Multi-Fidelity Aerodynamic Dataset Generation of a Fighter Aircraft with a Deep Neural-Genetic Network

Murat MILLIDERE [1] and Fazil Selcuk GOMEC [2]
Middle East Technical University, Ankara, Turkey
Turkish Aerospace, Ankara, Turkey

Huseyin Burak KURT [3]
Turkish Aerospace, Ankara, Turkey

Ferhat AKGUL [4]
Middle East Technical University, Ankara, Turkey

This paper is a follow-up study on prior research work on multi-fidelity aerodynamic dataset generation. The prior work studied a comparison of modified Variable-Complexity Modelling and co-Kriging methods applied to F-16 fighter aircraft. In this research, the multi-fidelity deep neural-genetic network method is introduced. The results provide evidence that the deep neural-genetic network method in this paper can be employed in dealing with the aerodynamic data fusion problem.

Nomenclature

HF	=	High-Fidelity	y	=	Model Response Data
LF	=	Low-Fidelity	n	=	Number of samples in the Dataset
MF	=	Multi-Fidelity	CFD	=	Computational Fluid Dynamics
C_D	=	Drag Force Coefficient	VCM	=	Variable Complexity Modelling
C_L	=	Lift Force Coefficient	ANN	=	Artificial Neural Network
C_m	=	Pitch Moment Coefficient	FNN	=	Feedforward Neural Network
α	=	Angle of Attack	MLP	=	Multi-Layer Perceptron
δ_{ht}	=	Horizontal Tail Deflection	DNN	=	Deep Neural Network
GA	=	Genetic Algorithm	GA	=	Genetic Algorithm
z	=	Observed Response Data	DNGN	=	Deep Neural-Genetic Network

I.Introduction

There are several sources to generate the aerodynamic dataset. These are semi-empirical datasheet methods, linear flow solvers, nonlinear flow solvers, small-scale wind tunnel tests, full-scale wind tunnel tests, and flight tests in the increasing order of fidelity. As fidelity increases, computational time and cost increase. For aerodynamic database generation, various computational and experimental methods can be utilized with respect to the cost and accuracy

[1] Graduate Student, Ph.D. Candidate, Department of Engineering Sciences, AIAA Student Member.
[2] Aeroacoustics and Aerodynamic Analysis Chief Engineer.
[3] Flight Dynamics, Simulation and Control Engineer.
[4] Assoc. Prof., Department of Engineering Sciences.

requirements of the design phase. A summary of these methods with sample tools and classifications is provided [1] [2]. As expected, the higher the cost, the more accurate the aerodynamic database. Each method's resource and time cost are essentially tens to hundreds of times higher than those of the less accurate method [3].

Data fusion is one of the functional approaches to obtain an aerodynamic dataset. However, using only a high-fidelity wind tunnel test or high-fidelity computational analysis for the aerodynamic dataset generation is expensive to perform in terms of cost, time, or resources. Therefore, the primary motivation of data fusion is to achieve high-fidelity data in a cheaper fashion where low-fidelity aerodynamic data provides the trend information.

This paper presents a follow-up study from prior work [4]. Ref. [4] has shown the comparison of two different data fusion techniques being modified Variable-Complexity Modelling (VCM) [5] and co-Kriging method [6] [7]. The prior work also showed the benefit of using the co-Kriging method. In addition, this paper introduces a new technique called a deep neural-genetic network (DNGN).

Neural networks have shown significant successes in dealing with large-scale data. They can easily handle linear or nonlinear problems at low- and high-dimensions [8] [9]. Artificial neural networks (ANN) work as a general function approximation and approximate any continuous function to any desired accuracy by appropriate network architecture. This ability of neural networks makes it popular in aerospace problems. However, some elements are not autonomously updated in the training process for neural networks called hyperparameters. The hyperparameter is not the variable that needs to be tuned or optimized through neural network training, but the variable that is set by a priori knowledge, e.g., number of layers, number of nodes in each layer, solver algorithm, backpropagation algorithm factors, activation function for each layer, initial weights and biases scale factor and so on. The performance of a neural network is highly sensitive to the choice of hyperparameters. Settings for these hyperparameters can significantly influence the resulting accuracy of the predictive models, and there are no clear defaults that work well for different problems and data sets.

Choosing the best hyperparameters is the primary challenge in designing a neural network. If the developer doesn't have solid experience with the neural network, it is difficult to obtain a good model with the neural network. The new technique aims to automate the process of hyperparameter setting without the need for expert intervene. This explains the necessity of the optimization of these hyperparameters. In this study, hyperparameter optimization of deep feedforward neural network (FNN) was examined using a genetic algorithm, which is called a deep neural-genetic network.

In this paper, the wind tunnel test data is considered a high-fidelity dataset, and the data from the semi-empirical approach (Datcom) is considered a low-fidelity dataset. The aerodynamic dataset is generated using the DNGN approach.

The rest of the paper is organized as follows: Section II presents the proposed approach. Then, the results and discussion are presented in Section III. And finally, conclusions are drawn, and future works are given in Section IV.

II.Multi-Fidelity Deep Neural-Genetic Network Algorithm

In this section, the structure of deep feedforward neural networks and how the genetic algorithm is applied to deep feedforward neural network structure are covered.

A. Deep Neural Network Structure

The simplest type of feedforward neural network is the perceptron (artificial neuron), which has only an input layer and an output layer. A perceptron simply has three sets of rules; multiplication, summation, and activation. In general, the mathematical formula of an artificial neuron is given in Eq. (1).

$$a = \sigma\left(\left(\sum_i w^i u^i\right) + b\right) \tag{1}$$

where

- u^i - the inputs to the artificial neuron
- w^i - the weight value corresponding to each input.
- b - the bias
- σ - the activation function of an artificial neuron.
- a - the activation (output value) of artificial neuron.

The multi-layer perceptron (MLP) is a feedforward neural network consisting of many perceptrons. MLP has three kinds of layers; input layer, hidden layer, and output layer. Each neural network must have one input and output layer, but they can have many hidden layers. MLP with two or more hidden layers between the input and output nodes are often classified as feedforward deep neural network or deep neural network [10]. The power of feedforward neural networks to approximate complex nonlinear functions lies in the universal approximation theorem [11]. A simple MLP structure is given in Figure 1.

