrease much. Hence, we choose $\{Neuron,\ Layer\} = \{72, 8\}$ as the best hyperparameters in this paper.

4.1.2 Prediction without filtering

Now the best hyperparameters have been identified, as summarized in Table 3, in which neurons and layers are optimized by cross-validation, while others are predefined. *Fold* 8 now can be used for the final test with these optimized hyperparameters. We will first test train and test using the pressure without filtering as baseline and check the influence of tubing dynamic response on MLP.

Table 3: Summary of the best hyperparameters

Hyperparameters	Neuron	Layer	Epoch	Activation Function	Learning Rate
Value/Item	72	8	50/200	PReLU	10^{-5}

Firstly the MLP model is trained on the training datasets in *Fold* 8, then the trained model is used to test with the test datasets in *Fold* 8. As the weights and biases of MLP model are initialized randomly, the model is repeated times and gets ensemble-averaged to minimize the effects of randomness. Then we calculate the standard deviation (STD) of the $\hat{C_L}$ of the repeated test to check out the number of repetitions required. The result is shown in Fig. 14, which indicates that the STD gets converged at around 20 repeats. To be conservative, we choose 30 repetitions for training.

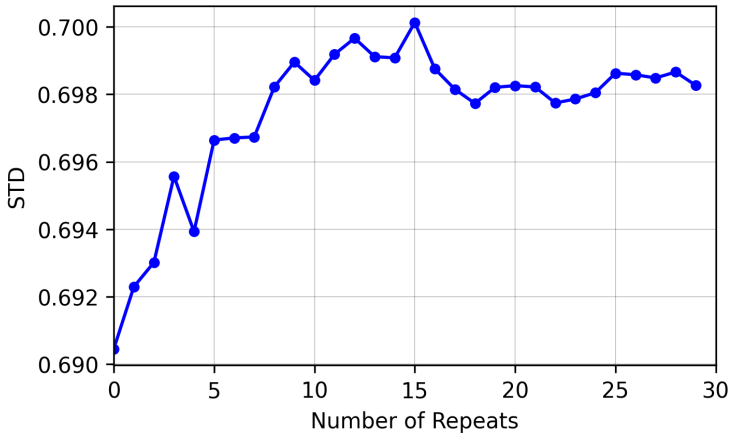


Figure 14: The standard deviation history with the number of repeats without filtering. Each repeat runs 200 epochs.

Figure 15 shows the MSE history with 30 repeats, in which only the MSE from 50 epochs to 200 epochs is shown. The MSE gets converged after about 150 epochs, which is consistent with the results of the cross-validation: 200 epochs.

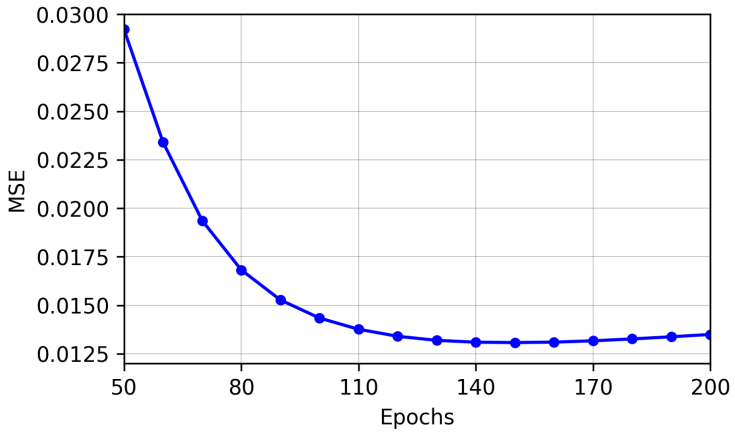


Figure 15: The averaged error history of training without filtering.

