Hindawi
International Journal of Aerospace Engineering
Volume 2022, Article ID 2918458, 17 pages
https://doi.org/10.1155/2022/2918458

Research Article

Robust Data-Driven Fault Detection: An Application to Aircraft Air Data Sensors

Yunmei Zhao,[1,2] Hang Zhao,[1] Jianliang Ai,[1] and Yiqun Dong[1]

[1]*Department of Aeronautics and Astronautics, Fudan University, Shanghai 200433, China*
[2]*School of Aerospace Engineering and Applied Mechanics, Tongji University, Shanghai 200092, China*

Correspondence should be addressed to Yiqun Dong; yiqundong@fudan.edu.cn

Received 1 November 2021; Revised 17 January 2022; Accepted 21 January 2022; Published 16 March 2022

Academic Editor: Guillermo Valencia-Palomo

Fault detection (FD) is important for health monitoring and safe operation of dynamical systems. Previous studies use model-based approaches which are sensitive to system specifics, attenuating the robustness. Data-driven methods have claimed accurate performances which scale well to different cases, but the algorithmic structures and enclosed operations are "black," jeopardizing its robustness. To address these issues, exemplifying the FD problem of aircraft air data sensors, we explore to develop a robust (accurate, scalable, explainable, and interpretable) FD scheme using a typical data-driven method, i.e., deep neural networks (DNN). To guarantee the scalability, aircraft inertial reference unit measurements are adopted as equivalent inputs to the DNN, and a database associated with 6 different aircraft/flight conditions is constructed. Convolutional neural networks (CNN) and long-short time memory (LSTM) blocks are used in the DNN scheme for accurate FD performances. To enhance robustness of the DNN, we also develop two new concepts: "large structure" which corresponds to the parameters that can be objectively optimized (e.g., CNN kernel size) via certain metrics (e.g., accuracy) and "small structure" that conveys subjective understanding of humans (e.g., class activation mapping in CNN) within a certain context (e.g., object detection). We illustrate the optimization process we adopted in devising the DNN large structure, which yields accurate (90%) and scalable (24 diverse cases) performances. We also interpret the DNN small structure via class activation mapping, which yields promising results and solidifies the robustness of DNN. Lessons and experiences we learned are also summarized in the paper, which we believe is instructive for addressing the FD problems in other similar fields.

1. Introduction

1.1. Motivation. Fault detection (FD) is important for safe operations of dynamical systems. For instance, aircraft air data sensors (ADS) provide measurements of aircraft's airspeed, angle of attack (AOA), and sideslip angle. The erroneous sensor measurements, however, were found to be the cause of many catastrophic flight accidents including the crashes of NASA X-31 [1], Airbus A330 [2], and most recently Boeing B737 MAX [3]. A robust fault detection strategy is imminent for the health monitoring of commercial airlines.

At present, hardware redundancy (HR) is widely used for FD problems. Particularly for ADS in the commercial airlines, HR consists of installing multiple sensors to produce redundant measurements of the air data. Outputs from all the sensors are continuously monitored by a voting logic, which detects (and isolates) the erroneous sensor. The correct measurement is then reported using the remaining other sensors [4–6].

One issue with the HR-based fault detection is the cost and weight penalty (due to the redundant sensors). Moreover, recent accidents indicate that HR is not sufficient in addressing the fault detection problem (e.g., the Boeing 737MAX accident due to AOA sensors). Alternative to HR, analytical redundancy (AR) has been investigated. A majority of the AR methods adopted model-based approaches. Different from HR, AR investigates each sensor separately. For a certain sensor, a mathematical model is developed in conjunction with other sensors. An inferred sensor measurement is then estimated and compared with the sensor's output to generate a residual. If the residual exceeds a

predefined bound, fault is claimed to be detected for that sensor [7].

The model-based AR, nevertheless, hinges on the model that is derived from system specifics, which is sensitive to operational conditions. The development of model-based AR typically requires ad hoc parameter tuning, which is time-consuming. Another line of AR adopts model-free and mostly data-driven methods. This does not require system specifics, but the recorded data (e.g., sensors measurements and the associated faults) only. In particular, deep neural networks (DNN) were widely used [8–14]. However, no explainable rules exist for the architecture devising in DNN, and most works adopted the trial-and-error methodology; mathematical operations enclosed within the DNN are also considered "black," and scalability of the DNN-based FD scheme is doubtful. Referring to the development of DNN in other fields (e.g., computer vision), many rules have been proposed (and widely accepted) in devising the DNN architectures (e.g., how many CNN filters should be used in each layer). Ablation studies [15] and DNN visualization (e.g. class activation mapping [16]) are widely adopted, which greatly elevate the physical understanding (robustness) of the DNN. Similar concepts/approaches may also be used in analyzing the DNN-based FD scheme.

We thus summarize the motivation of this paper: Though DNN-based methods have yielded accurate and scalable FD performances, the weaknesses are as follows: explainability in devising the DNN architecture and interpretability of the DNN operations. Exemplifying the FD problem of aircraft air data sensors, we propose to develop a robust DNN-based FD scheme. Whilst the FD accuracy and scalability must be guaranteed, we also explain the rules in devising the DNN architecture and interpret the operations enclosed in the DNN structure.

1.2. Related Work

1.2.1. Model-Based FD. Model-based FD hinges on a mathematical model to infer the sensor measurement, which being further used to generate a residual. A plethora of works were found to use Kalman filtering (KF), e.g., the extended KF [17–19], the unscented KF [20], the theoretical analysis of an adaptive three-step KF [21], and implementation of the KF-based method to real data in [22]. Other KF-based works included [23, 24], wherein fuzzy logic was used in conjunction with KF to consolidate the sensor data [23], and the hidden Markov model has been used to decide the sensors state (fault/healthy) based on the KF outputs in [24]. KF-based FD schemes, however, rely on the evolution model that is derived from the system dynamics/kinematics; ad hoc parameter tuning is imminent in adjusting KF to different systems (e.g., different aircraft) or operational cases (e.g., different flight conditions).

Other model-based FD methods adopted robust control theory in [25–28], wherein the robustness synthesis-based filter was constructed to output the residual, but a sensor state evolution model is needed, and no rules pertaining to the parameters tuning was studied. In [29–31], moving horizon estimator was developed. It compensated for both sensor faults and wind speed estimation in the fault tolerant control. However, the authors discussed limited aircraft/flight cases in the paper; scalability of the proposed methods is unclear. A scheme designed particularly for systems with two time-scale dynamics (e.g., phugoid and short periods in the aircraft's longitudinal plane) was discussed in [32, 33], wherein both nonlinear geometric approach and singular perturbation technique were involved, but computational load of the algorithm was relatively high, and parameter tuning was time-consuming. Barrier function-based learning observer was proposed in [34], and in [35], a set-value observer (SVO) was used. As acclaimed in the papers, these works significantly decreased the FD false alarm rate. The weaknesses of [34, 35], however, are also typical: model sensitiveness and unclear scalability.

1.2.2. Data-Driven FD. Different data-driven methods have been proposed for the FD problem. In [36], the fuzzy inference system in conjunction with thresholder was proposed for the FD of DC motors. In [37], 4 different Wiener models were ensembled for the fault analysis of an industrial gas turbine. Dynamical primary components analysis was used together with support vector machine in [38] for the FD problem of gear box. And finally, in [39], a total of 5 state-of-the-art algorithms were studied for FD of marine machinery. Although these methods have yielded promising results, however, the weakness is also obvious: heavy parameters tuning is usually needed in finalizing the algorithm structures.

Other data-driven schemes were found to use neural networks (NN). In [8, 9], fully connected cascaded NN was adopted, and the authors discussed fault detection and isolation for inertial reference unit (IRU). Similar works were found in [12, 40, 41], wherein feed-forward NN was used. In [42, 43], NN-based adaptive observer was developed to generate the sensor measurement residual; parameters of the NN were adjusted online via KF [42]. Also, in [10–12], NN was used to establish nonlinear identification models, which being used as a state observer to generate the residual. The essence of all these works was to regress a functional relation that maps from the designated input to the desired output (i.e., fault cases), but traditional NN lacks the efficiency in abstracting high-level features. It is usually used in a hybrid form with other methods (e.g., KF). In addition, no research pertaining to the explainability and interpretability analysis was thoroughly illustrated in the associated publications.

Recent NN developments advocate the deep neural networks (DNN) in many academic/industrial fields [44]. DNN typically has more ("deeper") layers which are activated using designated function (e.g., ReLU). More dedicated operations were also designed in convolutional neural network (CNN) and long-short time memory (LSTM) blocks for extracting both spatial and temporal features enclosed in the DNN input. Early works along the DNN-based FD line were found in [45–47], wherein recurrent neural network (RNN) was used. Later works adopted a variant of RNN, i.e., LSTM, which attenuates the error vanishment/explosion problems in the traditional RNN. CNN was also widely used either independently [48, 49] or in conjunction with LSTM; new data

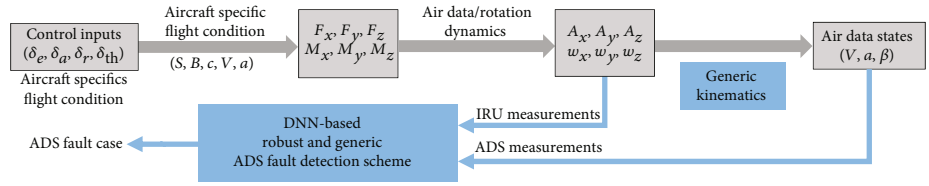

Figure 1: Aircraft motion equations and the fault detection scheme formulated in this paper.

Table 1: Different aircraft and flight conditions adopted in this paper.

Aircraft	General configuration	Weight	Span	Data source	Flight condition
Y	Large cargo airplane, quadruple piston engines, high wing	41.0 t	38.0 m	Simulation	(i) Low altitude, LTO, manual
B_1	Large passenger aircraft, quadruple turbo engines, low wing	174 t	59.6 m	Simulation	(i) High altitude, cruise, AP (ii) Low altitude, free flight, manual
B_2	Large passenger aircraft, double turbo engines, low wing	44.6 t	35.8 m	Real flight	(i) Low altitude, LTO, manual
D	General aviation, double piston engines, high wing	3.12 t	19.8 m	Simulation	(i) High altitude, cruise, AP
F	Fighter aircraft, double turbo engines, delta wing	10.5 t	11.4 m	Real flight	(i) Medium altitude, manual flight

formats defined as "state image" and "control image" were proposed in [50, 51], via which the sensor FD accuracy was significantly improved. The CNN-LSTM fusion-based DNN architecture has claimed promising results in [51] for air data sensors FD and most recently in [52] for fault estimation. Despite the rapid developments of various DNN-based FD architectures, however, research efforts along the explainability and interpretability analysis line are still rare.