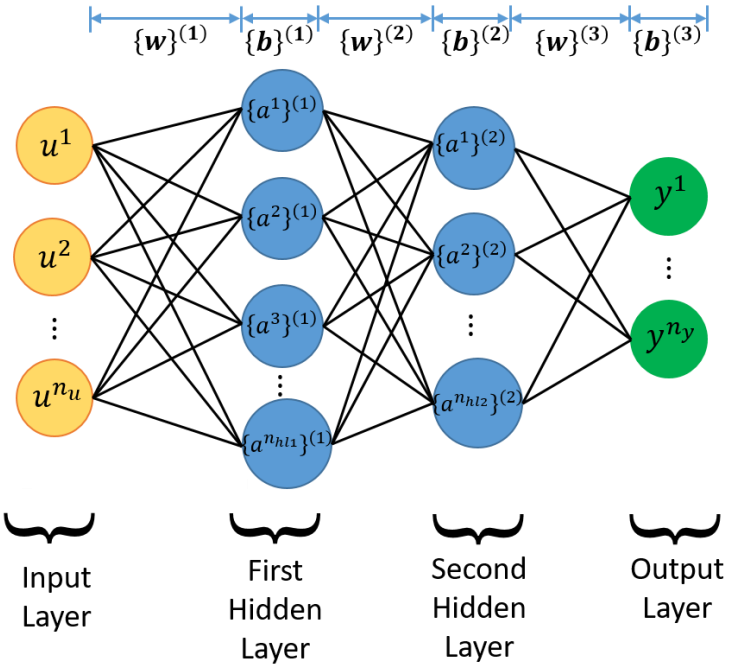


Figure 1 Simple neural network structure.

In Figure 1, there is a typical MLP neural network, which consists of four layers, and these are called the input layer, the first hidden layer, the second hidden layer, and the output layer, respectively. The output of each layer is obtained as follows:

$$\{a^i\}^{(k)} = \{\sigma\}^{(k)}\left(\left(\sum_j \{w^{ij}\}^{(k)}\{a^i\}^{(k)}\right) + \{b^i\}^{(k)}\right) \tag{2}$$

where

$\{a^i\}^{(k)}$ - the output of i^{th} neuron in the k^{th} layer

$\{w^{ij}\}^{(k)}$ - the weight of the connection from the j^{th} neuron in the $(k-1)^{\text{th}}$ layer to the i^{th} neuron in the k^{th} layer

$\left\{b^i\right\}^{(k)}$ - the bias term of the i^{th} neuron in the k^{th} layer.

$\{\sigma\}^{(k)}$ - the activation function in the k^{th} layer.

We refer to the activations of the input units as u_j and the activation of the output units as y_j

$$\{a^i\}^{(0)} = u^i \tag{3}$$

The outputs of the neurons in the last layer can be seen as the overall networks' outputs:

$$y^i = \{a^i\}^{(3)} \tag{4}$$

The activation function used in the artificial neuron is a critical element of a deep neural network. Without activation function, deep neural networks cannot learn nonlinear functions. Thus, the activation function provides a neural network some nonlinearities. There are many activation functions used in literature, but the most commonly used activation functions are illustrated in Figure 2.

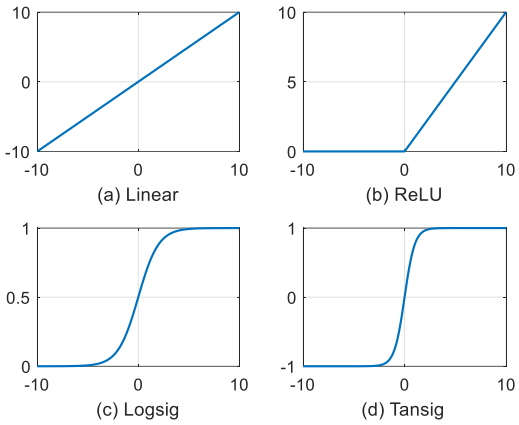


Figure 2 Most commonly used activation functions representation.

A two hidden layer deep neural network with linear transfer function at output layer is selected for this study. And, instead of matching the multiple outputs through a single network, multiple modules are used, each consisting of a suitable neural network characterizing just one aerodynamic coefficient. For regression, we use sum-of-squared errors as our measure of fit (objective function). The mean square error (MSE) method is used to calculate neural network loss, and MSE is given in Equation (5).

$$\mathrm{MSE}_{Train} = \frac{1}{n_{data}} \sum_{j=1}^{n_{data}} \left(z_j - y_j \right)^2 \tag{5}$$

j - The discrete data index
n_{data} - Number of data
z_j - Measured (actual) data in j^{th} discrete data index
y_j - Estimated data in j^{th} discrete data index

Another important concept of the neural network is used backpropagation algorithm. Among many backpropagation algorithms, the Levenberg-Marquardt backpropagation algorithm is chosen for this study since it is fast and gives better results [12]. It is a combination of Gradient Descent and Gauss-Newton optimization. Table 1 presents the hyperparameters of the Levenberg-Marquardt Algorithm.

Table 1 The hyperparameters of Levenberg-Marquardt algorithm.

#	Name	Symbol
1	Initial Dampening Factor	μ_0
2	Increase Dampening Factor	μ_{dec}
3	Decrease Dampening Factor	μ_{inc}
4	Maximum Dampening Factor	$\bar{\mu}$

1. Some Issues in Deep Neural Network Training

There are several important issues associated with the setup of the neural network, preprocessing, and initialization. Therefore, neural network training can be made more efficient if specific steps are performed for these issues. In this sub-section, several important issues are described [12] [13].

Scaling of the Inputs and Outputs (Feature Preprocessing)

Data scaling is recommended before applying the optimization methods [14]. This is found to be particularly the case while training feedforward neural networks. Here, scaling refers to arranging the values between the chosen lower and upper limits such that all of the variables have a similar order of magnitudes. All data is scaled to the range [0,1] or [-1,1] generally [12] [14] [15] [16]. All data is scaled to the range [-1,1] to reduce numerical error during neural network training as recommended by [12].

$$\boldsymbol{u}_n^i = -1 + \frac{2\left(\boldsymbol{u}^i - \boldsymbol{u}_{min}^i\right)}{\boldsymbol{u}_{max}^i - \boldsymbol{u}_{min}^i}$$
$$\boldsymbol{y}_n = -1 + \frac{2(\boldsymbol{y} - \boldsymbol{y}_{min})}{\boldsymbol{y}_{max} - \boldsymbol{y}_{min}} \tag{6}$$

where $(\quad)_{min}$, $(\quad)_{max}$, and $(\quad)_n$ are the minimum, maximum, and normalized values of the considered variable, respectively.