The final prediction of C_L on the test datasets of *Fold* 8 is shown in Fig. 16. Firstly, the MLP predicts very well the load response in the steady phase before and after the gust region, and this is reasonable, because the initial states and final states of the test cases are already included in the training datasets (see Table 1). And MLP can also accurately capture the load response in the early stage of the gusts, but the error becomes large after the gusts, except the *case*_28, which has the smallest MSE and hence the best prediction. The prediction of *case*_13 is good before half of the gust, and then it underestimates the load until the end of the gust. For *case*_20, MLP can totally capture the load response in the entire gust, but immediately has a large oscillation, and then quickly recovers. *case*_25 has a similar problem, but it underestimates the load at the end of the gust. In the prediction of all the 4 cases, three of them have problems during and after the gusts. One of the possible reasons is the dynamic response of the long pressure tubes. To verify this, we retrain the MLP model with the Bessel-filtered pressure data.

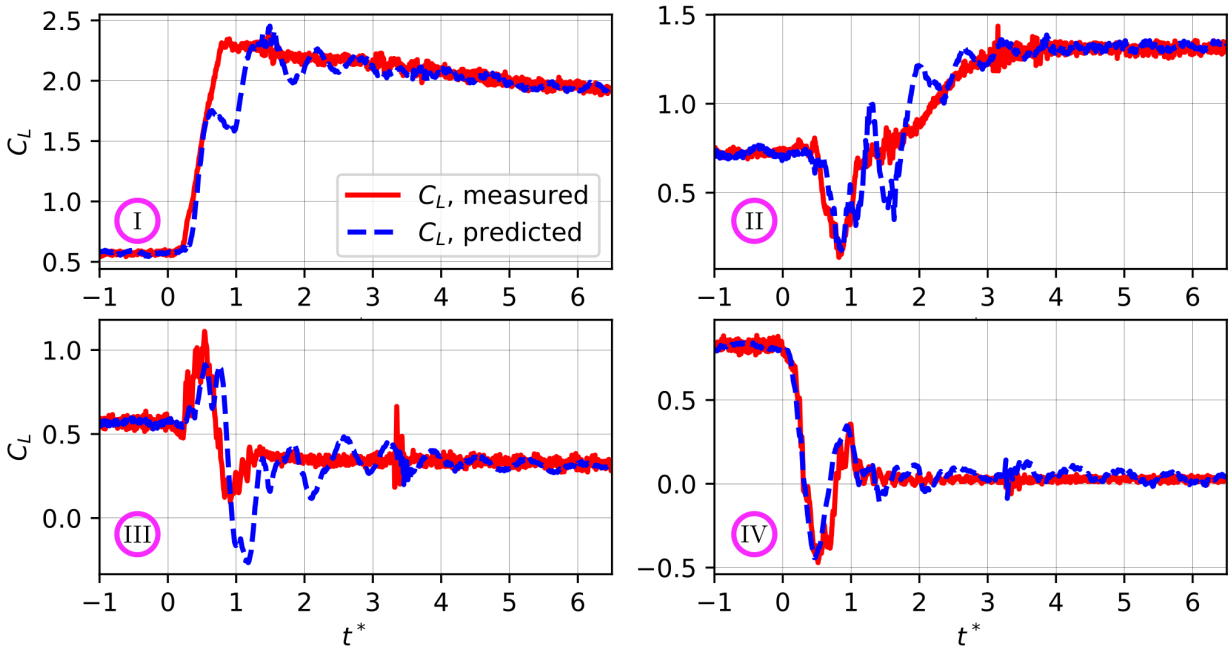
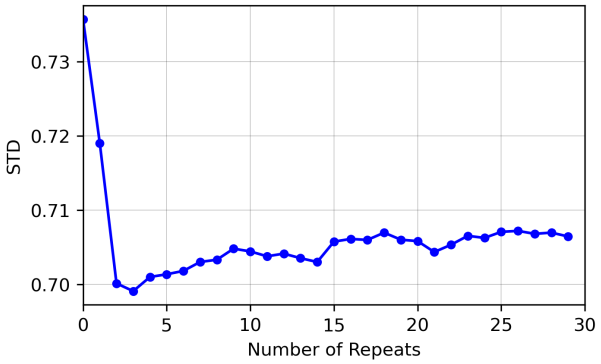


Figure 16: C_L prediction of the four test cases without filtering. which are *case*_13, *case*_20, *case*_25 and *case*_28 in order. The training costs 200 epochs.