1.2.3. Explainability/Interpretability Analysis of DNN Large/Small Structures. DNN is commonly considered "black": why the DNN specifics are devised as such and how the enclosed operations work. To address such issue, we propose the following: (1) to explain the large structure that corresponds to the DNN architecture specifics and (2) to interpret the small structure that depicts the operations enclosed in the DNN architecture. Similar works have appeared in literature.

The DNN large structure corresponds to the specifics (e.g., CNN kernel size) that can be objectively optimized via certain metrics (e.g., DNN testing accuracy). To explain the large structure, comparative studies were commonly used. In [53], different sets of parameters (number of CNN filters, kernel sizes, etc.) were assembled in the DNN. The authors then performed thorough training for each parameter set and decided the optimal one via gleaning the trained DNN. Technical tools designed specifically for optimizing the DNN training were also found, of which the most peculiar one is the Microsoft's NNI, which decides the best training coefficients (e.g., learning rate and iterative epochs) for a certain DNN architecture [54]. The Tree-structured Parzen Estimator (TPE) is a sequential model-based optimization (SMBO) approach, as a black-box optimization, which can be used in various scenarios and shows good performance, especially when limited by computation resources and can only try a small number of trials [55]. When an "optimal" DNN is found, ablation study is commonly used to verify the architecture (e.g., CNN branches and CNN-LSTM fusions), which involves cropping certain subarchitecture from the "optimal" one, and comparing the DNN performances. Typical examples are found in [15].

The DNN small structure depicts the operations enclosed within the DNN (e.g., a certain CNN filter). It is usually analyzed via mirroring the DNN outputs to what humans understand in a certain context. For instance, in the visual object classification problem, CNN is commonly used. The understandable terms of humans in such a context are the visual features that one hinges on to classify an object (e.g., "ear" or "nose" of a cat/dog). Class activation mapping (CAM) thus was proposed in [16] and rapidly developed in [56–61], which points out the highlighted region(s) wherein the CNN filters focus on. The CNN architecture may be considered reasonable (interpretable) if the highlighted region(s) corresponds to what humans tend to watch (e.g., the "ear"/"nose"). Related studies in such line have made promising progresses which promoted both academic researches and industrial applications of CNN in vision-related problems—but "vision-related" only; very rare studies pertaining to such line were found in other DNN-based works (e.g., DNN-based fault detection).

1.3. Overview of This Paper. In this paper, exemplifying the FD problem of aircraft air data sensors, we aim to develop a robust (accurate, scalable, explainable, and interpretable) DNN-based fault detection scheme. We highlight our contributions as follows:

(i) Accurate and scalable FD performances: we model the FD problem using aircraft IRU measurements as equivalent inputs; we also construct a dedicated dataset; via delicate architecture tuning, the DNN-based scheme claims accurate and scalable FD performances for different aircraft at various conditions

(ii) Explainable DNN large structure: we exhibit the methodology in devising the DNN architecture

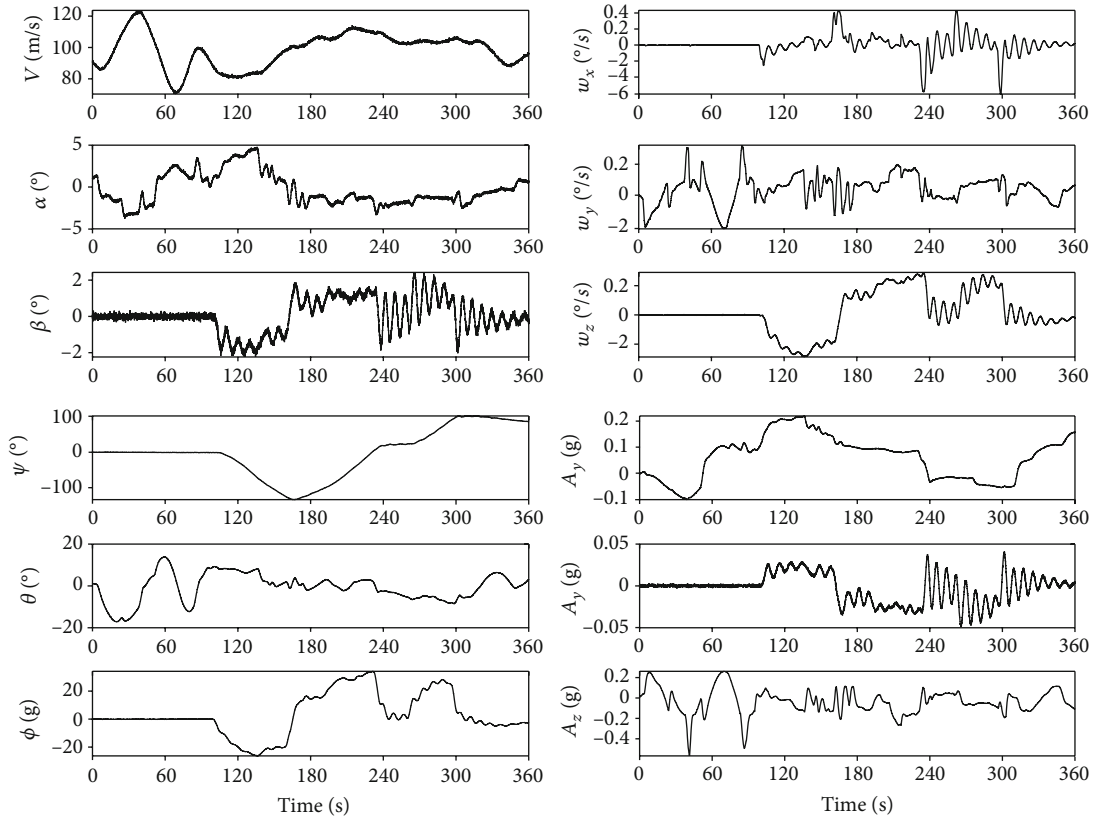

Figure 2: Plots of a sampled flight record allocated in our database.

and summarize the rules we adopted in optimizing several large structure parameters—this enhances the explainability on why such DNN specifics are adopted

(iii) Interpretable DNN small structure: we borrow feature visualization from computer vision in analyzing the DNN small structure, of which the results correspond to what human understand—this elevates the interpretability on how the DNN operations work

The remainder of this paper is organized as follows. Section 2 defines the problem. Section 3 illustrates the database. Section 4 prepares the data and experimental setup. Studies on both large and small structures are detailed in Section 5, wherein lessons and experiences we learned are also summarized. Finally, conclusions and future works are discussed in Section 6.

2. Problem Definition

To define the FD problem of aircraft air data sensors, we start with the air data evolution equations:

$$\begin{cases} \dot{V} = (A_x - gS_\theta)C_\alpha C_\beta + (A_y + gS_\phi C_\theta)S_\beta + (A_z + gC_\phi C_\theta)S_\alpha C_\beta, \\ \dot{\alpha} = (-A_x S_\alpha + A_z C_\alpha + gC_\phi C_\theta C_\alpha + gS_\theta S_\alpha)/VC_\beta + w_y - (w_x C_\alpha + w_z S_\alpha)S_\beta/C_\beta, \\ \dot{\beta} = \dfrac{\left[-(A_x - gS_\theta)C_\alpha S_\beta + (A_y + gS_\phi C_\theta)C_\beta - (A_z + gC_\phi C_\theta)S_\alpha S_\beta\right]}{V} + w_x S_\alpha - w_z C_\alpha, \end{cases} \tag{1}$$

Table 2: Measurement noise standard deviations in the simulation for aircraft D.

Sensor	Standard deviation	Unit
V_m	0.1	[m/s]
$\{\alpha, \beta\}_m$	$0.1\pi/180$	[rad]
$\{A_x, A_y, A_z\}_m$	0.01	$[\mathrm{m/s^2}]$
$\{p, q, r\}_m$	$0.01\pi/180$	[rad/s]
$\{\psi, \theta, \phi\}_m$	$0.01\pi/180$	[rad]

wherein S_* and C_* represent sin and cos operations, $\{V, \alpha, \beta\}$ are airspeed, AOA, and sideslip angle, g is the gravitational acceleration, and $\{w_x, w_y, w_z\}$ and $\{\psi, \theta, \phi\}$ denote rotational speeds and angles, respectively. In Eq. (1), $\{A_x, A_y, A_z\}$ indicates the accelerations along different axes of the aircraft body, which are defined as $\{A_i = F_i/m\}_{i=x,y,z}$, wherein m is the mass of the aircraft, and $\{F_x, F_y, F_z\}$ are the external forces generated by the control actions:

$$\begin{cases} F_x = F_x(\delta_{\mathrm{th}}, V, \alpha, S, \cdots), \\ F_y = F_y(\delta_r, \delta_a, V, \alpha, S, b, c, \cdots), \\ F_z = F_z(\delta_e, V, \alpha, S, c, \cdots). \end{cases} \tag{2}$$

In Eq. (2), δ_* indicates individual control input (e.g.,

Table 3: Training and testing data specifics in this paper.