The initial weights

Initial weight assignment is one of the essential steps in neural network setup. The initial weights are generally set to small random numbers to avoid saturation in the neurons. It was observed that the algorithm does not work correctly if the initial weights are either zero or poorly chosen nonzero values [14]. The use of exact zero weights leads to zero derivatives and perfect symmetry, and the algorithm never moves. Starting instead with large weights often leads to inadequate solutions [17]. To avoid the saturation region of the activation function, the weighted sum output value becomes near to 0 as much as possible, which can increase the weight adjustment range. Therefore, the initial connection weights are usually random numbers between [-1,1] so that the network is not significantly affected [18].

In this study, initial weights and biases are set between random numbers between [-1,1] and scaled by scale factors between [0,1].

2. Feedforward deep neural network model validation

The performance of the neural network is an important criterion to assess the optimized hyperparameters. The available dataset is divided into training and testing datasets, firstly. The testing dataset, which is not seen in the identification phase, is used to assess the final performance at the end of the model identification. However, we should remark that the training dataset can be used in different ways to assess the fitted model using various resampling methods in the identification phase. e.g., train-validation split (hold-out cross-validation), k-fold cross-validation, or bootstrap approaches. It was shown that the k-fold cross-validation approach is superior to the hold-out cross-validation and bootstrap approaches [19]. For this study, the dataset is divided two-part; training (80%), testing (20%), and k-fold cross-validation approach is used to validate the neural network.

B. Hyperparameter Optimization Methods

The hyperparameters that define a deep neural network can be separated into two categories: the ones that define the architecture of the network and the ones that affect the optimization process of the training phase. The proposed method considers both categories at once.

Tuning the hyperparameters of a deep neural network is a critical process that was mainly done manually, relying on the previous experience of the experts (manual search). This is usually done using a trial-and-error process. However, even with expertise in deep neural networks and their hyperparameters, the best set of these hyperparameters changes with different data and is time-consuming. To address this problem, the ability to find the optimal hyperparameters in an automated manner is needed. Since genetic algorithm theory is a mature and widely used optimization routine, the automated hyperparameter selection in Deep Neural Network is made via a genetic algorithm.

1. Genetic Algorithm

Genetic algorithms are optimization methods that are inspired by biological evolution. Genetic algorithms operate on a population of candidate solutions and apply the principle of survival of the fittest to evolve the candidate solutions towards the desired optimal solutions. In a genetic algorithm, candidate solutions are referred to as individuals or parameters. A population refers to a group of individuals. There are six phases in a genetic algorithm: initial population, fitness evaluation, selection, crossover, mutation, and elitism. The reader is referred to Refs. [20] [21] [22] [23] for detail description of genetic algorithms. Figure 3 illustrates the steps of a typical genetic algorithm.

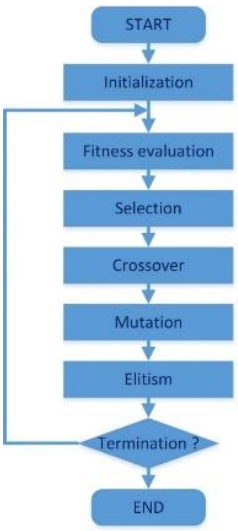


Figure 3 Typical Genetic Algorithm flowchart.

C. Hybridization of Genetic Algorithm with Deep Neural Network

Figure 4 shows a deep neural network flowchart. The measured output and predicted output are compared, and an objective function is measured to evaluate the neural network's performance. Weights and biases are updated using a solver (optimization) algorithm to minimize the objective function. Neural Network architecture/solver hyperparameters and initial weights/biases must be decided before the optimization routine starts.

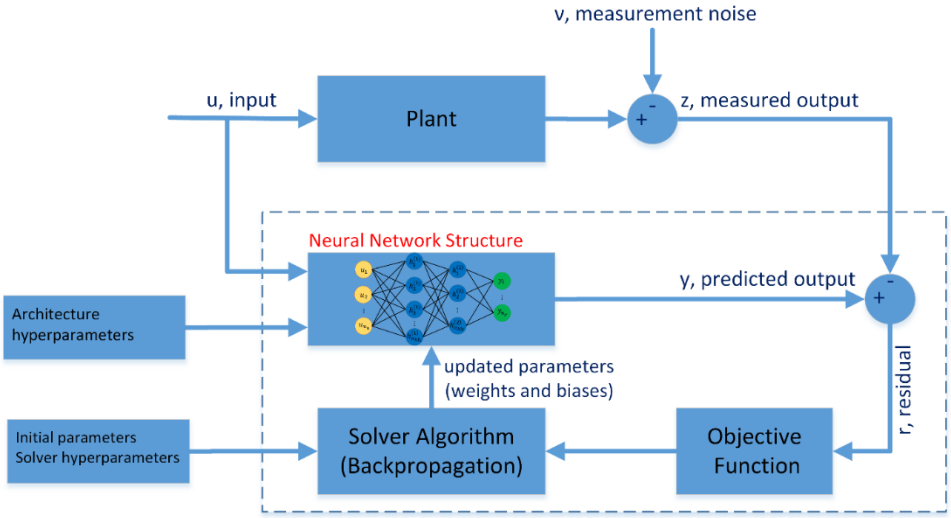


Figure 4 Deep Neural Network flowchart.

Table 2 and Table 3 summarize the hyperparameters responsible for defining the structure of the network.

Table 2 User-defined neural network hyperparameters.

#	Hyperparameter	Ranges and Functions
1	Architecture	2 hidden layer feed forward neural network
2	Fitness function	Training dataset MSE
3	Solver	Levenberg-Marquardt Optimization
4	Epoch	300
5	Maximum dampening factor for solver, $\bar{\mu}$	10000

Table 3 Deep Neural Network hyperparameters tuned by Genetic Algorithm.