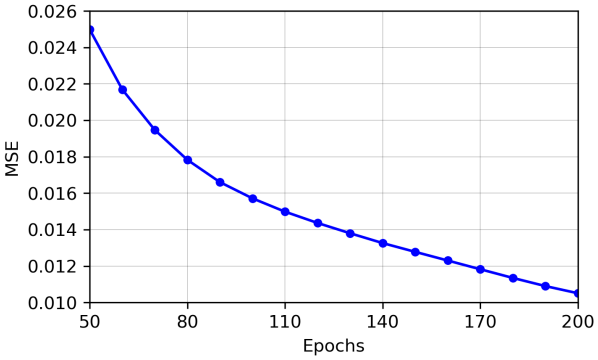
4.1.3 Prediction with Bessel filtering

The Bessel-filtered pressure is discussed in Subsection 4.1.1, and we find that the Bessel filter does not change the structure of the pressure distribution. Therefore, MLP training with Bessel-filtered pressure is checked.

Figure 17 shows the same operations as in the Subsection 4.1.2. Figure 17a also shows that 30 repeats are enough to get a converged STD, and Fig. 17b indicates 200 epochs is also suitable for training with Bessel-filtered data. Finally, the prediction of the 4 test cases is shown in Fig. 18. *case*_28 is still with high-quality prediction, while *case*_13 and *case*_25 get improved during the gust prediction, the underestimation is gone, while for the prediction of *case*_13, a phase lag is observed. Only prediction of *case*_20 is not improved, and it overestimated the load during the gust. Overall, the prediction gets much better compared to those without filtering. And that confirms the previous conjecture: the long pressure tubes do affect the learning of MLP. An added bonus is the MLP's ability to handle outliers caused by vibrations generated by the towing motor, as shown in *case*_25 and *case*_28, at about $t^* = 3.4$. MLP just ignores it and gets a good prediction.



(a) The STD history with the number of repeats with Bessel-filter.



(b) The averaged error history of training with Bessel-filter.

Figure 17: The STD history and MSE with Bessel-filter.