	Aircraft and condition	Duration (min)	Altitude (km)	Airspeed (m/s)	AOA (°)	Sideslip angle (°)	Case distribution in {0 ~ 5} (%)
Training	B_1 AP cruise	327	[9.63, 10.7, 0.35]	[227, 252, 9]	[-1.3, 0.6, 0.4]	[-1.9, 0.4, 0.6]	{27, 13, 16, 16, 14, 14}
	Y manual LTO	295	[0.03, 0.66, 0.11]	[93, 167, 14]	[-2.7, 7.7, 1.2]	[-2.2, 1.6, 0.3]	{28, 14, 13, 13, 18, 14}
	B_2 manual LTO	67	–	[75, 151, 12]	[0.8, 6.7, 0.8]	[-1.5, 0.5, 0.3]	{28, 20, 14, 11, 16, 11
Testing	F manual flight	30	–	[80, 141, 21]	[10, 11, 45]	[-3.8, 9.6, 2.5]	{10, 10, 17, 19, 15, 29}
	D AP cruise	162	[3.50, 4.00, 0.19]	[68, 71, 0.38]	[0.5, 3.2, 0.37]	[-7.3, 2.7, 2.2]	{23, 16, 17, 14, 17, 13}
	B_2 manual flight	151	[0.01, 1.61, 0.26]	[47, 276, 31]	[-14, 19, 4]	[-8, 5, 1]	{29, 16, 11, 14, 14, 16}

Note: [minimum, maximum, stand deviation] of clean altitude and ADS states are characterized; altitude was not recorded in B_2 and F real flight.

throttle δ_{th}, elevator δ_e, aileron δ_a, and rudder δ_r), and S, c, and b are the aircraft wing area, mean chord length, and span, respectively.

The kinematics of aircraft rotational speeds and angles is written as follows:

$$\begin{cases} \dot{\psi} = \dfrac{w_y \dot{S_\phi}}{C_\theta} + \dfrac{w_z C_\phi}{C_\theta}, \\ \dot{\theta} = w_y \dot{C_\phi} - w_z S_\phi, \\ \dot{\phi} = w_x + \dfrac{w_y S_\phi S_\theta}{C_\theta} + \dfrac{w_z C_\phi S_\theta}{C_\theta}, \end{cases} \tag{3}$$

and dynamics of the aircraft rotation yields

$$\begin{cases} I_x \dot{w}_x - I_{xy} \dot{w}_y = M_x + \left(I_y - I_z\right) w_y w_z - I_{xy} w_x w_z, \\ I_y \dot{w}_y - I_{xy} \dot{w}_x = M_y + \left(I_z - I_x\right) w_x w_z + I_{xy} w_y w_z, \\ I_z \dot{w}_z = M_z + \left(I_x - I_y\right) w_x w_y + I_{xy}\left(w_z^2 - w_y^2\right), \end{cases} \tag{4}$$

wherein $\{M_x, M_y, M_z\}$ are the external control moments, which are defined as

$$\begin{cases} M_x = M_x(\delta_a, \delta_r, V, \alpha, S, b, \cdots), \\ M_y = M_y(\delta_a, \delta_r, V, \alpha, S, b, \cdots), \\ M_z = M_z(\delta_e, V, \alpha, S, c, \cdots). \end{cases} \tag{5}$$

Figure 1 depicts an overall flow for above equations. Traditional works hinge on model-based approaches to monitor the control inputs and sensors outputs. Implicitly in such model, the external control forces/moments must be considered, which are generated using associated control actions, and directly related to the aircraft specifics (e.g., wing area and mass) and flight conditions (e.g., airspeed and AOA). Parameters within such model-based FD scheme typically require ad hoc tuning per aircraft/flight condition. Its scalability is therefore doubtful.

Despite the high dependency of control forces/moments upon aircraft specifics/flight conditions, their outcome (i.e., $\{A_i\}_{i=x,y,z}$ and $\{w_i\}_{i=x,y,z}$) can be directly measured using inertial reference unit (IRU). Via Eq. (3), rotational angles of the aircraft can also be calculated using $\{w_i\}_{i=x,y,z}$

Table 4: ADS fault cases adopted in this paper.

Case	Sensor	Fault type	Magnitude*
5	Sideslip angle	Extra noise	5°~10°
4	Sideslip angle	Drift	±5°~10°
3	AOA	Extra noise	5°~10°
2	AOA	Drift	±5°~10°
1	Airspeed	Drift	-50%~100%
0	Clean measurement with noises and disturbances, no fault		

*Noise standard deviation and drift values defined in this column.

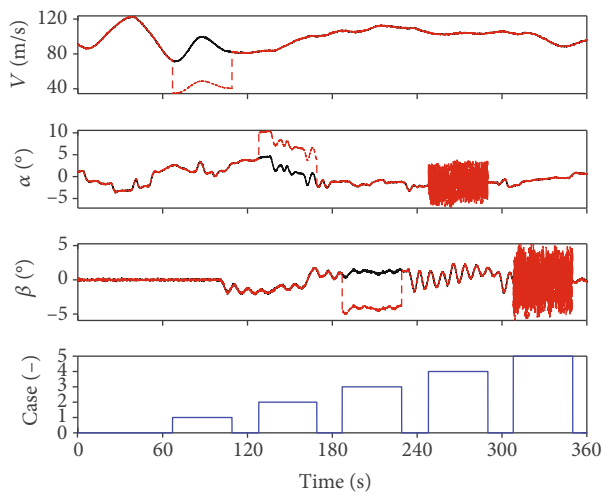

Figure 3: Illustrative plot for the fault injection. Black lines denote clean states from real flight/simulation, and red lines denote the fault-injected data. The FD scheme is expected to detect the different fault cases (blue thick line) based on available data measurements and proper DNN operations.

(although dedicated sensors do exist to directly measure them). We thus adopt IRU measurements as a probe into the overall system, model them as equivalent inputs to the air data evolution, and perform the air data sensors FD task directly.

To be specific, the problem in this paper is defined as to detect (classify) different faults that occur in the air data sensors, given the air data measurements $\{V, \alpha, \beta\}$, and other data resources which may include $\{A_x, A_y, A_z\}$, $\{w_x, w_y, w_z\}$, and $\{\psi, \theta, \phi\}$. The FD scheme is modeled as a mapping process

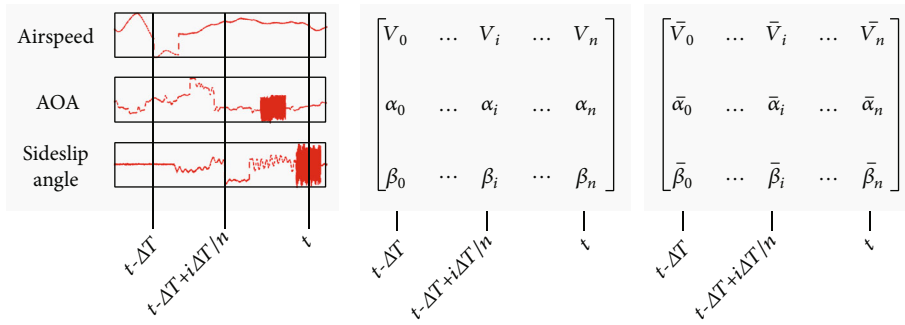

Figure 4: Data preprocessing for ADS. (a) ADS records are retrieved from real flight/simulation, faults are injected into the records; a time window (ΔT) is implemented to clip the data, which is further down-sampled to 1 Hz. (b) the clipped and downsampled data are stacked as the matrix. (c) along each row, the states are normalized linearly within the range of [0, 1], which eventually yields a 3×31 state image for the ADS data. Same approaches are also used for accelerations, rotational speeds, etc.

(input: available data resources, output: fault case), which we aim to regress via deep neural networks.

3. Fault Datasets

3.1. Diverse Aircraft and Flight Conditions. Data is crucial for DNN training and validation. Most previous works discussed 1 aircraft only. In this paper, we allocate both simulation and real flight data from 5 different aircrafts which include 1 large cargo airplane (Y [62]), 2 passenger aircrafts (B_1 and B_2, [63, 64]), 1 general aviation (D [50, 51]), and 1 fighter aircraft (F [65]). We also involve 6 different flight conditions to cover the aircraft's entire envelope, i.e., high, medium, and low altitudes for both cruise, manual free flight, and low-altitude landing/take-off cycle (LTO). Different control forms from both human pilot (manual) and automated control laws (autopilot, AP) are also considered. See Table 1 for more details. In Figure 2, we also plot a sampled data we allocated in our database.

3.2. Measurement Noises and Disturbances. Both simulation and real flight data are considered in the paper. While measurement noises and disturbances exist naturally in the real flight, we adopt the model following [66] in simulation. Dryden atmospheric disturbances are injected to perturb the flight states, on which the measurement noises are added to generate the noise-corrupted data. Measurement noises are assumed to follow Gaussian process distribution. Standard deviations for the noise of each sensor are characterized in Table 2 [17].

3.3. Designated Training and Testing. To avoid the overfitting problem, training and testing data are strictly separated. We put all the real flight data in testing to fully evaluate the DNN performance. Diversity is crucial in specifying the training data, as the training algorithm is expected to extract from this data for an efficient FD mapping. We therefore adopt the real data from B_2, F, and simulated data from D and B_1 manual flight for testing. As for training, we use the data from Y manual LTO and B_1 AP cruise, see Table 3. In the table, an overview of the data for each aircraft/flight condition is also characterized using the minimum, maximum, and stand deviation of key (clean) flight states (i.e., altitude, airspeed, AOA, and sideslip angle).

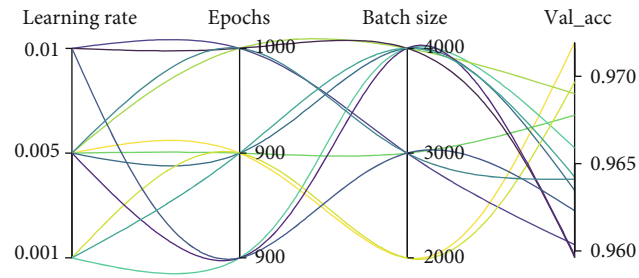

Figure 5: Illustration of NNI: different parameters set of learning rate, epochs, and batch size are examined; the combination of learning rate 0.005, iterative epochs 900, and batch size 2000 yields the optimal DNN training outcome (highest validation accuracy at 0.972).

3.4. ADS Fault Modeling and Injection. Different sensor fault types have been discussed in previous works, which include ramp bias, oscillations, and drift. For airspeed, most flight accidents happened due to the pitot tube being clogged by ice/rain. We thus consider drift fault for airspeed (measurement loss). For AOA and sideslip angles, the deflection vanes may be stuck or perturbed by external atmosphere, which causes drift (constant bias) and extra noises. As in Table 4, a total of 5 ADS fault cases are discussed in this paper, wherein the magnitude for each case is specified following [51].

