Journal of Engineering for Gas Turbines and Power,
Volume 145, Issue 6, June 2023, Article number 061013, Paper number GTP-22-1234
DOI:10.1115/1.4056128

Convolutional neural network denoising autoencoders for intelligent aircraft engine gas path health signal noise filtering

Junjie Zhao[1]
Center for Propulsion and Thermal Power Engineering, Cranfield University, Bedfordshire, MK43 0AL, UK, junjie.zhao@cranfield.ac.uk.

Yi-Guang Li
Center for Propulsion and Thermal Power Engineering, Cranfield University, Bedfordshire, MK43 0AL, UK, i.y.li@cranfield.ac.uk.

Suresh Sampath
Center for Propulsion and Thermal Power Engineering, Cranfield University, Bedfordshire, MK43 0AL, UK, s.sampath@cranfield.ac.uk.

Abstract – Removing noise from health signals is critical in gas path diagnostics of aircraft engines. An efficient noise filtering/denoising method should remove noise without using future data points, preserve important changes, and promote accurate diagnostics without time delay. Machine Learning (ML)-based methods are promising for high fidelity, accuracy, and computational efficiency under the motivation of Intelligent Engines. However, previous ML-based denoising methods are rarely applied in actual engineering practice because they cannot accommodate time series and cannot effectively capture important changes or are limited by the time delay problem. This paper proposes a Convolutional Neural Network Denoising Autoencoder (CNN-DAE) method to build a denoising autoencoder structure. In this structure, a convolutional operation is used to accommodate time series, and causal convolution is introduced to solve the problem of using future data points. The proposed denoising method is evaluated against NASA's Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) software. It has been proved that the proposed method can accommodate time series, remove noise for improved denoising accuracy and preserve the important changes for enhanced diagnostic information. NASA's blind test case results show that Kappa Coefficient of a common diagnostic method using the processed data is 0.731 and is at least 0.046 higher than the other diagnostic methods in the open literature. Processing health signals using the proposed method would significantly promote accurate diagnostics without time delay. The proposed method could support intelligent condition monitoring systems by exploiting historical information for improved denoising and diagnostic performance.

Key Words: Aircraft engine diagnostics; Time-series health signals; Noise filtering; Convolutional Neural Network Denoising Autoencoders.

1. INTRODUCTION

Gas turbine performance deterioration may be due to gradual degradation and/or abrupt or rapid faults of an engine [1]. Gradual performance degradation is normally due to fouling, erosion, corrosion, etc. [2], and it evolves on a slow timescale. Faults typically refer to sudden events and failures to the engine, such as Foreign Object Damage (FOD), Domestic Object Damage (DOD), bleed leaks & failures, variable geometry anomalies, actuator & instrumentation faults, and the like [3]. In general, gas path faults can be classified as component (for example, fan, compressor, and turbine) faults and sub-system (for example, sensor and actuator) faults [4]. Based on the evolution rate, faults may also be classified as abrupt faults that appear instantaneously but do not grow in magnitude over time and rapid faults that initiate and grow in magnitude over a short period [3].

Deviations in gas path measurements from an undamaged baseline engine, known as measurement deltas, are usually used for gas path diagnostics. Unfortunately, noise contaminates the measurement deltas, thereby reducing the signal-to-noise ratio [5]. This can hide key features in the signal. In addition, the key features always change with time

[1] Corresponding author: Junjie Zhao (Email: junjie.zhao@cranfield.ac.uk).

and are contained in windows of time-series data. A key objective of gas turbine diagnostics is to determine faults' existence and location from the noisy data. An efficient denoising method should remove noise without using future data points, preserve important changes in time-series data, and promote accurate diagnostics without time delay.

In signal processing, filtering methods are used to process the data. Traditionally, filtering methods used by the gas turbine industry are moving average (MA) [6,7] and exponential moving average (EMA) [8,9]. MA is a special case of the finite impulse response (FIR) filter, and EMA is a special case of the infinite impulse response (IIR) filter. Linear filters such as the FIR filters and the IIR filters can distort the sharp changes in the signals and they are also weak at outlier removal [10]. Details about both the FIR and IIR filters and their limitations for gas turbine health signal denoising are discussed by Ganguli [5]. Substantial research efforts have been conducted to find suitable alternatives to linear filters that are robust or resistant to the presence of impulsive noise. Among these works, nonlinear filters such as median filters have been proposed for noise removal from gas turbine signals. Median filters, such as FIR median hybrid (FMH) filters [11], center weighted idempotent median (CWIM) filters [12], and recursive median (RM) filters [10,13,14], can preserve edges while simultaneously reducing noise. A disadvantage is the diagnostic time delay, as median filters must use future data points. Details about median filters and their limitations for gas turbine health signal denoising are discussed by Uday et al. [10].

ML-based methods are promising under the motivation of Intelligent Engines. ML-based denoising methods can likewise be classified as nonlinear filters. Auto-Associative Neural Network (AANN), also called autoassociator, autoencoder, or Diabolo Network [15], is a special neural network (NN) that can be used for denoising. The name Auto-Associative Neural Network is commonly used in gas turbine gas path diagnostics. The concept of using a NN with a bottleneck to concentrate information has been firstly discussed in the context of "encoder/decoder" problems [16]. They have primarily been used to extract sparse internal representations of any input and reduce its dimensionality. Vincent et al. [17] introduced denoising autoencoder as an extension to classic autoencoders that is robust to noise. Noise filtering using autoencoders was introduced much earlier. The concept of AANN was introduced by Kramer [18] and was used for noise filtering, sensor replacement and gross error detection and identification. AANN was introduced to gas turbine diagnostics by Guo et al. [19] for sensor validation. Then, Lu et al. [20] used AANN for filtering gas path measurement noise. Other studies have also been conducted on AANN in sensor fault diagnostics and noise filtering [21,22]. The name autoencoder is always used in the context of deep network framework [17]. Hence, in this research, the name denoising autoencoder is used to describe the autoencoder for noise filtering, and the name conventional denoising autoencoder is used instead of AANN.

In ML-based gas turbine noise filtering, conventional denoising autoencoders are commonly used [5]. The number of input and output nodes for a conventional denoising autoencoder equals the number of measurements. Tensors are the basic data structure for all current ML systems. As containers for data, tensors are defined with the number of axes (rank), shape, and data type. One input sample for a conventional denoising autoencoder is a One-dimensional (1D) tensor, which consists of the measurements from selected sensor observations at a discrete-time instant. However, by using conventional denoising autoencoders, changes in the signals that evolve with time may not be adequately captured, resulting in poor noise filtering performance.

To accommodate time-series data, denoising autoencoders in a deep network setting that use windows of multivariate measurements (2D tensors) are investigated in this paper. The two fundamental deep-learning algorithms for sequence or time series modelling are Recurrent Neural Networks (RNNs) [23] and Convolutional Neural Networks (CNNs) [24]. RNNs are dedicated sequence models that maintain a vector of hidden activations propagated through time. This family of algorithms has gained tremendous popularity due to prominent applications in language modelling and machine translation [25]. Basic RNN algorithms are notoriously difficult to train, and more elaborate algorithms are commonly used instead, such as the Long Short-Term Memory Networks (LSTMs) [26] and the Gated Recurrent Units (GRUs) [27]. In this research, sequence modelling algorithms are integrated into the encoder and decoder of the denoising autoencoder to build a deep denoising autoencoder used for gas turbine gas path measurement noise filtering. By comparison, the denoising autoencoder structure based on CNN is selected to develop the efficient denoising method for gas turbine diagnostics.

The study's contribution is the novel Convolutional Neural Network Denoising Autoencoder (CNN-DAE) method that can remove noise without using future data points, preserve important changes in time-series data, and promote accurate diagnostics without time delay. Specifically,

- The denoising autoencoder structure can effectively remove the noise in health signals by reconstructing the denoised data from the noisy data.
- The CNN-DAE method can accommodate historical time-series data. The convolution operation can adequately capture changes in the signals that evolve with time.
- The causal convolution is introduced, where no future data points are needed. Accordingly, the diagnostic time delay problem can be solved.

- The proposed CNN-DAE denoising method can preserve the important changes in health signals for enhanced diagnostic information, which significantly improves diagnostic accuracy.

It is noticed that the scope of the paper is denoising. However, a complete diagnostic solution shall include processing and diagnostic algorithms. To validate the effect of the proposed CNN-DAE denoising method on diagnostic performance, in addition to the necessary case studies on denoising performance, case studies on diagnostic performance are also included. A common ML diagnostic method is introduced in the application section to compare the diagnostic performance with and without CNN-DAE denoising.