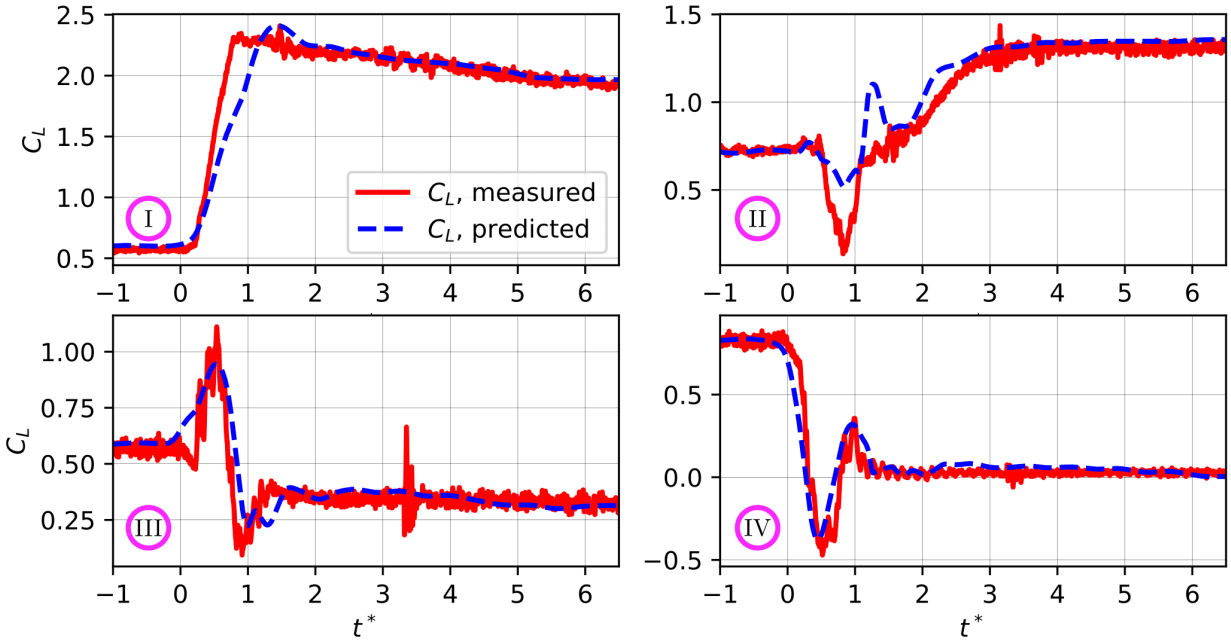


Figure 18: C_L prediction of the four test cases with Bessel filter. The training costs 200 epochs.

Is there any reason for the poor prediction of *case*_20? Let's go back to the test matrix (see Table 1), and plot the gust conditions together to get a Gust Envelope (analogous to the flight envelope, a range of safe flight). As shown in Fig. 19, the gust envelope has 3 axes, α_{start}, α_{end}, and u_{end}, and all 32 cases are represented as circles. The 4 test cases of *Fold* 8 are shown in red. A blue rectangle is inserted into each α_{end}, which means that all the cases in the same blue rectangle have the same α_{end}. *case*_20 is just at the top of the α_{end} rectangle, with no case above or outside of it, i.e. it is on the top boundary of the gust envelope. MLP can learn little about its neighbors, and that's why the prediction is poor. As for the other 3 cases, they are also on the boundary of the gust envelope, but the gust strength is not strong as *case*_20. This indicates that more cases outside of the gust envelope are needed if we want to predict well the cases on the boundary with strong gusts.

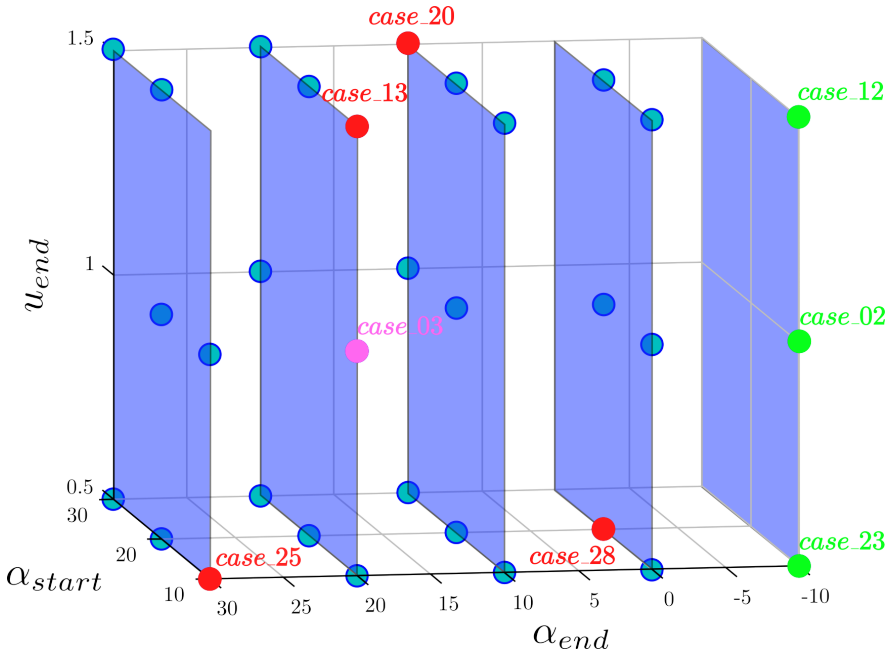


Figure 19: Gust Envelope of all the 32 cases.