We implement the ADS faults in an additive form, i.e., the "clean" data (Case 0 in Table 4) are retrieved from real flight/simulation. Sensor faults are then injected into the ADS data. Following [51], this injection is performed in a randomized manner, i.e., for every 60 seconds in the data, the fault cases occur randomly at randomized moments, with its duration (also randomized) not exceeding the 60 seconds. In Figure 3, different fault cases are injected to both airspeed, AOA, and sideslip angle for illustrative purposes. Table 3 also presents the distribution of different cases in the final data we adopt for the DNN training/testing.

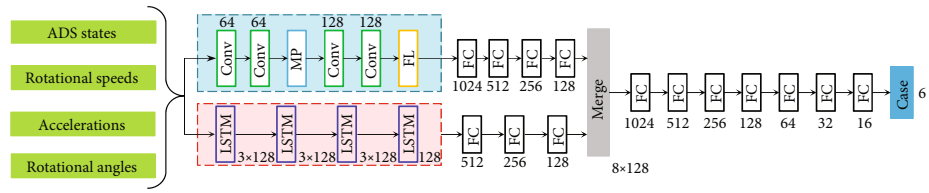

Figure 6: DNN-full; this "full" architecture adopts both CNN and LSTM operations for all variables that relate to the ADS evolution. CNN kernel size, filter numbers, LSTM nodes, and fully connection (FC) layers are specified directly following [50, 51]. Conv: convolutional; MP: max-pooling; FL: flatten.

Table 5: To simplify from a "full" DNN architecture to DNN-final.

Architecture	Operations	Air data	Rotational speeds	Accelerations	Rotational angles	Validation accuracy
DNN-full	CNN	✓	✓	✓	✓	0.925
	LSTM	✓	✓	✓	✓	
DNN-angles	CNN	✓	✓	✓		0.932
	LSTM	✓	✓	✓		
DNN-final	CNN	✓				0.946
	LSTM	✓	✓	✓		

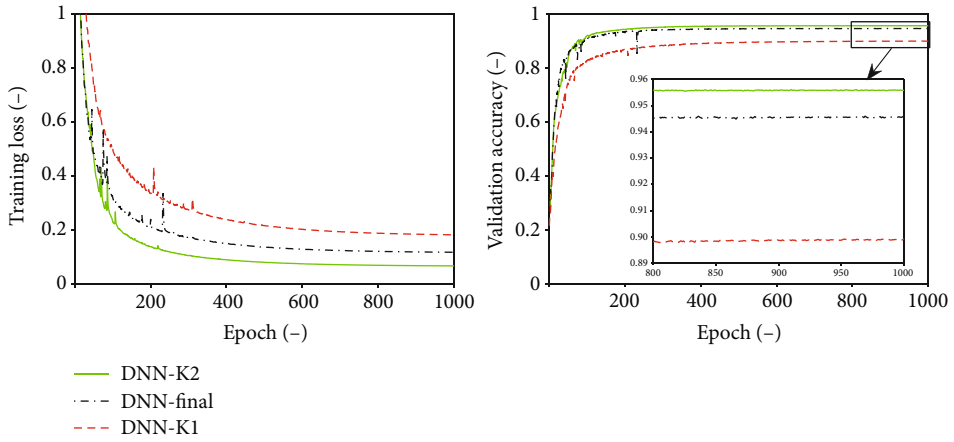

Figure 7: Studies on different CNN kernel sizes. The results are summarized from the training of DNN-final (kernel size 3×3), and 2 descendent architectures of DNN-final, i.e. DNN-K2 (kernel size tweaked as 2×2) and DNN-K1 (kernel size 1×1).

4. Premise for the DNN-Based ADS Fault Detection

4.1. A Brief Introduction of CNN and LSTM. We use both 2-dimensional CNN and LSTM in this paper. In CNN [67], convolutional filters scan the input (e.g., an image), of which the results are concatenated as feature maps. Multiple filters yield various feature maps, which being stacked to construct the designated mapping. Activation functions and pooling operations may also be used, with the former adding to nonlinearity in the mapping and the latter the noise tolerance [50, 51, 68].

LSTM is typically used for sequential data. Different from the spatial local-connectivity features extraction in CNN, it aims to abstract the temporal knowledge. Previous DNN including RNN [67] also targets the temporal features. RNN, however, may suffer error explosion/vanishing problems in the training. LSTM adopts gate operations to automatically select the historic input that may be useful in the mapping. As proved in many works, training efficiency, mapping accuracy, and deployment cost of LSTM can all be improved [50, 51].

In coping with the dynamics problem, both CNN and LSTM may be useful. Via CNN, spatial correlations of different states are abstracted. In LSTM, the temporal features of each state are modeled. Previous works advocated a fusion of CNN and LSTM in designing the DNN architecture [50, 51] for parameter identification, icing detection, etc. As there is not an input explicitly defined as "image," the key in implementing CNN and LSTM in these problems is to reshape the dynamic data (e.g., flight states and control commands) into an image-like format. We detail this in Section 4.2.

4.2. Data Preprocessing. We perform data preprocessing to generate the image-formatted input to CNN and LSTM, see Figure 4. Via real flight or simulation, we have the

Table 6: Different CNN architectures adopted in studying the CNN filter numbers; LSTM branches of all listed architectures remain the same as in Figure 6.

Architecture	Conv (filter no.)	Conv (filter no.)	MP layer	Conv (filter no.)	Conv (filter no.)	FC layers (FC node no.)				
1	8	8	MP	16	16	256	128			
2	16	16	MP	32	32	256	128			
3	24	24	MP	48	48	512	256	128		
4	32	32	MP	64	64	512	256	128		
5	40	40	MP	80	80	768	384	192	128	
6	48	48	MP	96	96	768	384	192	128	
7	56	56	MP	112	112	1024	512	256	128	
DNN-K2	64	64	MP	128	128	1024	512	256	128	
8	72	72	MP	144	144	1280	640	320	160	128
9	80	80	MP	160	160	1280	640	320	160	128

records of different states. We inject faults into the ADS states, allocate all other flight data, and stack them into a 2D matrix (middle plot). In this matrix, each row stands for the historic measurements of a certain state and column the value of that state at a certain moment. For each group of the aircraft flight state (e.g., air data, accelerations, rotational speed, and rotational angles), we stack this matrix separately.

Time window is also used. At each moment t, we consider the flight records in a range from previous $t - \Delta T$ to t (both included), wherein ΔT is 30 s (following [50, 51], this window may be understood as a compromise between the aircraft "fast" motion modes of which the periods are in seconds (e.g., longitudinal short period and lateral roll) and "slow" modes which typically last for tens/hundreds of seconds (e.g., longitudinal phugoid and Dutch roll)). For different aircrafts, the data is recorded in various sampling rates (e.g., 20 Hz for B_1 and 30 HZ for F). We downsample them to a unified frequency at $f_s = 1/\Delta t$, wherein $\Delta t = \Delta T/n = \Delta T/30 = 1s$. We then stack the state matrix using the resampled data.

In the downsampled flight data, the range of each state varies significantly (see Table 3). In practice, this may create numerical difficulties in the DNN training (singularities, error vanishing/explosion). Normalization is adopted to process the sampled data. Following [50, 51], this normalization is performed linearly along each row of the stacked matrix. After normalization, the "image" we obtain in the right plot of Figure 4 is adopted as input to the DNN.

4.3. Experimental Setup. In training/testing the DNN, we record both training loss and validation accuracy (with all testing data designated as the validation dataset). Following [51], we repeatably perform 30 training runs for all DNN architectures, excluding the best/worst 5 runs, and summarize the outcome via the remained 20 records. We also adopt Keras (version 2.0.8) API with Tensorflow (version 1.3.0) as backend in the programming. Our computational platform is configured with CUDA 8.0 and cuDNN 6.0 with Nvidia driver version 384.69 (GPU: Nvidia GTX2080Ti) in Windows 10 system (Python 3.7). The platform has one i9-9900K CPU and 32 GB RAM.

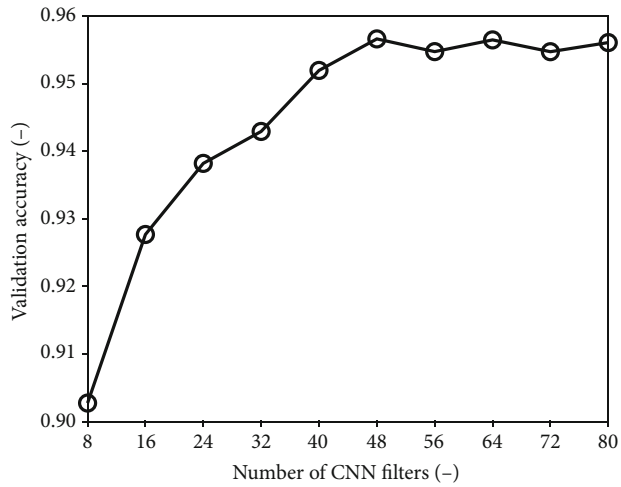

Figure 8: Training outcomes of DNN-K2 (with 64 CNN filters) and other 9 descendant architectures (Table 6); horizontal axis indicates the number of CNN filters enclosed in the first convolutional layer.

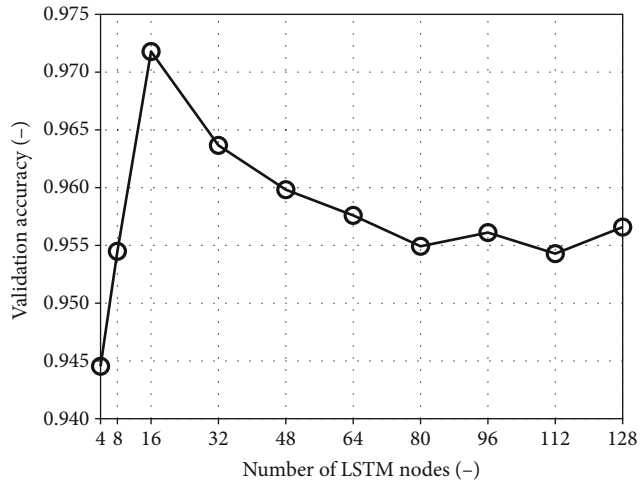

Figure 9: Training outcomes of DNN-C48 and other descendant architectures; horizontal axis indicates the number of nodes enclosed in the LSTM branches.