The remainder of this paper is organized as follows. Section 2 introduces the methodology to develop the CNN-DAE method. Section 3 describes the case studies using the ProDiMES software from NASA. The results and discussions are provided accordingly in Section 4. Section 5 draws the conclusions.

2. METHODOLOGY

2.1. Set Up Denoising Autoencoders

A denoising autoencoder is a NN that receives a corrupted data point as input and is trained to predict the original, uncorrupted data point as its output. A denoising autoencoder includes three hidden layers: the mapping layer, the bottleneck layer and the demapping layer, as shown in Fig. 1. The mapping phase where the input x is transformed into the hidden representation s is termed an encoder. A decoder is the part of an autoencoder where the input is reconstructed back to y from its hidden representation s. In Fig. 1, the encoding and decoding functions are denoted by $f_\theta(\cdot)$ and $g_{\theta'}(\cdot)$, respectively, and these mappings are correspondingly parametrized by vectors θ and θ'. Typically, the mapping functions comprise of affine mapping followed by certain nonlinearity. $\theta = \{W, b\}$ and $\theta' = \{W', b'\}$ are parameter sets with W and W' denoting the weight matrices and b and b' representing the bias vectors.

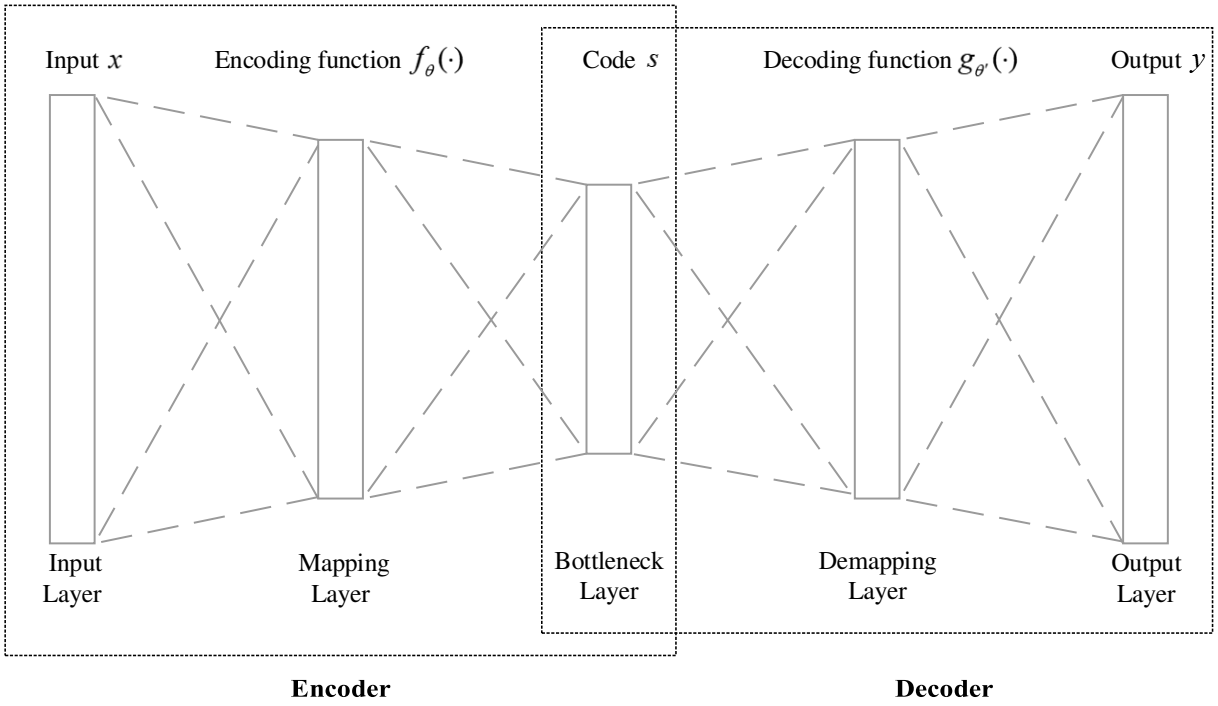


Fig. 1. The architecture of a denoising autoencoder with three hidden layers.

The goal of the autoencoder presented in Fig. 1 is to predict an uncorrupted output with the minimum reconstruction loss. The autoencoder presented in Fig. 1 has three hidden layers. However, more hidden layers can be used for autoencoders of higher complexity. Accordingly, the encoder and decoder will comprise a series of mappings in each.

2.2. Set Up CNN-DAEs

CNN-DAEs are denoising autoencoders that use convolution in place of general matrix multiplication in at least one of the hidden layers. The name "Convolutional Neural Network (CNN)" indicates that the network employs a mathematical operation called convolution. In this research, to accommodate 2D data of shape (timesteps, measurements), which can be viewed as a 1D grid taking samples of shape (measurements,) at a regular time interval, One-dimensional Convolutional Neural Networks (1DCNNs) [28,29] are used.

A CNN hidden layer contains a few functions from the convolution, pooling, and nonlinear activation [30]. The pooling function provides an approach to down sample, which produces invariance to local translation. Pooling will not be used in the proposed CNN-DAE because it is not useful when priority is on temporal order and the feature location. Meanwhile, pooling can complicate the autoencoder architectures that use top-down information.

2.2.1. Convolution Operation for Signal Processing

Convolution operates on two functions of a real-valued argument in its most general form. In the research context, suppose a sensor provides a single output $x(t)$ at time t and the sensor is somewhat noisy. From the point of signal processing, a weighting operation that is similar to the weighting coefficients in an FIR filter [5] is used to filter the measurements. If a weighted operation $w(a)$ is applied at every moment, a new function y providing a smoothed estimate is obtained:

$$y(t) = \int x(a)w(t-a)da \tag{1}$$

This operation is termed convolution. The convolution operation is typically denoted with an asterisk:

$$y(t) = (x * w)(t) \tag{2}$$

The first argument (in this case, the function x) is generally referred to as the input. The second argument (in this case, the function w) is commonly referred to as the kernel. The output is usually referred to as the feature map.

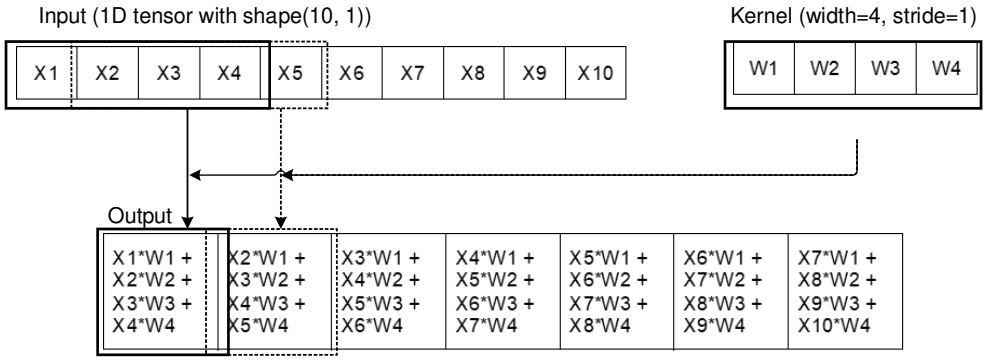


Fig. 2. An example of convolution operation [28].

In reality, a sensor can not produce measurements at every instant in time. Generally, when dealing with data on a computer, time is discretized, and one sensor will produce data at regular intervals. In this case, a more realistic assumption could be that one sensor produces a measurement, for example, once per second. Then, the time index t will only be able to take on integer values. If it is assumed that x and w are defined only on integer t, the discrete convolution can be described as:

$$y(t) = (x * w)(t) = \sum_a x(a)w(t-a) \tag{3}$$

Since the convolution operation is commutative, the discrete convolution can be equivalently written as:

$$y(t) = (w * x)(t) = \sum_a x(t-a)w(a) \tag{4}$$

Generally, many NN libraries implement the cross-correlation function but call it convolution. In this research, the convention of calling both operations convolution is followed. An example of convolution (without kernel flipping) is shown in Fig. 2. The input is a 1D tensor of shape (10,). The kernel size is four. The other factor that can influence convolution is the notion of strides. The description of convolution so far has assumed that the center tiles of the convolution windows are all contiguous. But the distance between two successive windows is a parameter of the convolution, called its stride. The length of the 1D convolution window is four, and the stride length of the convolution is one. Discrete convolution can be viewed as multiplication by a matrix.