This can be verified by the 3 cases in green, i.e. *case*_02, *case*_12 and *case*_23 (which are on the boundary with strong gusts). We can also pick one more case in the middle of the gust envelope, for example, *case*_03, which is in pink (should be easy to predict), to group a new *Fold*_9. By testing *Fold*_9, we can figure out that if the boundary with strong gusts can have an influence on the prediction of MLP or not.

We use all the same hyperparameters as the ones used in Fig. 18 (except ensemble-average), and the result is shown in Fig 20, which clearly shows the influence of the bounday. Predictions of *case*_02, *case*_12 and *case*_23 are really poor, even can not predict well after the gust. This is just because there are no other cases in the blue $\alpha_{end} = -10^{\circ}$ rectangle in Fig. 19. MLP cannot learn anything about $\alpha_{end} = -10^{\circ}$. While the prediction of *case*_02 is relatively well, which is in good agreement with our conjecture: *case*_03 is in the middle of the gust envelope (though on the boundary, but not strong) and should have a good prediction.

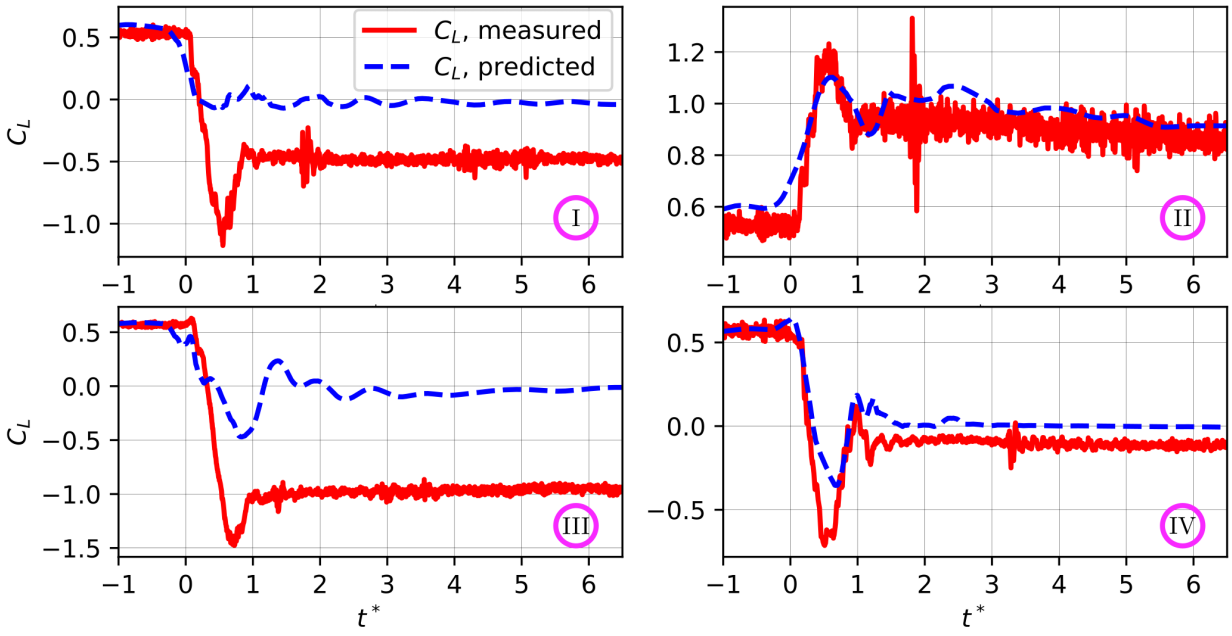


Figure 20: C_L prediction of *Fold*_9, which are *case*_02, *case*_03, *case*_12 and *case*_23 in order.