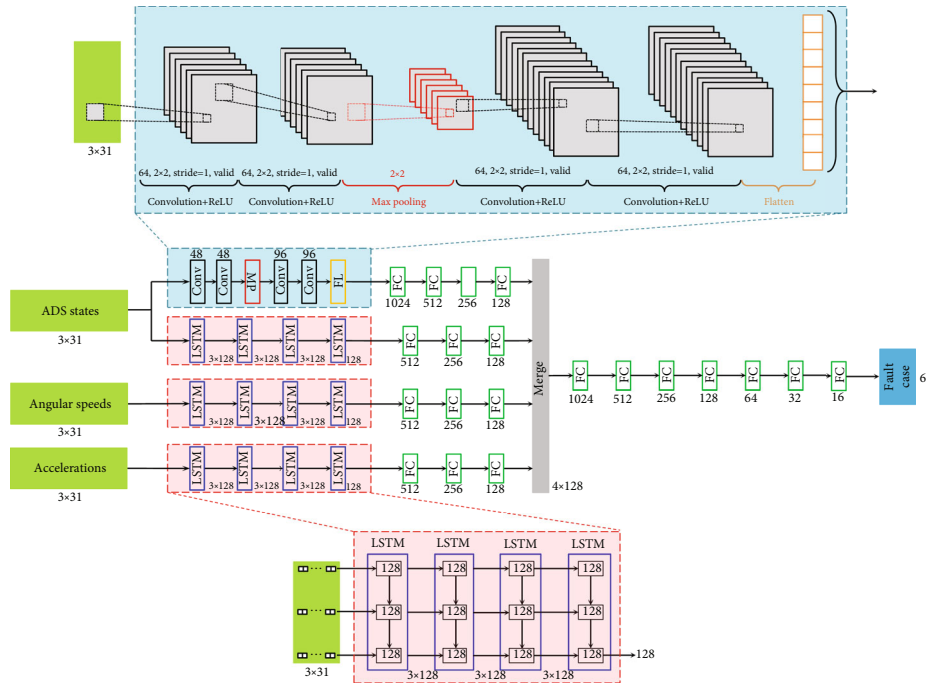

FIGURE 10: The DNN-opt architecture.

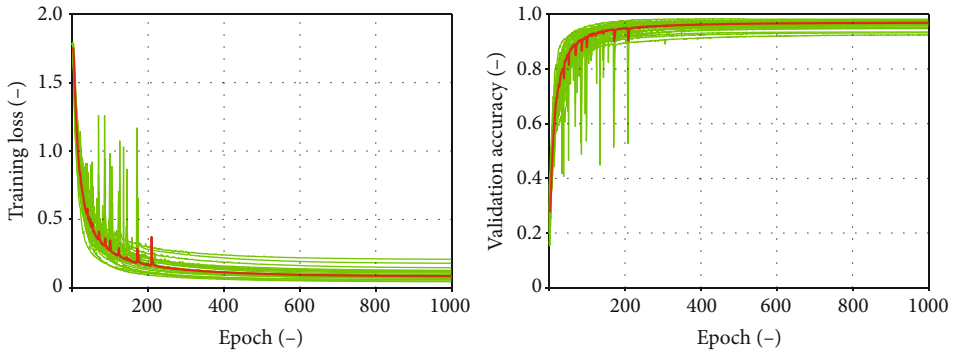

FIGURE 11: Training history of DNN-opt; green indicates the 30 independent runs, and red the summarized outcome (averaged value excluding the best/worst 5 runs).

4.4. Optimal DNN Training. As illustrated in following contents, comparative studies are adopted in evaluating the testing performances of different DNN architectures. Different training coefficients (e.g. learning rate and batch size in the training algorithm) may affect the training outcome, hence the testing results. We thus adopt NNI developed by Microsoft [54]. Given different DNN architectures, NNI probes into the architecture, analyzes the training data, and decides an optimal combination of the training parameters, see Figure 5. Via NNI, different DNN architectures are trained separately in associated "optimal" manners, which we believe provides more solid ground for the comparative studies.

5. Development and Study on the DNN-Based Fault Detection Scheme

Development of the DNN-based FD scheme involves two-folds: to devise the DNN large architecture and to optimize the parameters enclosed within, and both are detailed in Sections 5.1 and 5.2 and verified via ablation studies in 5.3. Interpretability analysis on the DNN small structure is given in 5.4. Experiences and lessons we have learned are discussed in Section 5.5.

5.1. Devising the Large Architecture. In devising the large architecture, we need to (1) select the DNN input (output being the fault cases directly) and (2) decide the CNN/LSTM branches. To the authors' best knowledge, there does not exist a universal rule for such issue. We refer to [50, 51] and start with a "full" architecture in Figure 6 (DNN-full). In Eq. (1), both accelerations, rotational angles, and rotational speeds relate to the ADS states. DNN-full thus absorbs all these variables in the input layer and uses both CNN and LSTM branches to fully extract the features. We follow [50, 51] in specifying the CNN filters/LSTM nodes. NNI is adopted to decide the optimal training coefficients, and

Table 7: Confusion matrix in testing the DNN-opt.

Aircraft and condition	FD accuracy in cases {0-5} (%)						Aircraft and condition	FD accuracy in cases {0-5} (%)					
B_2 manual LTO	92.08	0.40	0.24	1.28	3.16	2.84	D AP cruise	93.66	0	0	2.46	2.16	1.72
	7.24	92.43	0	0	0.33	0		0	100	0	0	0	0
	3.54	0	90.16	2.97	1.63	1.70		0.06	0	96.64	0.47	1.53	1.30
	2.18	0	0.07	96.70	0	1.05		2.28	0	0.78	95.52	0	1.42
	3.27	0	0	0.08	95.83	0.82		3.32	0	0.14	0.34	94.98	1.22
	1.71	0	0.47	0.54	0.31	96.97		3.95	0	0.25	0.67	0.34	94.79
Aircraft and condition	FD accuracy in cases {0-5} (%)						Aircraft and Condition	FD accuracy in cases {0-5} (%)					
F manual flight	90.00	0	0	0	2.00	8.00	B_1 manual flight	98.44	0	0	0	0.39	1.17
	0	100	0	0	0	0		0	100	0	0	0	0
	0	0	100	0	0	0		0	0	96.65	0	1.12	2.23
	0	0	0	100	0	0		1.27	0	0	95.55	0	3.18
	1.67	0	0	0	98.33	0		0	0	0	0	100	0
	0.56	0	2.23	0	0	97.21		4.46	0	1.27	3.82	0	90.45

Table 8: Ablation studies on DNN-opt.

Architecture	Operations	Air data	Rotational speeds	Accelerations	Validation accuracy
DNN-opt	CNN	✓			0.972
	LSTM	✓	✓	✓	
opt-accLSTM	CNN	✓			0.945
	LSTM	✓	✓		
opt-spdLSTM	CNN	✓			0.893
	LSTM	✓		✓	
opt-adsLSTM	CNN	✓			0.875
	LSTM		✓	✓	
opt-allLSTM	CNN	✓			0.773
	LSTM				
opt-adsCNN	CNN				0.668
	LSTM	✓	✓	✓	

training outcome in Table 5 yields promising results (validation accuracy 0.925).

Starting with DNN-full, we proceed to simplify the DNN architecture (whilst the performance still needs to be guaranteed). Referring to Eq. (3), the input of rotational angles may be redundant, as they can be completely calculated using rotational speeds. Input branches of the rotational angles are then cropped from DNN-full, and the rest remained unchanged as DNN-angles (Table 5). Again via NNI, DNN-angles are trained in an optimal manner, which yields an even better outcome when compared with DNN-full (validation accuracy 0.932).

The better training outcome of DNN-angles provides an import cue in devising the DNN architecture: whilst input (e.g., data resources) and operations (e.g., CNN and LSTM) enclosed within the DNN must be complete, the architecture should be as simple as possible—to render the training load lighter. In DNN-angles, referring to Eq. (1), we consider the that input air data, accelerations, and rotational speeds are complete (all necessary), as they are all independently related with the ADS evolution in (1). However, CNN is typically used to abstract the coupling effects. While the coupling does exist in air data, the 3 states of rotational speeds and accelerations are considered uncoupled as each state is generated via independent control actions. We thus exclude CNN in the rotational speeds and acceleration branches and propose the DNN-final architecture (Table 5). Again via NNI, an optimal training outcome of DNN-final yields performances even better than DNN-angles (0.946). DNN-final also yields the architecture we adopt in developing the DNN-based FD scheme.

5.2. Optimizing the Large Structure Parameters. In the previous chapter, via gradually cropping the DNN architecture and investigating the training outcome, we adopt DNN-final. Whilst this architecture is further verified in the next chapter via ablation studies, in this chapter, we focus on DNN-final and seek to find the optimal large structure parameters. In particular, we aim to find the best CNN

kernel size and decide the optimal CNN filters/LSTM node number in DNN-final.

5.2.1. CNN Kernel Size. In DNN-final, we follow [50, 51] and adopt a 3×3 CNN kernel size. The input size in this paper, however, is smaller (3×31) than in [50, 51] (11×31). We thus study 3 different CNN kernel sizes, i.e., 3×3, 2×2, and 1×1, see Figure 7. Based on the comparative results in the figure, kernel size 2×2 yields the best training outcome (i.e., DNN-K2).

DNN-K2 corresponds to the physical understanding of convolutional operations. On the one hand, although 1×1 kernels have appeared in literature to provide a delicate scanning on the input [69], the essence of CNN filters is to extract local-connectivity features. A kernel is imminent for "patching" the features. DNN-K1 in Figure 7 thus yields the worst performance. On the other hand, a full kernel which operates on the whole input dimension (e.g., 3×3 in our case for the 3×31 image) has appeared [70], which was claimed to provide a panoramic view of the entire image, but a larger CNN kernel may render the training more difficult, hence jeopardizing the training outcome, as yielded by DNN-final in Figure 7. In the following contents, we perform the iterative studies all starting with DNN-K2.