2.2.2. Causal Convolution

In a CNN-DAE, some constraints and modifications may be required. An important constraint is that the model cannot violate the ordering in which the data is modelled: the prediction $p(x_{t+1}|x_1,\ldots,x_t)$ emitted by the model at timestep t cannot depend on any of the future timesteps $x_{t+1}, x_{t+2},\ldots,x_T$. Fig. 3 shows an example of modelling a 1D tensor with two conventional convolutional hidden layers. The input and output shapes are both (5, 1). For the convolution operation, the kernel size is three, and the stride is one. To ensure the input and output are of the same shape, zero-padding is applied evenly to the left and right of the input. Zero-padding means adding zeros to the edge of the input matrix [30]. The output from conventional convolution depends on some of the future timesteps.

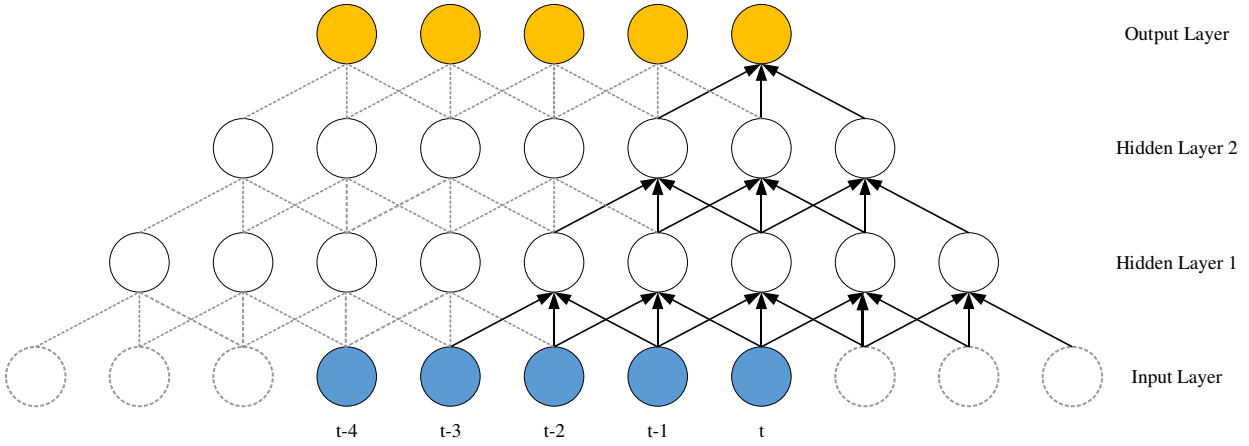


Fig. 3. An example of conventional convolutional hidden layers.

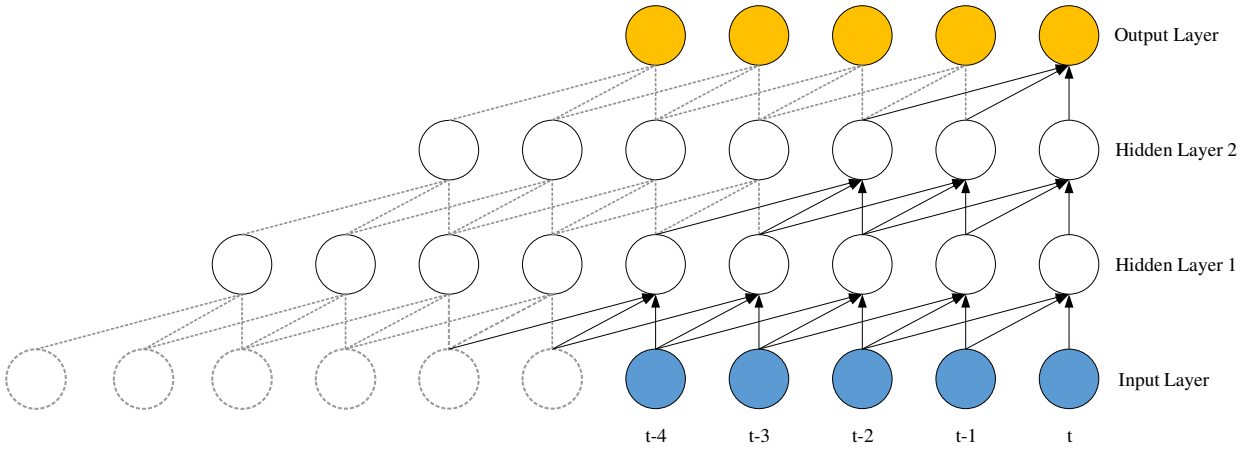


Fig. 4. An example of causal convolutional hidden layers.

To address the time delay issues, the 1D causal convolution adapted from WaveNet [31] is used in this research. In the causal convolution, output at timestep t is convolved only with elements from time t and earlier, without depending on input at next timestep $t + 1$. This is implemented by shifting the output of a normal convolution by a few timesteps for the 1D tensor. Fig. 4 shows an example of modelling the same input using a two hidden layer causal CNN. The kernel size for the convolution operation is three and the stride is one. Zero-padding is applied to the left of the input.

2.3. Training of CNN-DAEs

A single input of a training sample is a 2D tensor with shape (timesteps, features), which is a window of multivariate health signals with noise. A single output of a training sample has the same shape as the input, and it is a window of multivariate health signals without noise.

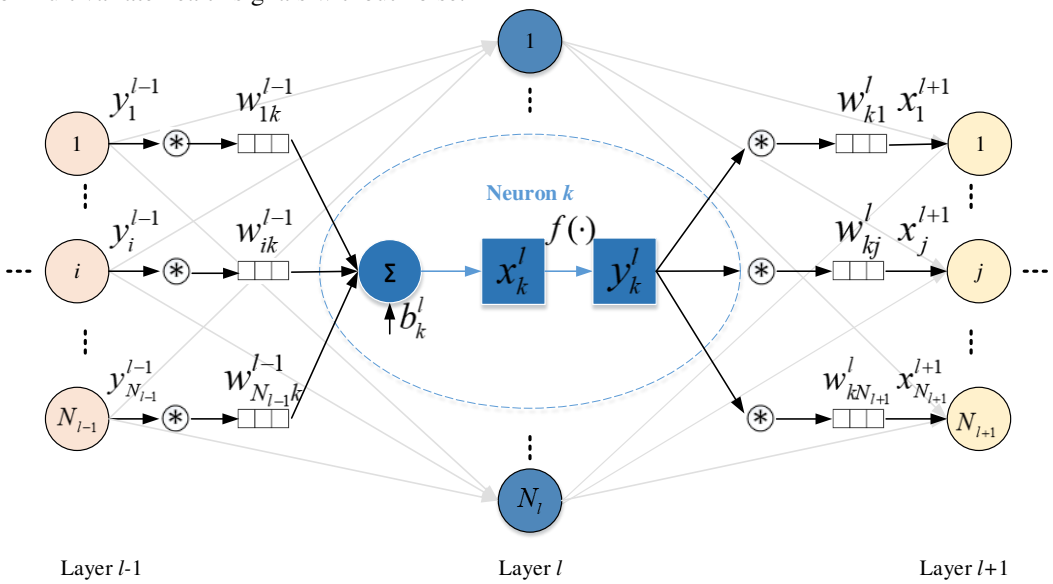


Fig. 5. A typical hidden layer consists of convolution and activation functions.

As shown in Fig. 5, a hidden layer of a CNN-DAE consists of convolution operation and nonlinear activation function in turn. The final output of the k^{th} neuron at layer l is y_k^l. In each layer, one-dimensional forward propagation (1D-FP) is expressed as follows:

$$y_k^l = f\left(x_k^l\right) \quad where \quad x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} conv1D\left(w_{ik}^{l-1}, y_i^{l-1}\right) \tag{5}$$

Where conv1D (.,.) is a 1D causal convolution with zero-padding on the boundaries, x_k^l is the input, b_k^l is the bias of the k^{th} neuron at layer l, and y_i^{l-1} is the output of the i^{th} neuron at layer $l-1$. w_{ik}^{l-1} is the kernel (weight) from the ith neuron at layer $l-1$ to the k^{th} neuron at layer l.