5 Conclusion

This paper mainly focuses on the aerodynamic load modeling in highly unsteady flows, typically in gust environments, using real-time noisy sparse pressure measurement, by MLP, a type of Machine Learning model. A non-slender delta wing experiencing such gusts is studied, and its gust response including aerodynamic loads and surface pressure is measured by a force/torque transducer and 16 pressure taps on the suction side/stagnation point of this delta wing. To enrich the gust environment as much as possible, two types of gust in both axial and vertical directions are combined together with various initial and end conditions, to form 32 gusts with different directions and strengths. An MLP model is built to train and test the datasets.

The 32 gusts contains contain rich flow structures, detached flows, attached flows, fully separated flows. These flow structures can be found through the interpolation surface pressure contour maps. That is good for Machine Learning to learn and train. The influence of the dynamic response of the long pressure tubes is analyzed. A Bessel filter is used to mitigate such dynamic responses, while keeping the distribution of the surface pressure.

The 32 cases are carefully splitted into 8 folds for cross-validation, in which 7 folds are used for hyperparameter optimization, and *Fold*_8 is for the final test. In this paper, only neuron and layer are selected to be optimized through cross-validation, while others are predefined. The neuron and layer are looped from a pool to be selected, which contains 8 different layers and 7 various neurons. The Mean Square Error of measured loads and validated loads of each model with different neurons and layers are ensemble-averaged on the 7 folds. The error contour map shows a clear pattern that the more neurons and layers, the lower the error. Then the neuron and layer with lowest error are chosen as the best hyperparameters.

To analyze the influence of pressure tubes' dynamic response on the MLP training, the pressure without filter is first used to train with the final *Fold*_8. The training process is repeated multiple times to eliminate the influence of random initialization of weights and biases. The results show that the dynamic response of the pressure tubes does affect MLP learning the load response during gusts. When using Bessel filter to smooth the pressure dynamic response, the prediction gets much better.

Finally, the boundary problems of the gust envelope are studied. When using data on the boundary, the predictions

are poor, and vice versa. This shows that the extrapolation characteristics of MLP are worse than the interpolation characteristics, which is confirmed by a new *Fold_9*. Therefore, it is very important to train the neural network with as much data as possible. This is not only about the amount, but more importantly, the training data should contain as many gust conditions as possible, so that the proportion of boundary data in the gust envelope can be reduced, thereby improving the overall prediction performance of MLP.

References

[1] John E Lamar. Recent studies of subsonic vortex lift including parameters affecting stable leading-edge vortex flow. *Journal of Aircraft*, 14(12):1205–1211, 1977.

[2] JZ Wu, AD Vakili, and JM Wu. Review of the physics of enhancing vortex lift by unsteady excitation. *Progress in Aerospace Sciences*, 28(2):73–131, 1991.

[3] Zhen-tao Zhao, Wei Huang, Li Yan, Tian-Tian Zhang, Shi-bin Li, and Feng Wei. Low speed aerodynamic performance analysis of vortex lift waveriders with a wide-speed range. *Acta Astronautica*, 161:209–221, 2019.

[4] Edward C. Polbamus. A concept of the vortex lift of sharp-edge delta wings based on a leading-edge-suction analogy. Report, NASA, 1966.

[5] Edward C. Polbamus. A concept of the vortex lift of sharp-edge delta wings based on a leading-edge-suction analogy. *NASA Technical note*, 1967.

[6] Maziar S Hemati, Scott TM Dawson, and Clarence W Rowley. Parameter-varying aerodynamics models for aggressive pitching-response prediction. *AIAA journal*, 55(3):693–701, 2017.

[7] Yong Zou, Reik V Donner, Norbert Marwan, Jonathan F Donges, and Jürgen Kurths. Complex network approaches to nonlinear time series analysis. *Physics Reports*, 787:1–97, 2019.