#	Hyperparameter	Ranges and Functions
1	Number of neurons in the first hidden layer	{1,2,…,50}
2	Activation function for the first hidden layer	{Logsig, Tansig, ReLU, Linear}
3	Number of neurons in the second hidden layer	{1,2,…,50}
4	Activation function for the second hidden layer	{Logsig, Tansig, ReLU, Linear}
5	Weights and biases initization factor	[0,1]
6	Initial Dampening factor for solver, μ_0	{0.01, 0.02, …, 0.1}
7	Increase factor for solver, μ_{inc}	{2, 3, …, 10}
8	Increase factor for solver, μ_{dec}	$\{1/2, 1/3, \ldots, 1/10\}$

D. Multi-Fidelity Deep Neural-Genetic Network Algorithm Implementation

The implementation of the Multi-Fidelity DNGN algorithm is as follows:

1. A low-fidelity DNGN model is build using a low-fidelity dataset.

$$f(u_{LF}) \approx f_{LF}(u_{LF}) \tag{7}$$

2. Predict low-fidelity values for each data point in the high-fidelity dataset. This step is necessary because the given dataset probably is not taken in the same aerodynamic condition.

$$f(u_{HF}) \approx f_{LF}(u_{HF}) \tag{8}$$

3. The increments or differences are calculated between low-fidelity data at each high-fidelity data point, which is calculated in step-2 and high-fidelity data.

$$\beta(u_{HF}) = f_{HF}(u_{HF}) - f_{LF}(u_{HF}) \tag{9}$$

4. The increment function is calculated by using the set of increments or differences data which are found in step-3 using the DNGN approach.
5. Data fusion function is a simple summation of low-fidelity Kriging and increment Kriging model for the desired data point.

$$f(u) \approx f_{LF}(u) + \beta(u) \tag{10}$$

When the broad range of aircraft flight envelope is considered, these aerodynamics coefficients have nonlinear relationships with their dependent variables. The two separate fully-connected neural networks can be employed to approximate the linear and nonlinear part of increment function for the low- and high-fidelity data [24]. But to reduce the computational effort, the nonlinearity in the increment function is expressed using spline functions. Spline functions are defined only on the subintervals and can approximate nonlinearities quite well. The angle of attack range is divided and added as dependencies to neural network inputs for improving performances. Parameters dependent on the angle of attack in longitudinal coefficients are expressed using spline functions [25] in the form

$$(\alpha - \alpha_i)_+^m = \begin{cases} (\alpha - \alpha_i)^m & \alpha \geq \alpha_i \\ 0 & \alpha < \alpha_i \end{cases} \tag{11}$$

III.Results and Discussions

A. Data Preparation and Evaluation

In this study, a two-dimensional aerodynamic force data fusion problem was considered. Digital Datcom constructs the aerodynamic database of F-16 aircraft to function the angle of attack and the horizontal tail deflection. Moreover, wind tunnel tests are adapted to improve the database. The former represents the low-fidelity dataset, while the latter does the high-fidelity dataset. The wind tunnel results are obtained from the open-source F-16 aircraft data [26]. Digital

Datcom includes empirical methods for aerodynamic data predictions. Thus, it enables fast and reasonably accurate computations in conceptual design phases.

The primary geometry of F16 is obtained from OpenVSP-Hangar, as shown in Figure 2 [27]. The fuselage geometry is prepared from this geometry while the wing, horizontal tail, and vertical tail geometries are modified with respect to a supersonic wind tunnel test campaign [28]. The model is on a 1/15 scale.

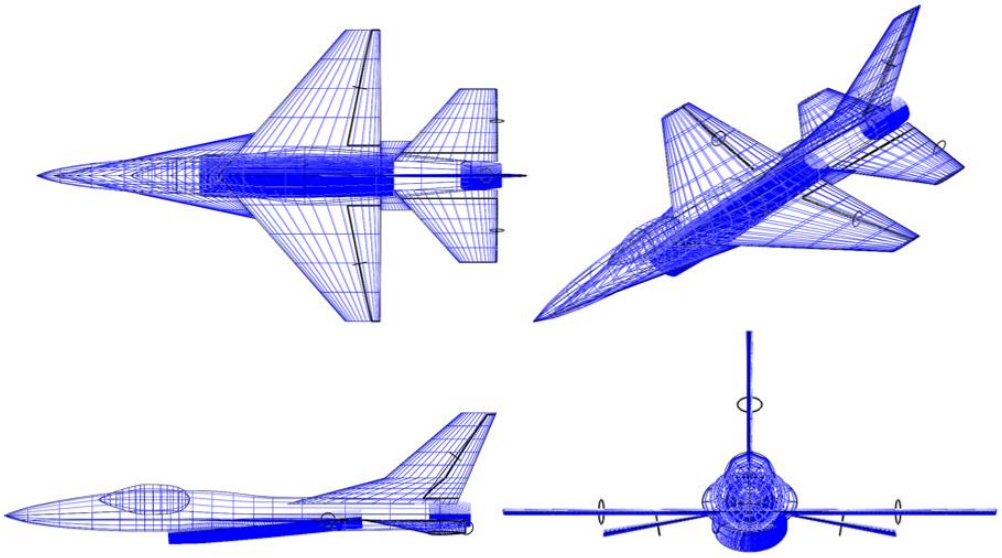


Figure 5. Top, isometric, bottom, and left-side view of F16-OpenVSP geometry

The aerodynamic database is created for clean aircraft and horizontal tail deflections. The angle of attack ranges between 0° and 30° while the Mach number is set to 0.2 at sea level flight condition. For the non-dimensional coefficients, the mean aerodynamic chord is selected as 3.45 *m* while the span is set to 9.15 *m*. Additionally, the wing area is set to 27.86 m^2.

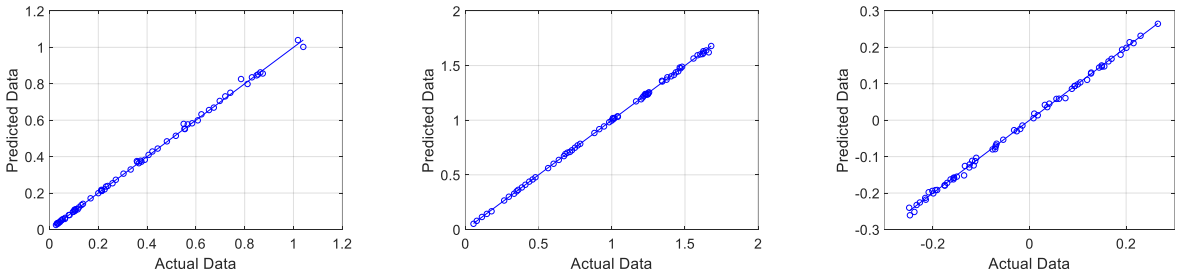
There were 775 and 341 points at the low- and high-fidelity levels, respectively. The data encompassed α and δ_{ht} values in the ranges:

$$\begin{aligned} \alpha_{LF} &= \{0°, 1°, 2°, \cdots, 30°\} \\ \delta_{ht_{LF}} &= \{-25°, -22°, -20°, -18°, -16°, -14°, -12°, -10°, -8°, -6°, -4°, -2°, 0°, 2°, \cdots, 22°, 25°\} \\ \alpha_{HF} &= \{0°, 1°, 2°, \cdots, 30°\} \\ \delta_{ht_{LF}} &= \{-25°, -20°, -15°, -10°, -5°, 0°, 5°, 10°, 15°, 20°, 25°\} \end{aligned} \tag{12}$$

Drag, lift and pitch moment coefficients of F-16 aircraft geometry for low- and high-fidelity datasets are compared in the prior work [4]. Since the main idea is to improve the data quality of a low-fidelity model, these comparisons are helpful to observe differences between low- and high-fidelity models. Lift and drag coefficients of the low-fidelity dataset are much more correlated than the pitch moment coefficient for the high-fidelity dataset.