Let $l = 1$ and $l = L$ be the input and output layers, respectively. For an input vector p, and its corresponding output vector, $[y_1^L, \dots, y_{N_L}^L]$, let $[t_1, \dots, t_{N_L}]$ be the target class vector. The mean absolute error (MAE) in the output layer can then be expressed as

$$\mathrm{E} = \mathrm{E}\left(y_1^L, \dots, y_{N_L}^L\right) = \frac{\sum_{i=1}^{N_L} |y_i^L - t_i|}{N_L} \tag{6}$$

Two more elements are stored for each neuron: the delta error $\Delta_k^l = \frac{\partial E}{\partial x_k^l}$ and the derivative of the intermediate output $f'(x_k^l)$, to accomplish Back Propagation (BP) training.

Consequently, the iterative flow of the BP in the training set can be stated as follows:

a) Initialize weights and biases of the network.
b) For each BP iteration, do as follows:
   i. FP: Forward propagate from the input layer to the output layer to find outputs of each neuron at each layer, $y_i^l \forall i \in [1, N_l]$, and $\forall l \in [1, L]$.
   ii. BP: Compute delta error at the output layer and back-propagate it to the first hidden layer to compute the delta errors, $\Delta_k^l \ \forall k \in [1, N_l]$, and $\forall l \in [1, L]$.
   iii. PP: Post-process to compute the weight and bias sensitivities.
   iv. Update the weights and biases with the (accumulation of) sensitivities.

When the training process is complete, the well-trained CNN-DAE is ready for noise filtering.

3. APPLICATION

3.1. Case Study Description

The investigated case study is based on the ProDiMES software, which provides a standard benchmark problem enabling users to develop, evaluate, and compare diagnostic methods. An introduction to ProDiMES is in reference [4], and detailed instructions on its application can be found in reference [8].

Two types of case studies are conducted in this section to evaluate the performance of the proposed CNN-DAE denoising method on noising filtering and diagnostics. To evaluate the effect of the proposed CNN-DAE denoising method on diagnostic performance. A single flat MLP classifier is developed for fault diagnostics and details are described in Section 4.2.

3.2. Data Generation and Processing

3.2.1. Data Generation

An Engine Fleet Simulator (EFS) in ProDiMES based on a steady-state version of the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) high-bypass two-spool turbofan engine simulation is used to generate the data for this study. The EFS produces simulated "snapshot" engine measurements, with relevant measurement noise, as if collected from a fleet of engines over multiple flights. Within ProDiMES, engine operating conditions, deterioration profiles, fault magnitudes, and sensor noise are randomly generated to emulate realistic behaviour.

Four sets of data were used in this study. The first data set is the training data to develop denoising autoencoders for noise filtering. It includes noisy and noise-free measurements from a fleet of 18963 engines conducting 50 flights each. The second data set includes a fleet of 1896 engines, conducting 50 flights each to train and validate the diagnostic algorithms. The description of the second data is shown in Table 1. It is worth noting that the training data size could be modified during the training optimization. The third data set is the test data that includes a fleet of 9993 engines conducting 50 flights each. The fourth data set includes the blind test case data (i.e., a data set where the true fault state of the engines contained in the data set is unknown to the end-users) from NASA. It is noted that only takeoff data is used in the case studies.

Table 1 Description of the initial training data from the second data set.

Name	Value
Number of engine health conditions	19
Number of engines per health condition	100
Number of engines that do not converge	4
Number of flights per engine	50
Fault initiation	Random
Minimum initiation flight	11
Fault evolution	Random
Rapid fault evolution rate (minimum)	9
Rapid fault evolution rate (maximum)	9
Sensor noise	On

3.2.2. Data Processing

Data processing consists of parameter correction, gradual deterioration elimination, data standardization, and noise filtering.

The parameter correction and deterioration elimination methods in this case study are identical to the methods given in ProDiMES User's Guide [4]. As an initial step of data processing, all engine measurement data are corrected to standard ISA condition at sea level to eliminate ambient conditions' impact on the measurement data variations. Then, a gradual deterioration trend monitoring approach is applied to capture the gradual performance changes in the form of residuals, or measurement deltas, relative to a fleet average engine model or 50 percent deteriorated engine. For each individual engine, corrected data collected during each flight are then referenced against the fleet average engine model to calculate measurement deltas, $\Delta u_{\alpha,\beta}$, as:

$$\Delta u_{\alpha,\beta}(\gamma) = u_{\alpha,\beta}(\gamma) - u_{\alpha\_baseline}(\gamma) \quad (7)$$

Where $u_{\alpha,\beta}(\gamma)$ is the corrected value of the α^{th} measurement collected on β^{th} engine during the γ^{th} flight, and $u_{\alpha\_baseline}(\gamma)$ is the fleet average engine value for the α^{th} measurement at the corresponding pressure altitude, Mach number, and corrected fan speed values of the γ^{th} flight.

Standardization is the process to transform the data to center it by removing the mean value of each feature, then scaling it by dividing non-constant features by their standard deviation:

$$\Delta v_{\alpha,\beta}(\gamma) = (\Delta u_{\alpha,\beta}(\gamma) - \mu_\alpha)/\sigma_\alpha \quad (8)$$

Where $\Delta v_{\alpha,\beta}(\gamma)$ is the standardscaled value of the α^{th} measurement collected on β^{th} engine during the γ^{th} flight. μ_α is the mean of the α^{th} measurement in the training samples and $\sigma_\alpha = \sqrt{\frac{1}{M}\sum_{\beta=1}\sum_{\gamma=1}\left((\Delta u_{\alpha,\beta}(\gamma) - \mu_\alpha\right)^2}$ is the standard deviation of α^{th} measurement in the training samples. M is the number of α^{th} measurement in the training samples. Approximate standard normally distributed data are obtained with data standardization.

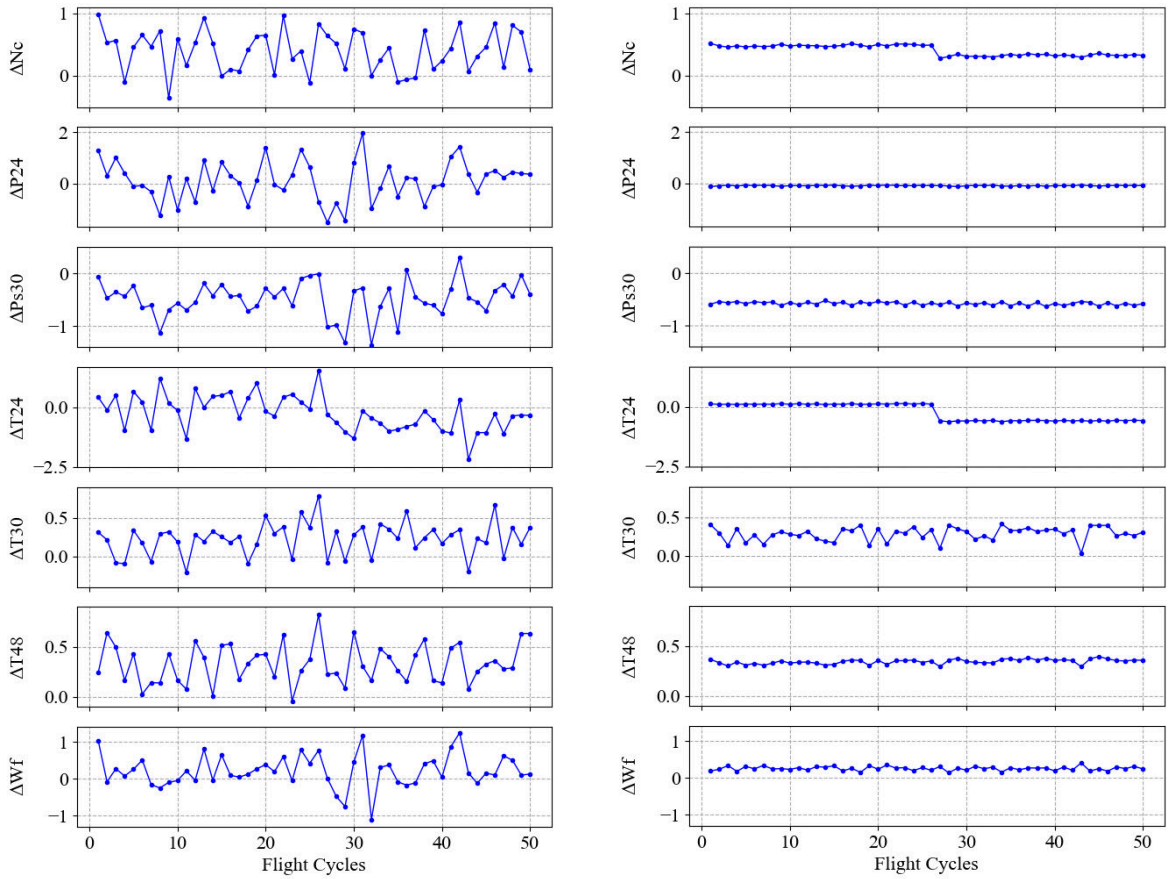
3.3. Noise Filtering with Denoising Autoencoders

The training data set for the denoising autoencoders (the first set of data) has 18963 training samples (inputs), and each sample is a 2D tensor with shape (50, 7). The training data set for the denoising autoencoders also has 18963 training outputs, and each output is a 2D tensor with shape (50, 7). The shape of the 2D tensor means each engine conducts 50 flights and seven sensors are used for each flight. Fig. 6 shows an example of a takeoff training sample with an abrupt T24 sensor fault emerging at flight cycle 27 and the corresponding noise-free output. The data shown has been processed with parameter correction, deterioration elimination, data standardization.