[8] Kunihiko Taira, Aditya G Nair, and Steven L Brunton. Network structure of two-dimensional decaying isotropic turbulence. *Journal of Fluid Mechanics*, 795, 2016.

[9] Abin Krishnan, RI Sujith, Norbert Marwan, and Jürgen Kurths. Suppression of thermoacoustic instability by targeting the hubs of the turbulent networks in a bluff body stabilized combustor. *Journal of Fluid Mechanics*, 916, 2021.

[10] AH Shirazi, G Reza Jafari, J Davoudi, J Peinke, M Reza Rahimi Tabar, and Muhammad Sahimi. Mapping stochastic processes onto complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(07):P07046, 2009.

[11] Daniel Fernex, Bernd R Noack, and Richard Semaan. Cluster-based network modeling—from snapshots to complex dynamical systems. *Science Advances*, 7(25):eabf5006, 2021.

[12] Aditya G Nair, Chi-An Yeh, Eurika Kaiser, Bernd R Noack, Steven L Brunton, and Kunihiko Taira. Cluster-based feedback control of turbulent post-stall separated flows. *Journal of Fluid Mechanics*, 875:345–375, 2019.

[13] Giovanni Iacobello, Frieder Kaiser, and David E Rival. Load estimation in unsteady flows from sparse pressure measurements: Application of transition networks to experimental data. *Physics of Fluids*, 34(2):025105, 2022.

[14] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Assessment of supervised machine learning methods for fluid flows. *Theoretical and Computational Fluid Dynamics*, 34(4):497–519, 2020.

[15] Jian Yu and Jan S Hesthaven. Flowfield reconstruction method using artificial neural network. *Aiaa Journal*, 57(2):482–498, 2019.

[16] Binglin Li, Zixuan Yang, Xing Zhang, Guowei He, Bing-Qing Deng, and Lian Shen. Using machine learning to detect the turbulent region in flow past a circular cylinder. *Journal of Fluid Mechanics*, 905, 2020.

[17] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[18] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[19] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Dennis J Linse and Robert F Stengel. Identification of aerodynamic coefficients using computational neural networks. *Journal of Guidance, Control, and Dynamics*, 16(6):1018–1025, 1993.

[22] Scott J Schreck, William E Faller, and Marvin W Luttges. Neural network prediction of three-dimensional unsteady separated flowfields. *Journal of aircraft*, 32(1):178–185, 1995.

[23] William E Faller and Scott J Schreck. Neural networks: applications and opportunities in aeronautics. *Progress in aerospace sciences*, 32(5):433–456, 1996.

[24] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

[25] Hugo FS Lui and William R Wolf. Construction of reduced-order models for fluid flows using deep feedforward neural networks. *Journal of Fluid Mechanics*, 872:963–994, 2019.

[26] Xiang Chen, Haifeng Wang, Jingxia Zhan, Ke Chen, and Yuan Cao. Unsteady aerodynamic modeling based on recurrent neural network. *Aerodynamic Research Experiment*, 32(1):101, 2020.

[27] Qing Wang, Weiqi Qian, and Kaifeng He. Unsteady aerodynamic modeling at high angles of attack using support vector machines. *Chinese Journal of Aeronautics*, 28(3):659–668, 2015.

[28] Siva M Mangalam. Phenomena-based real-time aerodynamic measurement system (prams). In *2003 IEEE Aerospace Conference Proceedings (Cat. No. 03TH8652)*, volume 7, pages 3347–3356. IEEE, 2003.

[29] Arun Mangalam, Siva Mangalam, and Peter Flick. Unsteady aerodynamic observables for gust load alleviation. In *49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 16th AIAA/ASME/AHS Adaptive Structures Conference, 10th AIAA Non-Deterministic Approaches Conference, 9th AIAA Gossamer Spacecraft Forum, 4th AIAA Multidisciplinary Design Optimization Specialists Conference*, page 1725, 2008.