B. Multi-Fidelity Deep Neural-Genetic Network Results

The qualitative accuracy spreads of the testing samples of the longitudinal coefficients are shown in Figure 6. All points are clustered along the equality line; the predicted values are close to the true value, which shows that the proposed method shows good performance in the testing data sets.



a) C_D b) C_L c) C_m

Figure 6 Accuracy spread of the high-fidelity testing points for aerodynamic coefficients.

The GA has its own set of hyperparameters. The choice of genetic algorithm hyperparameters should be carried out carefully. Any wrong choice produces meaningless results. The hyperparameters are defined with respect to Ref. [29]. The hyperparameters for the genetic algorithm that is set in this application are provided in Table 4.

Table 4 Genetic Algorithm hyperparameters.

#	Hyperparameter	Value and Description
1	Number of population	50
2	Number of generation	20
3	Fitness function	k-fold CV estimate of DNN
4	Selection method	Roulette wheel
5	Crossover operator	Crossover operator
6	Crossover fraction	0.8
7	Mutation operator	Gaussian
8	Mutation fraction	0.05
9	Elite count	2
10	Data used	Training dataset

In this case, there were two hidden layers used within the low-fidelity neural network, while two hidden layers were used within increment function neural networks, respectively. The training time of the model was closely related to the number of training data and the number of epochs. The training MSE of longitudinal coefficients already reached steady-state values at around 150 epochs, before which the weights of the network were rapidly tuned to optimize the prediction model. Hence, the maximal number of epochs was set to 250 for the proposed method to reduce the training time.

It is observed that the deep neural network is not sensitive to the LM solver hyperparameters because the relative standard deviation, which is the standard deviation over the mean of the hyperparameters, is higher than one. That's why we can use the average results for these hyperparameters, which are tabulated in Table 5. The optimum hyperparameters yielding the minimum MSE over 20 runs of the proposed method are provided in Tables 6-8.

Table 5 Proposed Levenberg-Marquardt solver hyperparameters

#	Hyperparameter	Proposed Value
1	Initial Dampening factor for the solver, μ_0	0.46
2	Increase factor for solver, μ_{inc}	6.00
3	Increase factor for solver, μ_{dec}	0.49

Table 6 Tuned Deep Neural Network hyperparameters for the lift coefficient.

#	Hyperparameter	Low-Fidelity Neural Network	Increment Neural Network
		Value	Value
1	Number of neurons in the first hidden layer	15	8
2	Activation function for the first hidden layer	Tansig	Tansig
3	Number of neurons in the second hidden layer	9	13
4	Activation function for the second hidden layer	Tansig	Tansig
5	Bias initialization factor	0.60	0.80
6	Initial weight scale factor	0.40	0.40
7	The first hidden layer weight scale factor	0.80	0.70
8	The second hidden layer weight scale factor	0.30	1.00

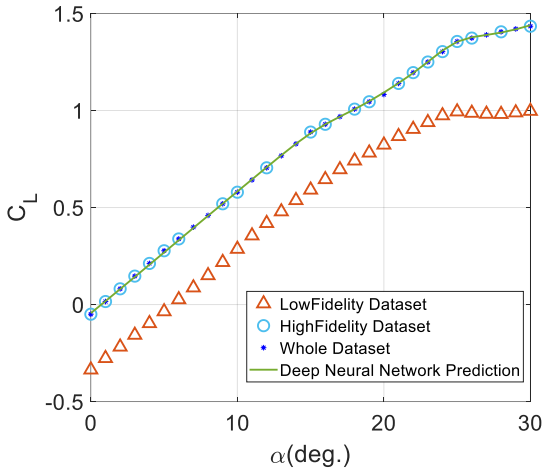
Table 7 Tuned Deep Neural Network hyperparameters for the drag coefficient.

#	Hyperparameter	Low-Fidelity Neural Network	Increment Neural Network
		Value	Value
1	Number of neurons in the first hidden layer	9	6
2	Activation function for the first hidden layer	Logsig	Tansig
3	Number of neurons in the second hidden layer	12	6
4	Activation function for the second hidden layer	Logsig	Logsig
5	Bias initialization factor	0.70	0.40
6	Initial weight scale factor	0.40	0.80
7	The first hidden layer weight scale factor	0.90	0.40
8	The second hidden layer weight scale factor	0.60	0.40

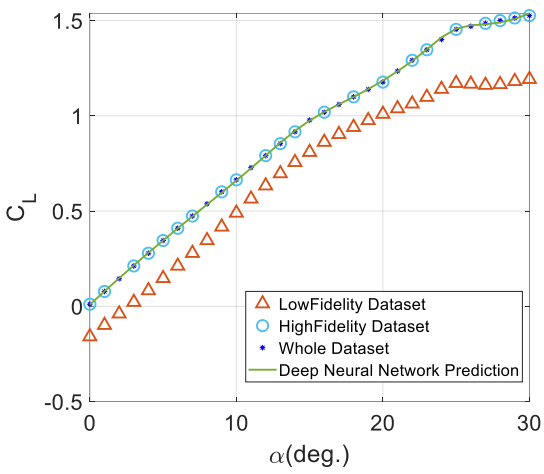
Table 8 Tuned Deep Neural Network hyperparameters for the pitch moment coefficient.