5.2.2. CNN Filter Numbers. Based on DNN-K2, we proceed to decide how many filters should be used in each layer of the CNN. In DNN-K2, we use 64 filters for 2 consecutive CNN layers, followed by another 2 layers with 128 filters, with a max-pooling layer laid in between. In our studies, we retain this structure and inflate/deflate CNN filter numbers in the 4 layers synchronously, thus creating a total of 9 descendant architectures (Table 6). Note that in the table, subsequent fully connection layers are also tuned to match the CNN output size. Training outcomes of the 10 architectures in Table 6 are plotted in Figure 8, wherein 48 CNN filters yields the best performance.

The plots in Figure 8 correspond to the understanding of CNN filters. To start with, multiple CNN filters must be used to stack various features. With more filters being adopted in the CNN, the performance may elevate. However, there also exists a certain level of CNN filters that yields "saturated" feature extraction which is illustrated by the plateau in Figure 8. We mark the DNN-K2 with 48 CNN filters as DNN-C48, starting from which we perform the iterative studies on LSTM node number in following contents.

5.2.3. LSTM Node Numbers. In DNN-C48, all LSTM operations remain unchanged as in [50, 51], and 128 LSTM nodes are used ubiquitously in all LSTM layers. We retain the CNN branches in DNN-C48, tweak the node number in all LSTM layers (synchronously), and create multiple descendant architectures. Again, via NNI, we perform optimal training for all these architectures. The associated training results are shown in Figure 9. Clearly, the training outcome reaches a peak at 16 nodes.

In Figure 9, starting from 4 nodes to 16, we observe an almost linear elevation in the training outcome—this probably is due to the relative simple activation functions being used in the LSTM, and more LSTM nodes provides better performance as more temporal features are extracted. After the 16 nodes, however, there exists a slow drop (also almost linear) towards 80 nodes, mainly due to the heavier LSTM training load. The plateau is again found after 80 nodes, and the reason is similar: LSTM node number is saturated already; purely increasing the LSTM nodes will not elevate the performance.

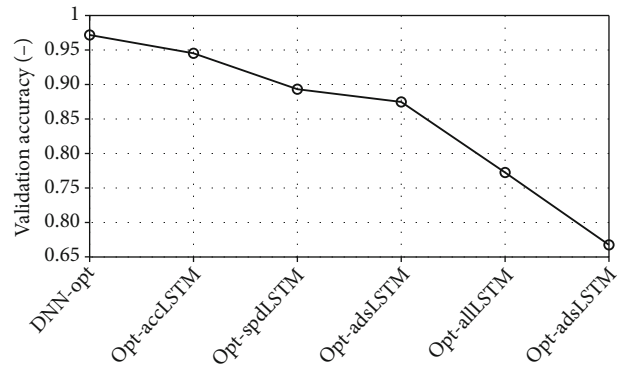

Figure 12: Ablation studies on DNN-opt. Specifics of each ablated architecture are presented in Table 6. DNN-opt yields the best validation accuracy at 0.972.

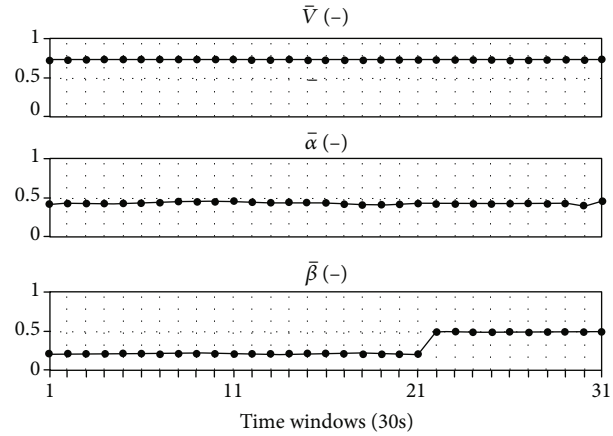

Figure 13: Normalized flight records for features visualization; as in Figure 4, the data has been downsampled to 1 Hz; the sideslip drift fault occurs at last 1/3 of the window, i.e., 21 ~ 31 s.

Figure 14: State image visualization for the records in Figure 13; width of the image corresponds to the time window (30s), and the 3 rows (from top to bottom) indicate normalized airspeed, AOA, and sideslip angle, respectively; the sideslip drift fault is marked with a red box.

5.2.4. DNN-Opt. In Sections 5.2.1-5.2.3, we optimize the large structure parameters enclosed within DNN-final. We mark the best DNN-final descendant as DNN-opt (CNN

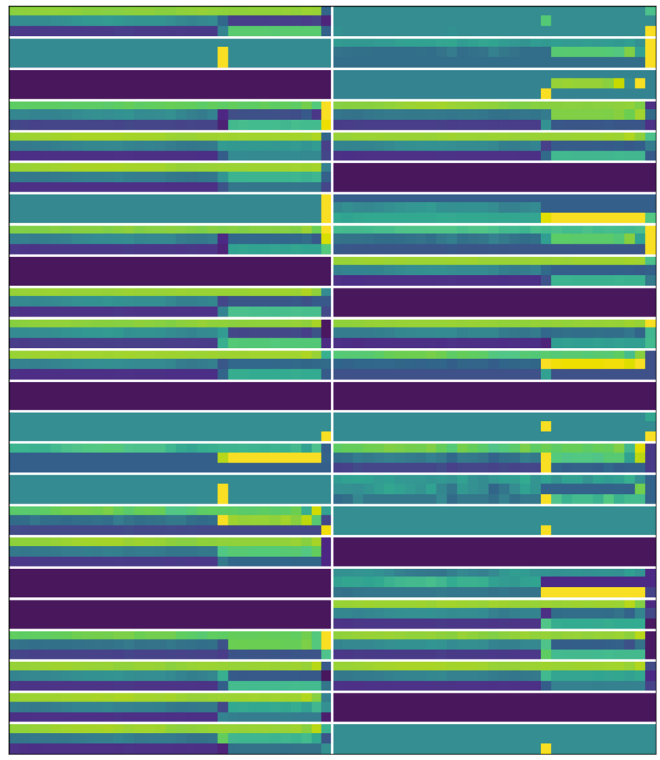

Figure 15: Feature visualization of the first convolution layer (48 in total).

kernel size 2×2, filter number 48, and LSTM node number 16) and detail its architecture in Figure 10. Via NNI, training coefficients of DNN-opt are optimized in Figure 5. Training histories of DNN-opt are illustrated in Figure 11.

We also implement DNN-opt for the testing data in Section 3 and characterize the testing confusion matrix in Table 7. In the table, the FD accuracies in 6 different cases (see Table 4) for 4 different aircraft and conditions (see Table 3) were investigated. In the confusion matrix for each aircraft and condition, the horizontal direction indicates the real cases and vertical the detected cases from DNN-opt. Shadowed diagonal elements in the matrix represents the detection accuracy for each case, and the offdiagonal values indicate wrong detection rates (e.g., 2.46% in the D AP cruise matrix indicates that 2.46% of the real case 3 data were detected as case 0 via DNN-opt). As shown in the table, DNN-opt yields promising results (90% accuracy) in the ADS fault detection problem for all 4 aircraft at diverse flight conditions.

5.3. Ablation Studies on Large Structure. The DNN-opt architecture originates from DNN-final, which was obtained via cropping redundant operations from DNN-full architecture. In this part, we verify the DNN-opt architecture via ablation studies, see Table 8 and Figure 12. We crop the LSTM and CNN branches for designated inputs from DNN-opt, optimize the training of the remained part via NNI, and investigate the training outcomes. As found in Table 8 and Figure 12, convolution of the air data branch in DNN-opt relates to the DNN-opt performance most significantly (accuracy of opt-adsCNN drops most drastically). This corresponds to the previous analysis in cropping from DNN-full to DNN-final: convolution strives to abstract the coupling effects of the input; in our case, we rely primarily on the correlations in air data to assert the fault. Although other structures (e.g., LSTM for accelerations and opt-accLSTM) does not relate as significantly as the air data convolution, the associated performances still deteriorate from the best DNN-opt. The results in Table 8 and Figure 12 prove that DNN-opt claims the best performance as compared with other ablated structures.

5.4. Interpretable Analysis on Small Structure. We explore to interpret the air data convolutional operations in DNN-opt, as it affects the DNN-opt performance most significantly. In Figure 13, flight records corresponding to fault case 4 (Table 4, sideslip angle drift) are plotted. The associated state image is stacked in Figure 14. Features abstracted in the 4 convolution layers in DNN-opt are shown in Figures 15–18. Note that we adopt "same" padding in DNN-opt, and feature dimension exported from the first 2 convolutional layers thus remains the same as input state image. The highlighted features on Figures 15 and 16 corresponds to the fault occurrence marked with red box in Figure 14 (last 1/3 of the time window, i.e., 21 ~ 31 s). In the second 2 convolutional layers after the max-pooling, size of the features is reduced to half; the highlighted features also reflect the fault marked on Figure 14 (last 1/3 along the horizontal axis).

To better illustrate the feature extraction in DNN-opt, we adopt the class activation mapping (CAM) technique proposed in [61] and visualize the CAM plots for the 6 cases separately in Figure 19. For illustrative purposes, the state image imported to DNN-opt is also shown on the left of

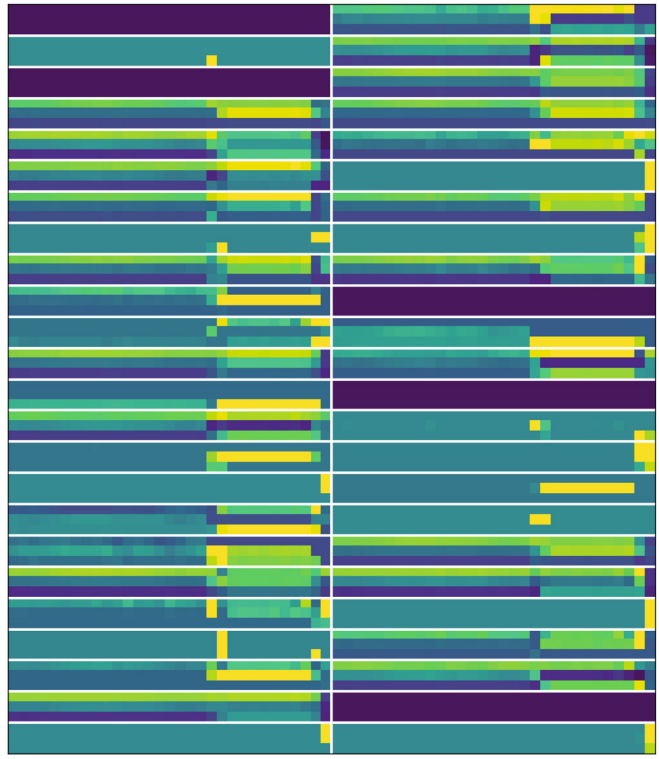

Figure 16: Feature visualization of the second convolution layer (48 in total).