Table 2 Hyperparameter optimization results.

Hyperparameters	Searching ranges	Optimal result
Convolutional layer number	(1, 6, 1)	3
Kernel number	(50, 200, 10)	150/ 70/ 150
Kernel size	(2, 10, 1)	7/ 5/ 7
Optimizer	(rmsprop, sgd, adam)	adam
Learning rate of the optimizer	default	default
Activations	(relu, sigmoid, tanh)	relu
Last-layer activation	linear	linear
Loss function	(mse, mae)	mae
Epochs	1000	1000
Batch size	(1, 2000, 1)	72
Early stopping	(yes, no)	yes
Evaluation protocols	(hold-out validation, K-fold cross-validation, iterated K-fold validation)	K-fold cross-validation

An initial CNN-DAE with three 1D convolution layers is developed. Then, an optimization process is conducted to decide the model size and hyperparameters. Bayesian Optimization [32] is used to optimize the denoising autoencoders. In the Bayesian Optimization method, firstly, a domain of hyperparameters is decided. In this case, the searching ranges for hyperparameters are defined in the second column of Table 2 to form a searching space. Secondly, an objective function takes in hyperparameters and outputs a score that indicates how well a set of hyperparameters performs on the validation set. In this case, MAE is chosen as the objective function. Thirdly, the next set of hyperparameters is selected based on a model of the objective function called a surrogate. Finally, each time the algorithm proposes a new set of candidate hyperparameters, it evaluates them with the actual objective function and records the result in a pair (score, hyperparameters). The best hyperparameters can be selected from the history now. The optimization results are shown in Table 2. Once training and optimization processes have been completed, the well-trained denoising autoencoder is ready to be used for testing.



(a) Noisy Standardscaled Residual Data (b) Noise-free Standardscaled Residual Data

Fig. 6. A takeoff training sample and the output with a -1.47σ abrupt T24 sensor fault at flight 27.

4. RESULTS AND DISCUSSION

This section consists of two subsections. Section 4.1 states the test case results on the denoising performance of the proposed CNN-DAE denoising method. Section 4.2 states the blind test case results that evaluate the effect of the proposed CNN-DAE denoising method on diagnostic performance.

4.1. Test Case Results

The second data set described in Section 3.2 includes a fleet of 1896 engines, conducting 50 flights each to test the denoising methods. In this case study, noisy and noise-free takeoff data are used as the input and corresponding output.

Two types of error criteria are used to obtain a quantitative idea of noise reduction. The MAE measures the difference between the filtered and the noise-free data. In the MAE criterion, the error is defined as:

$$MAE = \frac{\sum_{m=1}^{M} |y_m - x_m|}{M} \tag{9}$$

Where M is the number of measurements in the data set. y_m is the filtered measurement and x_m is the noise-free measurement.

A parameter called noise reduction rate, which is used for the efficiency measurement of these filters in terms of noise reduction is also used. The noise reduction rate is defined as:

$$\rho = \frac{MAE^{(noisy)} - MAE^{(filtered)}}{MAE^{(noisy)}} \times 100(\%) \tag{10}$$

4.1.1. Comparison with Conventional Denoising Autoencoders

Conventional denoising autoencoder is the most widely used ML method in gas turbine gas path measurement noise filtering. Multilayer Perceptrons (MLPs), also often called feedforward neural networks or deep feedforward networks [30], are the quintessential learning models for conventional denoising autoencoders. An MLP-based denoising autoencoder (MLP-DAE) is developed and compared with the CNN-DAE. A typical MLP hidden layer is shown in Fig. 7. Feedforward propagation is expressed as follows:

$$y_k^l = f(x_k^l) \quad where \quad x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} w_{ik}^{l-1} y_i^{l-1} \tag{11}$$

The training samples of an MLP can only be 2D tensors (matrices) of shape (samples, features). A single input sample is a 1D tensor (vector), which means there can only be one 0D tensor (scalar) for an input layer neuron. The number of input layer neurons of a typical MLP-DAE equals the number of the measurements. There should be seven neurons for the typical MLP-DAE input layer in the case study corresponding to the seven sensor measurement features. After training and optimization, a well-trained MLP-DAE with the shape of 7-32-6-32-7 is obtained and is named MLP1-DAE.

MLP1-DAE cannot consider temporal order; hence, the 3D tensors of shape (samples, timesteps, features) must be "flattened" as 2D tensors of shape (samples, timesteps*features). Details of the flatten process can be found in reference [28]. There should be 350 neurons for the flattened MLP-DAE input layer in the case study. After training and optimization, a well-trained MLP-DAE with the shape of 350-100-10-100-350 was obtained and was named MLP2-DAE. However, MLP2-DAE would induce a time delay because of using future data points. It is noted that the data was processed with the parameter correction and data standardization processes before the noise filtering process.

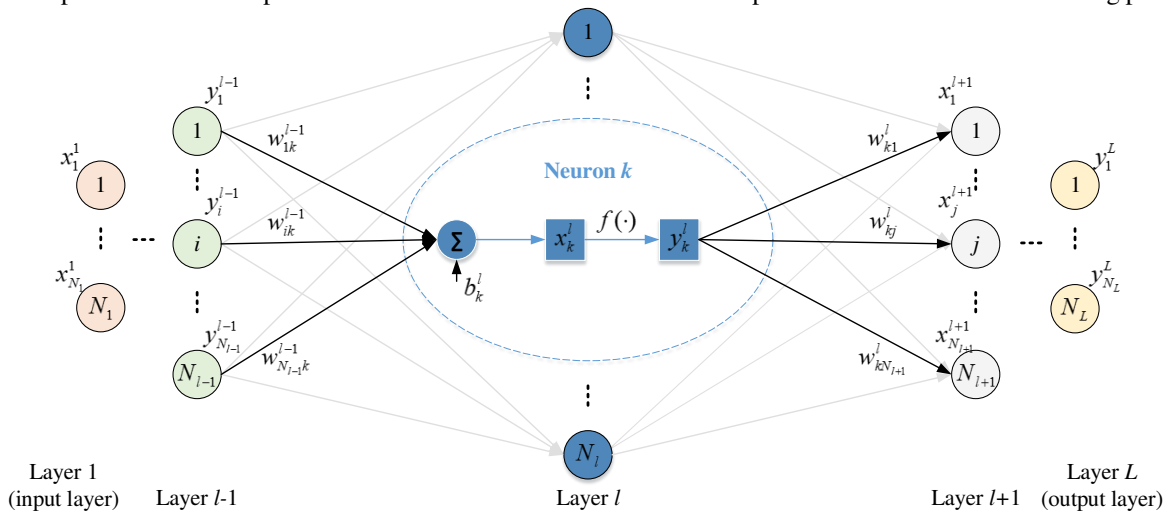


Fig. 7. Typical MLP hidden layers.

Fig. 8 and Fig. 9 visually represent the effects of MLP1-DAE and CNN-DAE on two samples in the test data set, respectively. It illustrates that MLP1-DAE has poorer denoising performance when faults emerge, especially when the fault magnitudes are comparable with the noise magnitudes. MLP1-DAE has good performance in learning the information from discrete snapshots. However, continuous 2D data representing the initiation and growth in magnitude of a fault event is needed. Using the conventional denoising autoencoders, changes in the signals that evolve with time may not be adequately captured, eventually resulting in bad denoising performance.