#	Hyperparameter	Low-Fidelity Neural Network	Increment Neural Network
		Value	Value
1	Number of neurons in the first hidden layer	13	14
2	Activation function for the first hidden layer	Tansig	Logsig
3	Number of neurons in the second hidden layer	12	10
4	Activation function for the second hidden layer	Logsig	Tansig
5	Bias initialization factor	0.30	0.60
6	Initial weight scale factor	0.80	1.00
7	The first hidden layer weight scale factor	0.10	0.70
8	The second hidden layer weight scale factor	0.30	0.70

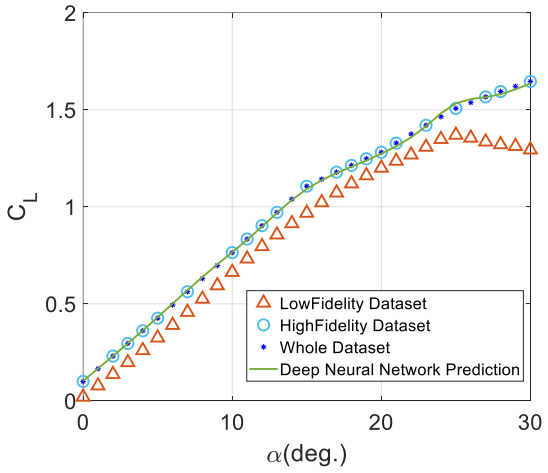
Figure 7 to Figure 9 shows that the predicted trends of longitudinal characteristics obtained with the DNGN method are shown with a true low- and high-fidelity dataset. C_L vs α, C_D vs α , and C_m vs α plot for Mach 0.2 and $-20°, -10°, 0°, 10°, 20°$ horizontal tail deflections are given in Figure 7 to Figure 9 respectively.



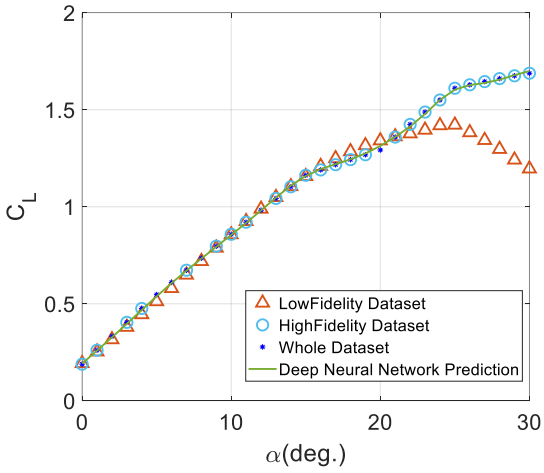
(a) $\delta_{ht} = -20°$



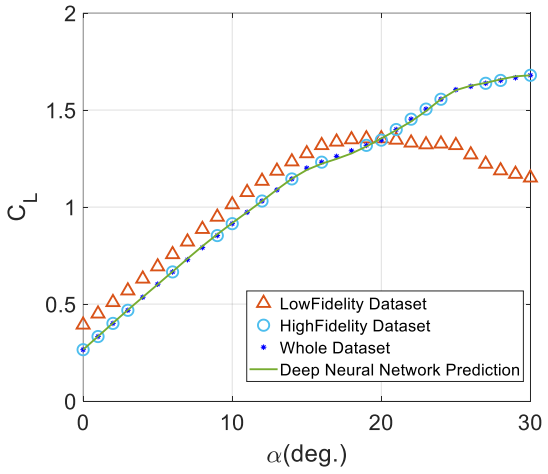
(b) $\delta_{ht} = -10°$



(c) $\delta_{ht} = 0°$

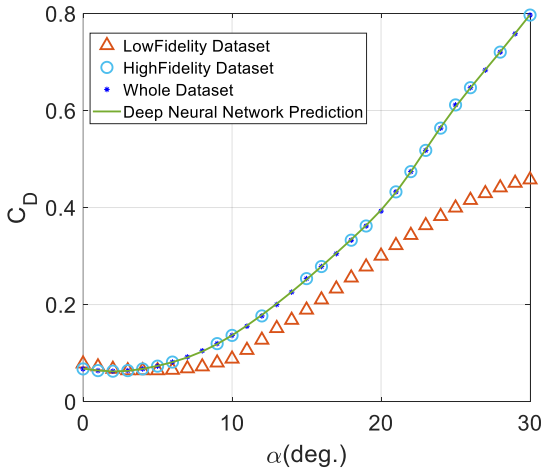


(d) $\delta_{ht} = 10°$

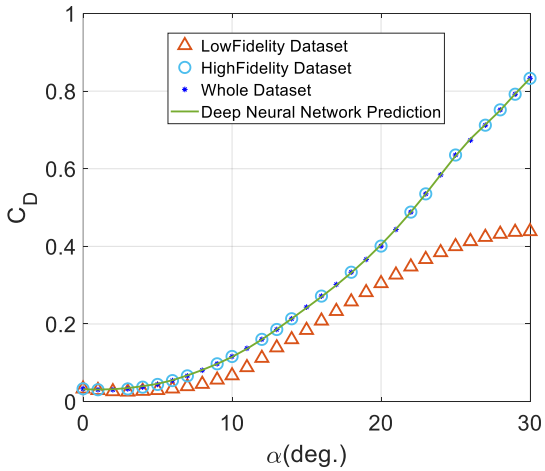


(e) $\delta_{ht} = 20°$

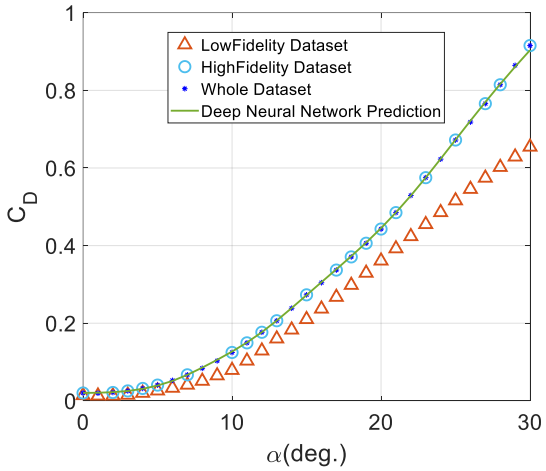
Figure 7 The predicted trends of C_L coefficient with the DNGN method.