Figure 17: Feature visualization of the third convolution layer (96 in total).

the figure. Note that a "hotter" mapping on CAM indicates the highlighted regions that convolution hinges on to assert the FD output. In Figure 19, CAM corresponds to the general understanding of the FD problem, as the highlighted hotter (red) regions overlap the areas that the fault occurs (marked with red).

5.5. Experiences and Lessons. In Sections 5.1-5.4, we have completed the training/testing of DNN-opt; explainability and interpretability analysis is also presented. In this part, we summarize our experiences and lessons in developing the DNN-based FD scheme:

Start with a "full" DNN. Iterative studies are usually adopted in devising the DNN large structure and optimizing the associated parameters. This iteration, however, must start from a certain architecture. Although there still lacks a rule as to how to initialize such a DNN architecture, staring from a "full" one proves to be effective in our work (i.e., DNN-full). The full DNN should involve all available data sources in the input layer and implement all potentially useful DNN

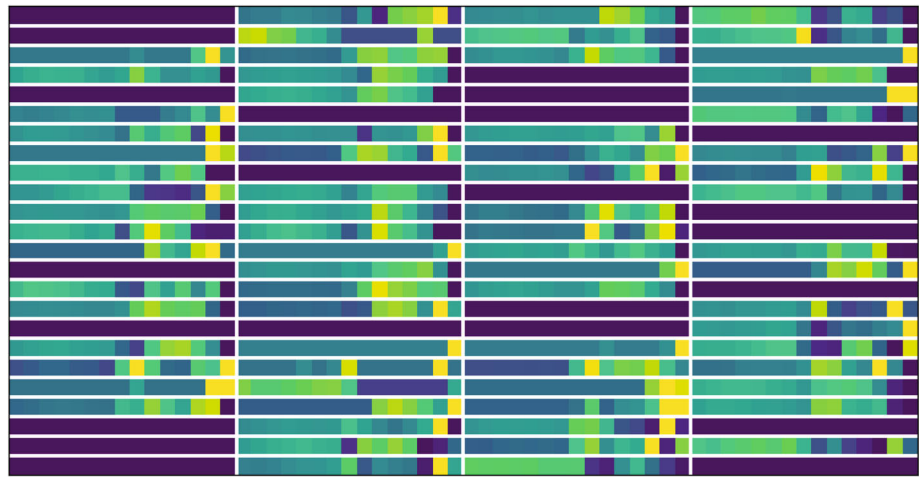

Figure 18: Feature visualization of the fourth convolution layer (96 in total).

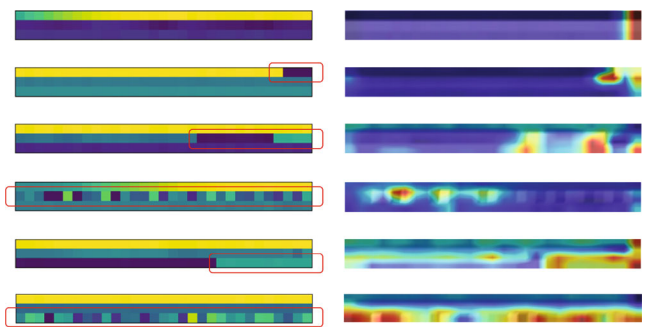

Figure 19: Illustrative CAM plots for the 6 cases (top to bottom, cases $0 \sim 5$ in Table 4); the highlighted (red hot) features correspond to the faults occurred on the state image (marked with red boxes).

operations to fully extract the features (e.g., CNN for spatial and LSTM for temporal).

(i) Simplify the architecture: although redundant inputs/operations in the DNN architecture may provide extra information in extracting the features, the training load is usually high. The DNN architecture should be simplified (cropped) as much as possible, to the point that an accuracy plateau/peak is found (e.g., DNN-C48 in our work)

(ii) The iteration policy: multiple large structure parameters need to be studied in simplifying the DNN architecture (e.g., CNN kernel size and filter number). Whilst we suggest to study one parameter at a time, the iteration policy is eminent: which to iterate first. In our studies, we have tried different combinations in iterating the CNN kernel size, CNN filter number, and LSTM node number and decided the work presented in the paper prove to be most effective. In other related works, more parameters may need to be tuned (e.g., fully connection node number); similar policy still needs to be studied

(iii) Explainability and interpretability analysis: neural networks were long considered as a "black box". Recent developments in computer vision and natural language processing, however, indicate that certain rules do exist in explaining (devising) the DNN large structure, and the enclosed operations correspond to what humans understand. Similar concepts/approaches are advocated in developing/analyzing the DNN-based FD schemes (e.g., ablation studies and CAM which claim promising results in our work)

6. Conclusion and Future Works

Exemplifying the fault detection (FD) problem of aircraft air data sensors, we aim to develop a robust DNN-based FD scheme in this paper. We model the FD problem using aircraft inertial referent unit measurements as equivalent inputs and construct a dedicated database which involves different aircraft/conditions; both provide a solid basis in training/testing the DNN. In devising the DNN architecture, we adopt iterative studies on specifying the large structures and optimizing the parameters enclosed within. Ablation studies are also adopted to explain the constructed DNN architecture. Whilst the developed DNN yields promising training/testing performances, we adopt methods widely adopted in computer vision (i.e., feature visualization and CAM) in interpreting the DNN small structure operations, which correspond to what humans understand in similar contexts. Combining all the above, the developed DNN is considered robust: performance accurate and scalable, large structure explainable, and small structure interpretable.

As a continuation of the work, we plan to implement similar formulations to other FD problems, e.g., aircraft actuators faults and communication datalink failures in commercial airlines. Interpretation of other operations (e.g., LSTM) in the fault detection context will also be studied.

Data Availability

Data is available upon request to the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was sponsored by the Shanghai Sailing Program under Grant No. 20YF1402500 and Natural Science Foundation of Shanghai under Grant No. 22ZR1404500.

References

[1] J. Levine, *X-31's loss*, 2004, Accessed September 2020 URL http://www.nasa.gov/centers/dryden/news/X-Press/stories/2004/013004/new_x31.html.

[2] F. Bureau d'Enquetes, *et d'Analyses pour la securite de l'aviation civile, Paris, Report 3: on the accident on 1 june 2009 To the Airbus a330–203 Registered f-Gzcp Operated by Air France Flight Af 447 Rio de Janeiro–Paris*, 2011, Accessed September 2020 URL https://www.bea.aero/docspa/2009/f-cp090601e3.en/pdf/f-cp090601e3.en.pdf.

[3] FAA, *FAA Updates on Boeing 737 MAX*, 2020, Accessed October, 2020 https://www.faa.gov/news/updates/?newsId=93206.

[4] P. Goupil, J. Boada-Bauxell, A. Marcos, E. Cortet, M. Kerr, and H. Costa, "Airbus efforts towards advanced real-time fault diagnosis and fault tolerant control," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3471–3476, 2014.

[5] E. Dubrova, *Fault-tolerant design*, Springer, 2013.

[6] P. Goupil, "Airbus state of the art and practices on FDI and FTC in flight control system," *Control Engineering Practice*, vol. 19, no. 6, pp. 524–539, 2011.

[7] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, "Model-based fault diagnosis for aerospace systems: a survey," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 226, no. 10, pp. 1329–1360, 2012.

[8] S. Hussain, M. Mokhtar, and J. M. Howe, "Sensor failure detection, identification, and accommodation using fully connected cascade neural network," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1683–1692, 2014.

[9] S. Hussain, M. Mokhtar, and J. M. Howe, "Aircraft sensor estimation for fault tolerant flight control system using fully connected cascade neural network," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2013.

[10] M. L. Fravolini, G. Del Core, U. Papa, P. Valigi, and M. R. Napolitano, "Data548 driven schemes for robust fault detection of air data system sensors," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 1, pp. 234–248, 2017.

[11] M. L. Fravolini, M. Rhudy, S. Gururajan, S. Cascianelli, and M. Napolitano, "Experimental evaluation of two pitot free analytical redundancy techniques for the estimation of the airspeed of an UAV," *SAE International Journal of Aerospace*, vol. 7, no. 1, pp. 109–116, 2014.

[12] S. Gururajan, M. L. Fravolini, H. Chao, M. Rhudy, and M. R. Napolitano, "Performance evaluation of neural network based approaches for airspeed sensor failure accommodation on a small UAV," in *21st Mediterranean Conference on Control and Automation*, pp. 603–608, IEEE, 2013.

[13] M. Kordestani, M. Saif, M. E. Orchard, R. Razavi-Far, and K. Khorasani, "Failure prognosis and applications—a survey of recent literature," *IEEE transactions on reliability*, vol. 70, no. 2, pp. 728–748, 2019.

[14] M. Rezamand, M. Kordestani, R. Carriveau, D. S. Ting, M. E. Orchard, and M. Saif, "Critical wind turbine components prognostics: a comprehensive review," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 12, pp. 9306–9328, 2020.

[15] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, "Ablation studies in artificial neural networks," http://arxiv.org/abs/1901.08644.

[16] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

[17] L. Van Eykeren and Q. Chu, "Sensor fault detection and isolation for aircraft control systems by kinematic relations," *Control Engineering Practice*, vol. 31, pp. 200–210, 2014.

[18] M. Ariola, M. Mattei, I. Notaro, F. Corraro, and A. Sollazzo, "An SFDI observer–based scheme for a general aviation aircraft," *International Journal of Applied Mathematics and Computer Science*, vol. 25, no. 1, pp. 149–158, 2015.

[19] Q. He, W. Zhang, P. Lu, and J. Liu, "Performance comparison of representative model-based fault reconstruction algorithms for aircraft sensor fault detection and diagnosis," *Aerospace Science and Technology*, vol. 98, p. 105649, 2020.