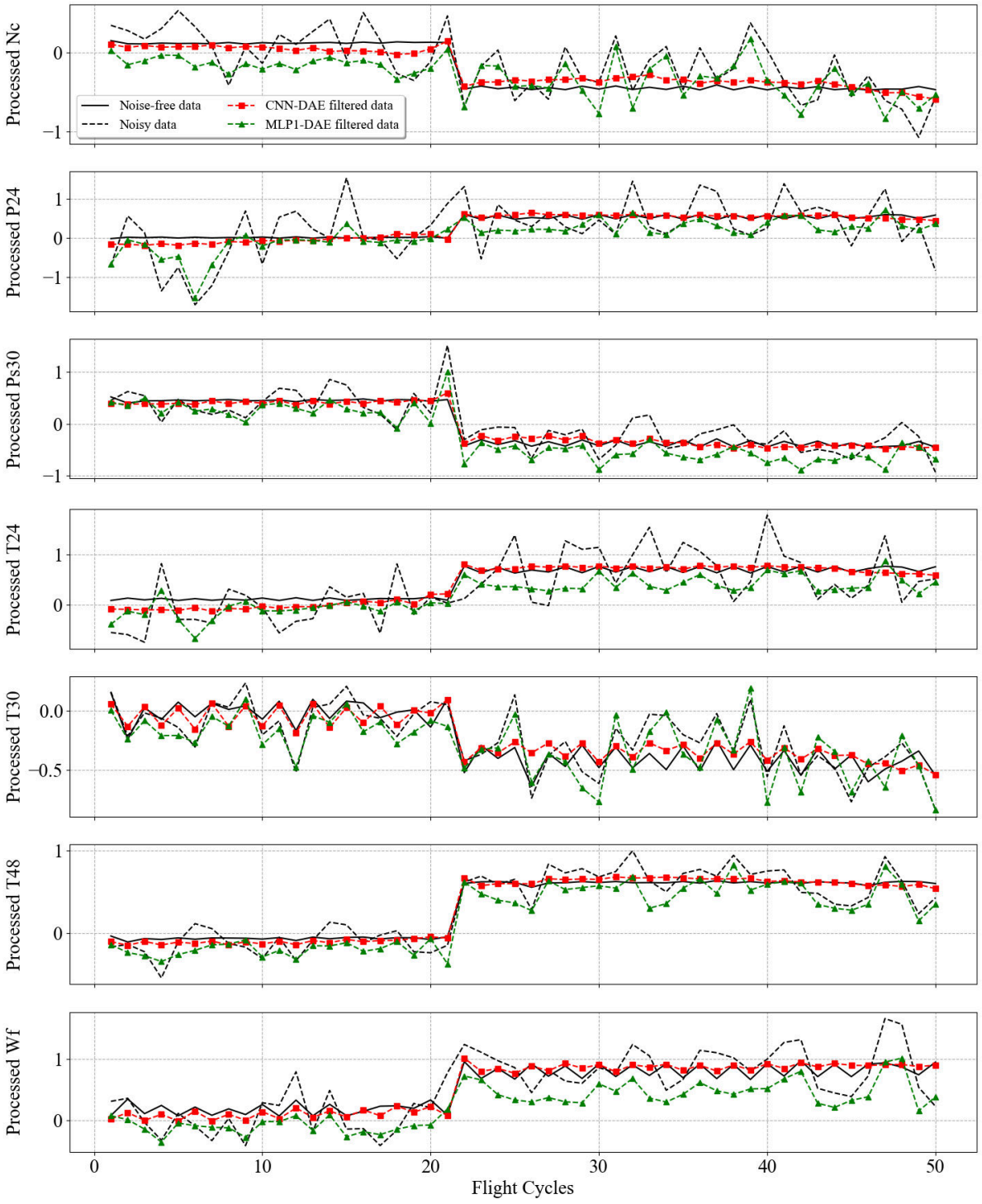


Fig. 8. Effect of MLP1-DAE on noisy takeoff data (1.57% HPT component abrupt fault at flight cycle 22).

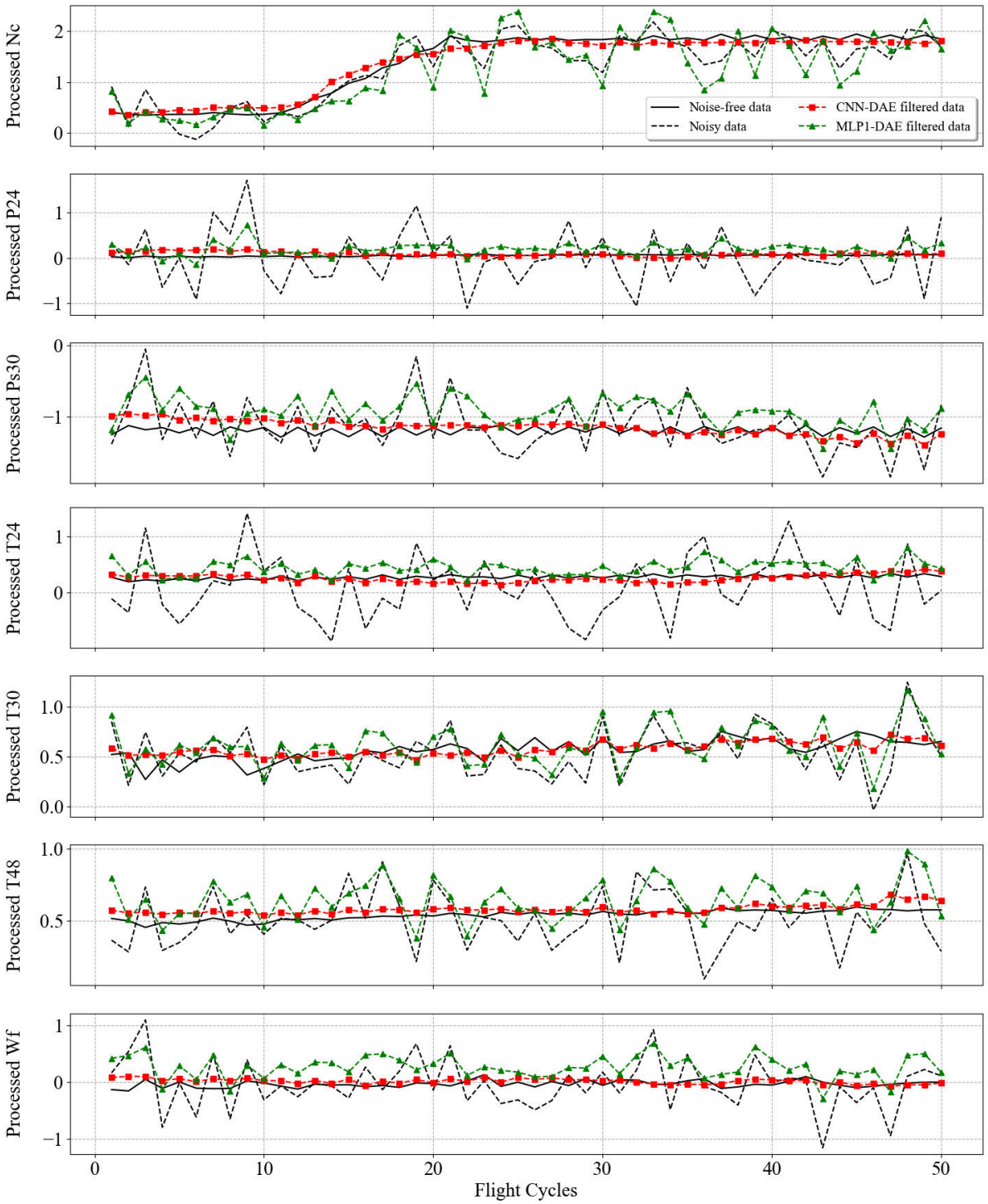


Fig. 9. Effect of MLP1-DAE on noisy takeoff data (1.34% VSV actuator rapid fault at flight cycle 12).

Table 3 shows a comparison of the denoising performance of the CNN-DAE against the other methods considered in this study. As is shown in Table 3, with the flattened data, MLP2-DAE offers better denoising performance than MLP1-DAE. However, the denoising performance of the flattened methods will be worse with the increase of the sample length because the model and computation complexity are increased for the MLPs.

Table 3 Denoising performance of different methods.

Method	MAE	ρ [%]
CNN-DAE	0.084	73.91
MLP1-DAE	0.186	42.24
MLP2-DAE	0.099	69.25
EMA	0.187	41.92
WRM	0.155	51.86
LSTM-DAE	0.268	16.77
WaveNet-DAE	0.221	31.37

4.1.2. Comparison with other Optional Signal Processing Methods

EMA filter is the state-of-the-art linear filter, and median filter is the state-of-the-art nonlinear filter. An EMA filter and a weighted recursive median (WRM) filter are developed for comparison purposes in this case study.