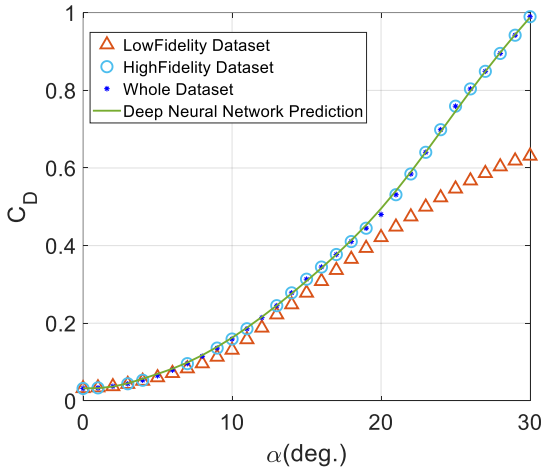
(a) $\delta_{ht} = -20°$



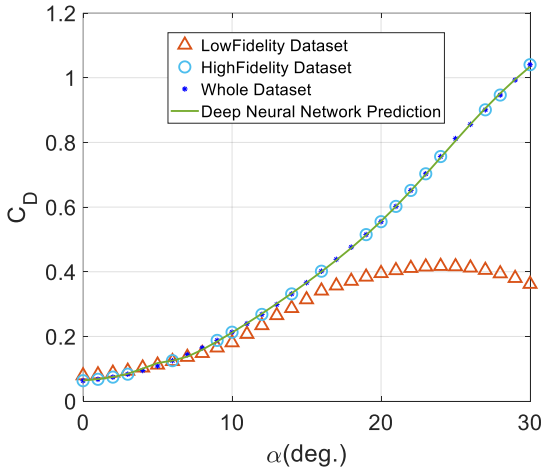
(b) $\delta_{ht} = -10°$



(c) $\delta_{ht} = 0°$

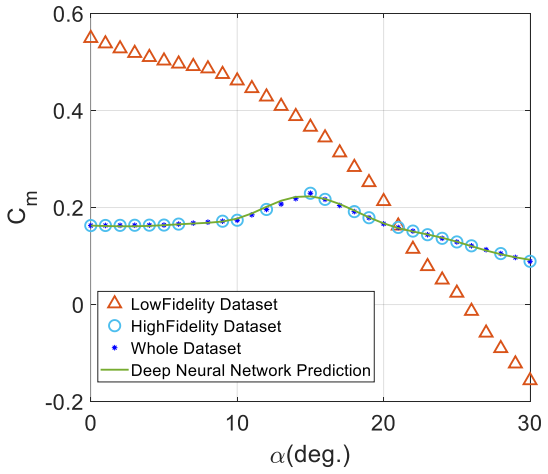


(d) $\delta_{ht} = 10°$

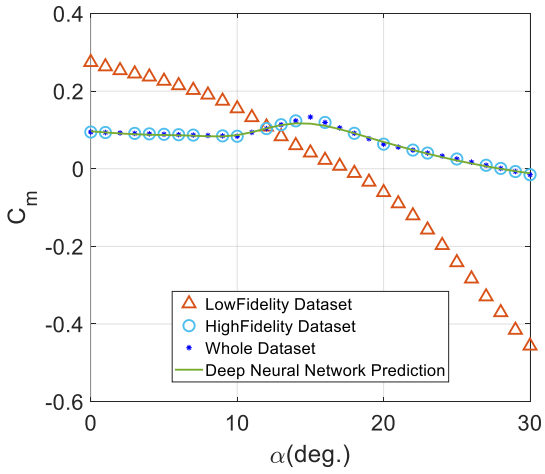


(e) $\delta_{ht} = 20°$

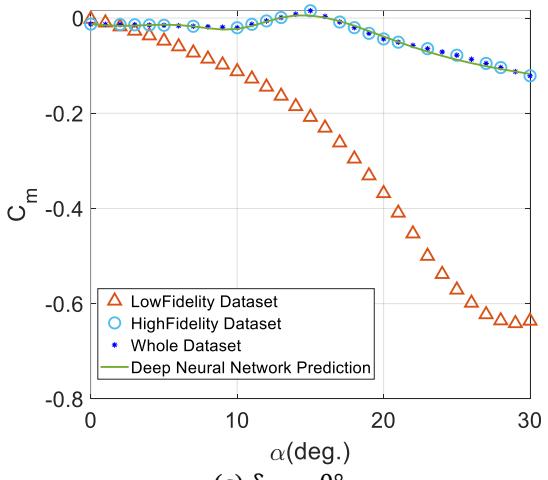
Figure 8 The predicted trends of C_D coefficient with the DNGN method.



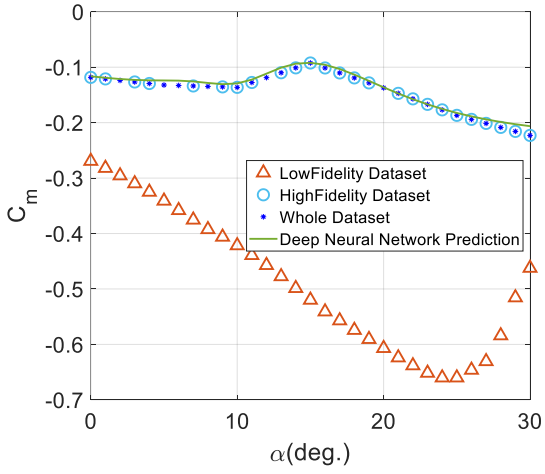
(a) $\delta_{ht} = -20°$



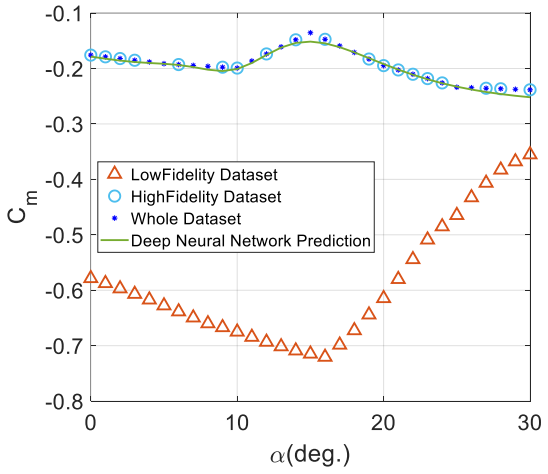
(b) $\delta_{ht} = -10°$



(c) $\delta_{ht} = 0°$



(d) $\delta_{ht} = 10°$



(e) $\delta_{ht} = 20°$

Figure 9 The predicted trend of C_m coefficient with the DNGN method.