[20] P. Lu, L. Van Eykeren, E.-J. Van Kampen, Q. P. Chu, and B. Yu, "Adaptive hybrid unscented Kalman filter for aircraft sensor fault detection, isolation and reconstruction," in *AIAA Guidance, Navigation, and Control Conference*, 2014.

[21] P. Lu, L. Van Eykeren, E. Van Kampen, C. De Visser, and Q. Chu, "Adaptive three-step Kalman filter for air data sensor fault detection and diagnosis," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 3, pp. 590–604, 2016.

[22] P. Lu, L. Van Eykeren, E.-J. Van Kampen, and Q. P. Chu, "Air data sensor fault detection and diagnosis with application to real flight data," in *AIAA Guidance, Navigation, and Control Conference*, 2015.

[23] D. Berdjag, J. Cieslak, and A. Zolghadri, "Fault detection and isolation of aircraft air data/inertial system," *Progress in Flight Dynamics, Guidance, Navigation, Control, Fault Detection, and Avionics*, vol. 6, pp. 317–332, 2013.

[24] K. Rudin, G. J. Ducard, and R. Y. Siegwart, "A sensor fault detection for aircraft using a single Kalman filter and hidden Markov models," in *2014 IEEE Conference on Control Applications (CCA)*, pp. 991–996, IEEE, 2014.

[25] P. Freeman, P. Seiler, and G. J. Balas, "Air data system fault modeling and detection," *Control Engineering Practice*, vol. 21, no. 10, pp. 1290–1301, 2013.

[26] M. Mattei and G. Paviglianiti, "Managing sensor hardware redundancy on a small commercial aircraft with H∞ FDI observers," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 347–352, 2005.

[27] F. Amato, C. Cosentino, M. Mattei, and G. Paviglianiti, "A direct/functional redundancy scheme for fault detection and isolation on an aircraft," *Aerospace Science and Technology*, vol. 10, no. 4, pp. 338–345, 2006.

[28] Y. Xue, Z. Zhen, L. Yang, and L. Wen, "Adaptive fault-tolerant control for carrier-based uav with actuator failures," *Aerospace Science and Technology*, vol. 107, p. 106227, 2020.

[29] Y. Wan and T. Keviczky, "Real-time fault-tolerant moving horizon air data estimation for the reconfigure benchmark," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 3, pp. 997–1011, 2018.

[30] Y. Wan, T. Keviczky, and M. Verhaegen, "Robust air data sensor fault diagnosis with enhanced fault sensitivity using moving horizon estimation," in *2016 American control conference (ACC)*, pp. 5969–5975, IEEE, 2016.

[31] Y. Wan and T. Keviczky, "Implementation of real-time moving horizon estimation for robust air data sensor fault diagnosis in the reconfigure benchmark," *IFAC-PapersOnLine*, vol. 49, no. 17, pp. 64–69, 2016.

[32] P. Castaldi, N. Mimmo, and S. Simani, "Avionic air data sensors fault detection and isolation by means of singular perturbation and geometric approach," *Sensors*, vol. 17, no. 10, p. 2202, 2017.

[33] P. Castaldi, W. Geri, M. Bonfe, S. Simani, and M. Benini, "Design of residual generators and adaptive filters for the FDI of aircraft model sensors," *Control Engineering Practice*, vol. 18, no. 5, pp. 449–459, 2010.

[34] X. Zhu, J. Chen, and Z. H. Zhu, "Adaptive learning observer for spacecraft attitude control with actuator fault," *Aerospace Science and Technology*, vol. 108, 2021.

[35] P. Rosa and C. Silvestre, "Fault detection and isolation of LPV systems using set-valued observers: an application to a fixed-wing aircraft," *Control Engineering Practice*, vol. 21, no. 3, pp. 242–252, 2013.

[36] R. Toscano and P. Lyonnet, "Diagnosis of the industrial systems by fuzzy classification," *ISA Transactions*, vol. 42, no. 2, pp. 327–335, 2003.

[37] M. Mousavi, M. Moradi, A. Chaibakhsh, M. Kordestani, and M. Saif, "Ensemble based fault detection and isolation of an industrial gas turbine," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2351–2358, IEEE, 2020.

[38] M. Kordestani, M. Rezamand, M. Orchard, R. Carriveau, D. Ting, and M. Saif, "Planetary gear faults detection in wind turbine gearbox based on a ten years historical data from three wind farms," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 10318–10323, 2020.

[39] Y. Tan, J. Zhang, H. Tian et al., "Multi-label classification for simultaneous fault diagnosis of marine machinery: a comparative study," *Ocean Engineering*, vol. 239, p. 109723, 2021.

[40] L. Garbarino, G. Zazzaro, N. Genito, G. Fasano, and D. Accardo, "Neural network based architecture for fault detection and isolation in air data systems," in *2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC)*IEEE.

[41] M. Kordestani, M. F. Samadi, and M. Saif, "A distributed fault detection and isolation method for multifunctional spoiler system," in *2018 IEEE 61st international Midwest symposium on circuits and systems (MWSCAS)*, pp. 380–383, IEEE, 2018.

[42] A. Abbaspour, P. Aboutalebi, K. K. Yen, and A. Sargolzaei, "Neural adaptive observer-based sensor and actuator fault detection in nonlinear systems: application in UAV," *ISA Transactions*, vol. 67, pp. 317–329, 2017.

[43] M. L. Fravolini, M. R. Napolitano, G. Del Core, and U. Papa, "Experimental interval models for the robust fault detection of aircraft air data sensors," *Control Engineering Practice*, vol. 78, pp. 196–212, 2018.

[44] Y. Dong, J. Tao, Y. Zhang, W. Lin, and J. Ai, "Deep learning in aircraft design, dynamics, and control: review and prospects," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 4, pp. 2346–2368, 2021.

[45] H. A. Talebi, K. Khorasani, and S. Tafazoli, "A recurrent neural-network-based sensor and actuator fault detection and isolation for nonlinear systems with application to the satellite's attitude control subsystem," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 45–60, 2008.

[46] M. Chen, P. Shi, and C.-C. Lim, "Adaptive neural fault-tolerant control of a 3-DOF model helicopter system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 2, pp. 260–270, 2015.

[47] E. Sobhani-Tehrani, H. A. Talebi, and K. Khorasani, "Hybrid fault diagnosis of nonlinear systems using neural parameter estimators," *Neural Networks*, vol. 50, pp. 12–32, 2014.

[48] L. Chen, J. Cao, K. Wu, and Z. Zhang, "Application of generalized frequency response functions and improved convolutional neural network to fault diagnosis of heavy-duty industrial robot," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102228, 2022.

[49] R. M. Souza, E. G. Nascimento, U. A. Miranda, W. J. Silva, and H. A. Lepikson, "Deep learning for diagnosis and classification of faults in industrial rotating machinery," *Computers & Industrial Engineering*, vol. 153, p. 107060, 2021.

[50] Y. Dong, "An application of deep neural networks to the in-flight parameter identification for detection and characterization of aircraft icing," *Aerospace Science and Technology*, vol. 77, pp. 34–49, 2018.

[51] Y. Dong, "Implementing deep learning for comprehensive aircraft icing and actuator/sensor fault detection/identification," *Engineering Applications of Artificial Intelligence*, vol. 83, pp. 28–44, 2019.

[52] B. Eroglu, M. C. Sahin, and N. K. Ure, "Autolanding control system design with deep learning based fault estimation," *Aerospace Science and Technology*, vol. 102, 2020.

[53] Y. Lin, J.-W. Zhang, and H. Liu, "Deep learning based short-term air traffic flow prediction considering temporal–spatial correlation," *Aerospace Science and Technology*, vol. 93, 2019.

[54] Microsoft, *NNI*https://github.com/microsoft/nni.

[55] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[56] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

[57] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-CAM++: generalized gradient-based visual explanations for deep con-volutional networks," in *2018 IEEE winter conference on applications of computer vision (WACV)*, pp. 839–847, IEEE, 2018.

[58] D. Omeiza, S. Speakman, C. Cintas, and K. Weldermariam, "Smooth Grad-CAM++: an enhanced inference level visualization technique for deep convolutional neural network models," http://arxiv.org/abs/1908.01224.

[59] H. Wang, Z. Wang, M. Du et al., "Score-CAM: score-weighted visual explanations for convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 24-25, 2020.

[60] H. Wang, R. Naidu, J. Michael, and S. S. Kundu, "SS-CAM: smoothed score-CAM for sharper visual feature localization," http://arxiv.org/abs/2006.14255.

[61] S. Desai and G. Ramaswamy, "Ablation-CAM: visual explanations for deep convolutional network via gradient-free localization," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 983–991, 2020.

[62] Y. Dong, Y. Zhang, and J. Ai, "Full-altitude attitude angles envelope and model predictive control-based attitude angles protection for civil aircraft," *Aerospace Science and Technology*, vol. 55, pp. 292–306, 2016.

[63] L. Höhndorf, J. Siegel, J. Sembiring, P. Koppitz, and F. Holzapfel, "Reconstruction of aircraft states during landing based on quick access recorder data," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2393–2398, 2017.

[64] R. C. Nelson, *Flight Stability and Automatic Control*, vol. 2, WCB/McGraw Hill New York, 1998.

[65] V. Klein, T. R. Ratvasky, and B. R. Cobleigh, *Aerodynamic parameters of high-angle-of attack research vehicle (HARV) estimated from flight data*, NASA, NASA-TM-102692, 1990.

[66] Department of Defense, *Flying Qualities of Piloted Aircraft*, 1997, Accessed October 2020 URL https://cafe.foundation/v2/pdf_tech/Flying.Qualities/PAV.FlyQual.Mil1797A.pdf.

[67] I. Goodfellow, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1, MIT press Cambridge, 2016.

[68] Y. Dong, "Deep learning-based opponent aircraft attitude detection in autonomous air combat," *Journal of Aerospace Information Systems*, vol. 16, no. 4, pp. 162–167, 2019.

[69] M. Lin, Q. Chen, and S. Yan, *Network in network*, http://arxiv.org/abs/1312.4400.

[70] M. Tan and Q. V. Le, *Mixconv: Mixed depthwise convolutional kernels*http://arxiv.org/abs/1907.09595.