[30] Z Wu, Y Cao, and M Ismail. Gust loads on aircraft. *The Aeronautical Journal*, 123(1266):1216–1274, 2019.

[31] Frederic M Hoblit. Gust loads on aircraft: concepts and applications. 1988.

[32] Kaman Thapa Magar, Gregory W Reich, Corey Kondash, Keith Slinker, Alexander M Pankonien, Jeffery W Baur, and Brian Smyers. Aerodynamic parameters from distributed heterogeneous cnt hair sensors with a feedforward neural network. *Bioinspiration & biomimetics*, 11(6):066006, 2016.

[33] Ihab Samy, Ian Postlethwaite, Da-Wei Gu, and John Green. Neural-network-based flush air data sensing system demonstrated on a mini air vehicle. *Journal of aircraft*, 47(1):18–31, 2010.

[34] He Shen, Yunjun Xu, and Charles Remeikas. Pitch control of a micro air vehicle with micropressure sensors. *Journal of Aircraft*, 50(1):239–248, 2013.

[35] Kenneth Thompson, Yunjun Xu, and Benjamin T Dickinson. Aerodynamic moment model calibration from distributed pressure arrays. *Journal of Aircraft*, 54(2):716–723, 2017.

[36] R Zhu, P Liu, XD Liu, FX Zhang, and ZY Zhou. A low-cost flexible hot-film sensor system for flow sensing and its application to aircraft. In *2009 IEEE 22nd International Conference on Micro Electro Mechanical Systems*, pages 527–530. IEEE, 2009.

[37] Ruiyi Que and Rong Zhu. Aircraft aerodynamic parameter detection using micro hot-film flow sensor array and bp neural network identification. *Sensors*, 12(8):10920–10929, 2012.

[38] Wei Hou, Darwin Darakananda, and Jeff D Eldredge. Machine-learning-based detection of aerodynamic disturbances using surface pressure measurements. *AIAA Journal*, 57(12):5079–5093, 2019.

[39] Kieran T Wood, Sergio Araujo-Estrada, Thomas Richardson, and Shane Windsor. Distributed pressure sensing–based flight control for small fixed-wing unmanned aerial systems. *Journal of Aircraft*, 56(5):1951–1960, 2019.

[40] Louis A Burelle, Wenchao Yang, Frieder Kaiser, and David E Rival. Exploring the signature of distributed pressure measurements on non-slender delta wings during axial and vertical gusts. *Physics of Fluids*, 32(11):115110, 2020.

[41] Xiaowei He and David R Williams. Pressure feedback control of aerodynamic loads on a delta wing in transverse gusts. *AIAA Journal*, pages 1–16, 2022.

[42] Xiaowei He and David R. Williams. Pressure feedback control of aerodynamic loads on a delta wing in transverse gusts. *AIAA Journal*, pages 1–16, 2022.

[43] J. Rynkiewicz. General bound of overfitting for mlp regression models. *Neurocomputing*, 90:106–110, 2012. Advances in artificial neural networks, machine learning, and computational intelligence (ESANN 2011).

[44] Shadi Abpeikar, Mehdi Ghatee, Gian Luca Foresti, and Christian Micheloni. Adaptive neural tree exploiting expert nodes to classify high-dimensional data. *Neural Networks*, 124:20–38, 2020.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[46] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, 31, 2018.

[47] Sophia Susan Raju, Boyan Wang, Kashyap Mehta, Ming Xiao, Yuhao Zhang, and Hiu-Yung Wong. Application of noise to avoid overfitting in tcad augmented machine learning. In *2020 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pages 351–354, 2020.

[48] Takuya Kawata, Hoshito Maeda, and Shinnosuke Obi. An attempt to measure fluctuating local pressure in free turbulent flow in water. *Journal of Fluid Science and Technology*, 9(2):JFST0014–JFST0014, 2014.