IV. Conclusion

A simple F16 fighter aircraft geometry is obtained from publicly open resources, and its geometric details are corrected. Datcom input file is created, and the subsonic aerodynamic database of F16 is generated for the baseline and deflected horizontal tail configurations. This database is then assigned as the low-fidelity dataset. On the other hand, subsonic wind tunnel data of F16 is obtained as the high-fidelity data. The differences in low and high-fidelity datasets are clarified: similar trends are observed for the lift and drag coefficients, although some sorts of variable shifts exist between these datasets. However, the trends in pitch moment are not compatible with the lift and drag coefficients in the similarity of the low- and high-fidelity datasets.

The Deep Neural-Genetic Network approach is used to derive high-fidelity datasets using more low-fidelity datasets and less high-fidelity datasets to reduce the computational cost of high-fidelity dataset generation. The proposed method yielded promising results, particularly for the drag and lift coefficients.

In the Deep Neural-Genetic Network approach, the deep neural network hyperparameters, including the number of neurons in each layer, activation function for each layer, biases and weights scale factors, and Levenberg-Marquardt solver hyperparameters, were optimized by a Genetic Algorithm. It is observed that different choices of Levenberg-Marquardt hyperparameters are seen to give essentially the same performance because this algorithm adjusts the damping at each iteration. It is also observed that the number of neurons in each layer and activation function for each layer are the most dominant hyperparameters affecting deep neural network performance.

Experiment design is the critical factor for surrogate prediction accuracy. In future work, different design of experiments approaches will be implemented.

V.References

[1] B. Peerlings, "A review of aerodynamic flow models, solution methods and solvers and their applicability to aircraft conceptual design," Delft University of Technology, Delft, 2018.

[2] M. Tyan, M. Kim, V. Pham, C. Choi, L. Nguyen and J. W. Lee, "Development of Advanced Aerodynamic Data Fusion Techniques for Flight Simulation Database Construction," in *AIAA Aviation Forum*, Atlanta, GA, 2018.

[3] Q. Liu, P. Zhang, W. Xiao and J. D. Diao, "Correction Methods of Aerodynamic Force and Moment Coefficients Based on the Identification Data," in *Chinese Control Conference*, Guangzhou, China, 2019.

[4] H. B. Kurt, M. Millidere, F. S. Gömeç and Ö. Uğur, "Multi-fidelity Aerodynamic Dataset Generation of a Fighter Aircraft," in *AIAA SciTech Forum*, Virtual Event, 2021.

[5] C. Tang, K. Gee and S. Lawrence, "Generation of Aerodynamic Data Using a Design of Experiment and Data Fusion Approach," in *AIAA Aerospace Science Meeting and Exhibit*, Reno, NV, 2005.

[6] A. Forrester, A. Sobester and A. Keane, Engineering Design via Surrogate Modelling: A Practical Guide, A John Wiley and Sons Inc, 2008.

[7] A. Forrester, A. Sobester and A. Keane, "Multi-fidelity optimization via surrogate modelling," *Proceedings of the Royal Society: A Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251-3269, 2007.

[8] X. Meng and G. E. Karniadakis, "A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems.," *J. Comput. Phys.*, 2020.

[9] K. K. Horni, M. Stinchcombe and H. White, "Multi-layer feed forward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989.

[10] H. S. Das and P. Roy, in *A Deep Dive Into Deep Learning Techniques for Solving Spoken Language Identification Problems*.

[11] A. S. Tenney, M. N. Glauser and Z. P. Berger, "Application of Artificial Neural Networks to Stochastic Estimation and Jet Noise Modeling," *AIAA Journal*, 2010 ???.

[12] H. Demuth and M. Beale, "Neural Network Toolbox," The MathWorks, Natick,MA, 2004.

[13] C. C. Aggarwal, Neural Networks and Deep Learning, Yorktown Heights, NY: Springer, 2018.

[14] R. V. Jategaonkar, Flight Vehicle System Identificiation: A Time-Domain Methodology, Reston, VA: AIAA Process in Astronautics and Aeronautics, AIAA, 2015.

[15] T.-A. Nguyen, H.-B. Ly and B. T. Pham, "Backpropagation Neural Network-Based Machine Learning Model for Prediction of Soil Friction Angle," *Hindawi Mathematical Problems in Engineering*, 2020.

[16] I. H. Witten, E. Frank and M. A. Hall, "Data Management Systems," in *Chapter 7- data transformations*, Burlington, MA, Morgan Kaufmann, 2011, pp. 305-349.

[17] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Stanford, CA: Springer, 2008.

[18] T. Tan, Z. Yang, F. Chang and K. Zhao, "Prediction of the FirstWeighting from the Working Face Roof in a Coal Mine Based on a GA-BP Neural Network," *Appl. Sci.*, 2019.

[19] M. Millidere, H. B. Kurt, H. Ballı and Ö. Uğur, "Kalman Based Neural Network Analysis with Resampling Methods for Longitudinal Aerodynamic Coefficient Estimation," in *AIAA Aviation Forum*, Virtual Event, 2020.

[20] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Boston, MA: Addison Wesley Longman Publishing Co., Inc., 1989.

[21] S. N. Sivanandam and S. N. Deepa, Introduction to Genetic Algorithms, Berlin, Heidelberg: Springer, 2008.

[22] A. Chipperfield, P. Fleming, H. Pohlheim and C. Fonseca, "Genetic Algorithm Toolbox User's Guide," University of Sheffield, Sheffield, 2007.

[23] S. D. Sudhoff, "Genetic Optimization System Engineering Tool (GOSET) For Use with MATLAB," Purdue University, Purdue, 2007.

[24] L. He, W. Qian, T. Zhao and Q. Wang, "Multi-Fidelity Aerodynamic Data Fusion with a Deep Neural Network Modeling Method," *Entropy*, 2020.

[25] V. Klein and E. A. Morelli, Aircraft System Identification: Theory and Practices, Reston, Va: AIAA Education Series, 2006.

[26] L. Nguyen, M. Ogburn, W. P. Gilbert, K. S. Kibler, P. W. Brown and P. L. Deal, "Simulator Study of Stall/PostStall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability," NASA Technical Paper 1538, Hampton, VA, 1979.

[27] William2002730. [Online]. Available: hangar.openvsp.org/vspfiles/343. [Accessed 05 06 2020].

[28] M. Fox and D. Forrest, "Supersonic Aerodynamic Characteristics of an Advanced F-16 Derivative Aircraft Configuration," NASA Technical Paper 3355, 1993.

[29] K. Abhary, S. D. Dao and R. Marian, "Maximising Performance of Genetic Algorithm Solver in Matlab," *Engineering Letters*, 2016.