The EMA filter is given as:

$$\Delta y_{\alpha,\beta}(\gamma) = \tau \cdot \Delta y_{\alpha,\beta}(\gamma - 1) + (1 - \tau) \cdot \Delta x_{\alpha,\beta}(\gamma) \tag{12}$$

Where $\Delta y_{\alpha,\beta}(\gamma)$ is the EMA of α^{th} measurement collected on β^{th} engine during the γ^{th} flight. The weighting between previous and current data is established by the constant τ $(0 < \tau < 1)$. In the example solution, τ was chosen to be 0.65.

An O-point WRM filter [10] is given as:

$$\begin{aligned}\Delta y_{\alpha,\beta}(\gamma) = median(&w_{\gamma-o} * \Delta y_{\alpha,\beta}(\gamma - o), w_{\gamma-o+1} * \Delta y_{\alpha,\beta}(\gamma - o + 1), \dots, w_{\gamma} \\ &* \Delta x_{\alpha,\beta}(\gamma), \dots, w_{\gamma+o-1} * \Delta x_{\alpha,\beta}(\gamma + o - 1), w_{\gamma+o} * \Delta x_{\alpha,\beta}(\gamma + o))\end{aligned} \tag{13}$$

Where $O = 2o + 1$ is the window length of the filter. $\Delta y_{\alpha,\beta}(\gamma)$ is the output of the median filter. $median()$ is a function that takes o points surrounding the central point and gives their median as the output. A disadvantage of the median filters is that they induce a time delay of o time steps. In the present case, a value of $o = 11$ is selected as it offers the best noise reduction performance.

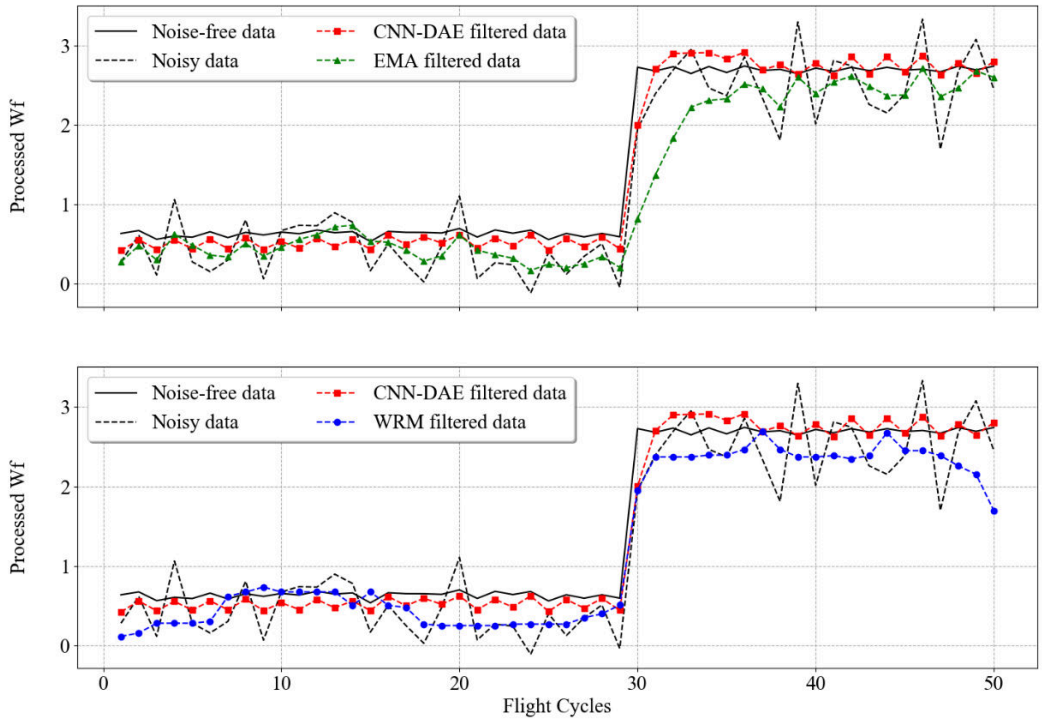


Fig. 10. Effect of EMA and WRM filters (8.16σ Wf sensor abrupt fault at flight 30).

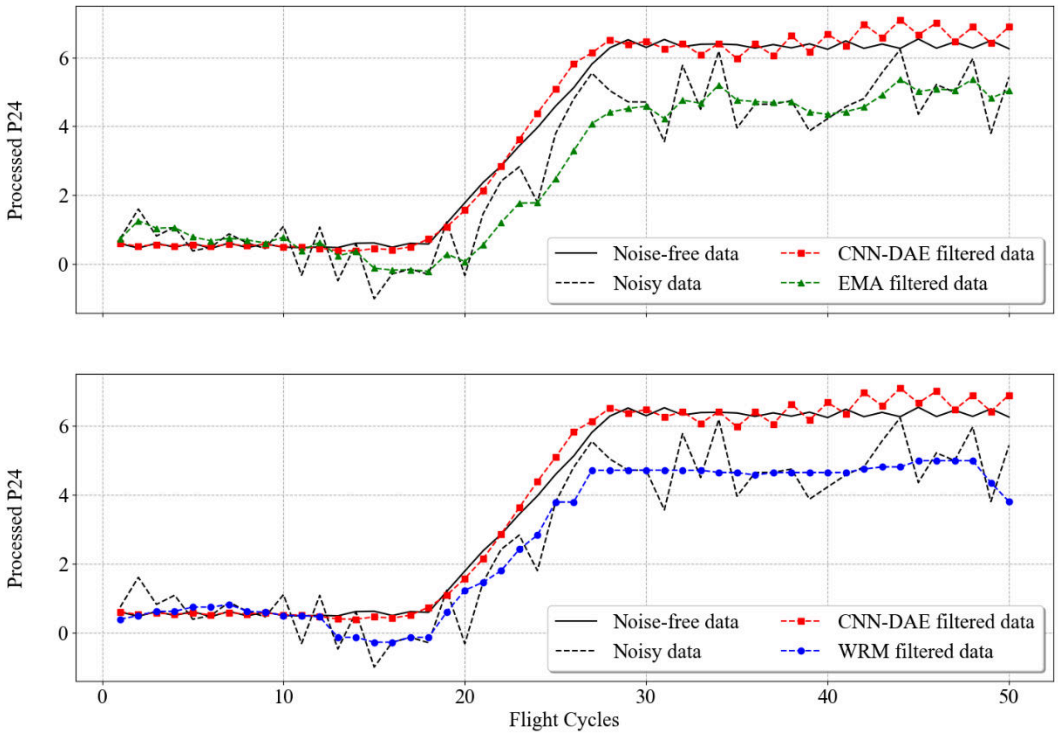


Fig. 11. Effect of EMA and WRM filters (8.91σ P24 sensor rapid fault at flight 19).

The fourth and fifth rows of Table 3 show the denoising performance of the EMA filter and WRM filter for the whole test data set. Fig. 10 and Fig. 11 visually represent the EMA filter and WRM filter effects on two samples of takeoff data in the test data set, respectively. The EMA filter provides a noise reduction of 41.92%, and the MAE of the method is 0.187. EMA filters are often used in gas turbine fault diagnostics to smooth data. But, as is typical in linear filters, they can also smooth out important signal features. The WRM filter can provide a noise reduction of 51.86%, and the MAE of the method is 0.155. The WRM filter performs better than the EMA filter on noise filtering. The WRM filter has better denoising performance by removing the noise while preserving important features such as the changes in the measurements caused by abrupt or rapid faults. Nevertheless, the CNN-DAE provides much better denoising performance than these two signal processing methods.

4.1.3. Comparison with Other Deep Denoising Autoencoders

In this study, sequence modelling algorithms, including LSTM [30] and a special deep NN called WaveNet [31], are used to construct deep denoising autoencoders for comparison. After setting the models and optimization, trained LSTM-based denoising autoencoders (LSTM-DAEs) and WaveNet-based denoising autoencoders (WaveNet-DAEs) were obtained and used in the test case for comparison purposes.

The sixth and seventh rows of Table 3 show the denoising performance of LSTM-DAE and WaveNet-DAE for the whole test data set. Fig. 12 and Fig. 13 visually represent the effects of LSTM-DAE and WaveNet-DAE on two samples of takeoff data in the test data set, respectively. It is observed that LSTM-DAE provides a noise reduction of 16.77% and the MAE of the method is 0.268. The WaveNet-DAE can provide a noise reduction of 31.37%, and the MAE of the method is 0.221. While LSTM and WaveNet are good sequence modelling algorithms, denoising methods based on them show much worse denoising performance than the CNN-DAE.

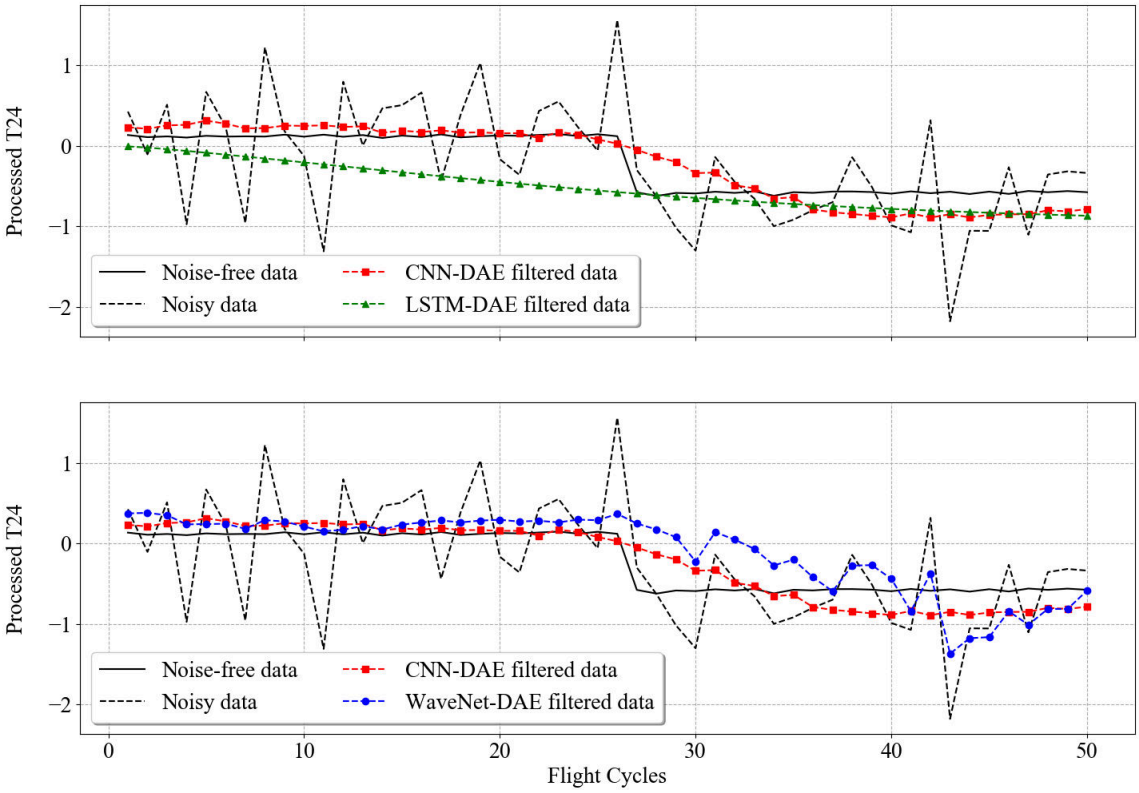


Fig. 12. Effect of LSTM-DAE and WaveNet-DAE (-1.47σ T24 abrupt fault at flight 27).

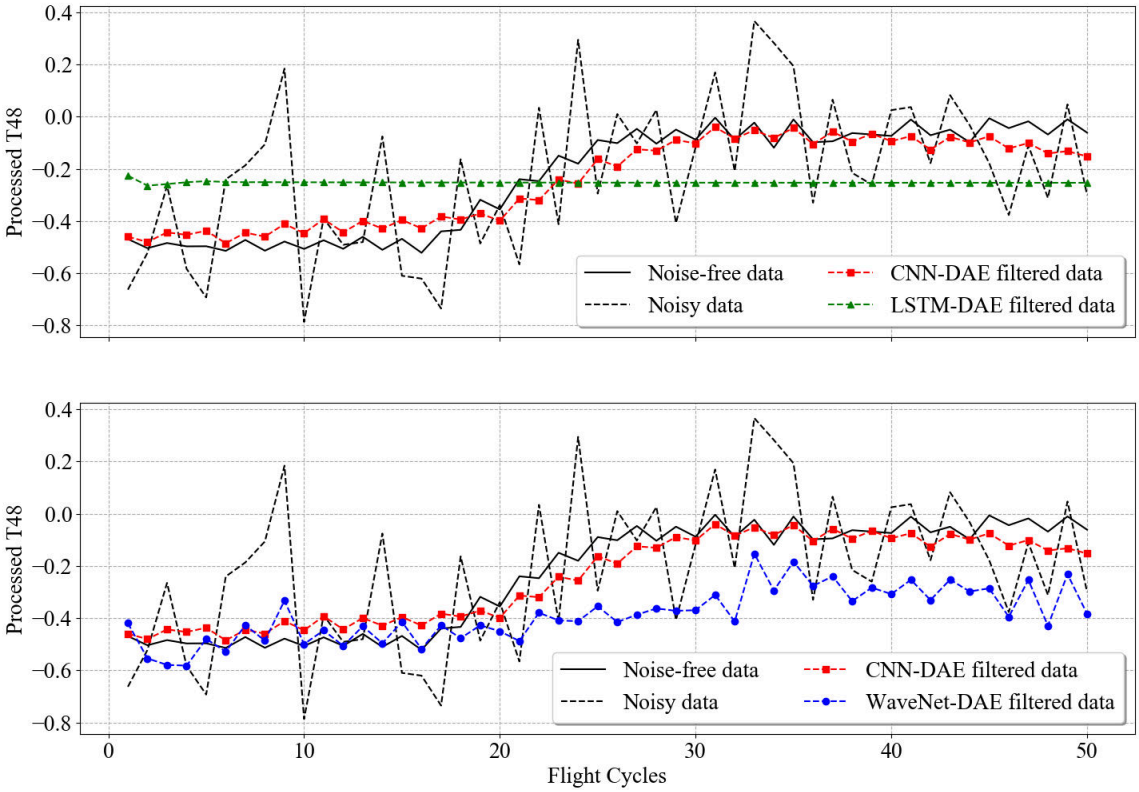


Fig. 13. Effect of LSTM-DAE and WaveNet-DAE (2.40σ T48 rapid fault at flight 17).

4.1.4. Discussion

An overall comparison for all the denoising methods described above is conducted, and the results are shown in Fig. 14 and Fig. 15. Specifically,

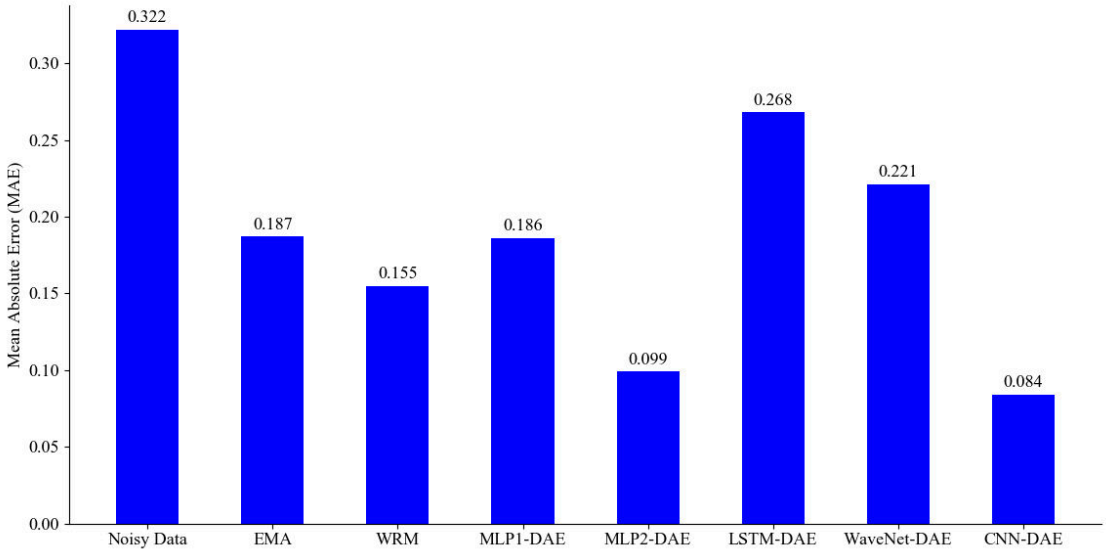


Fig. 14. MAE of different filters for the test data set.

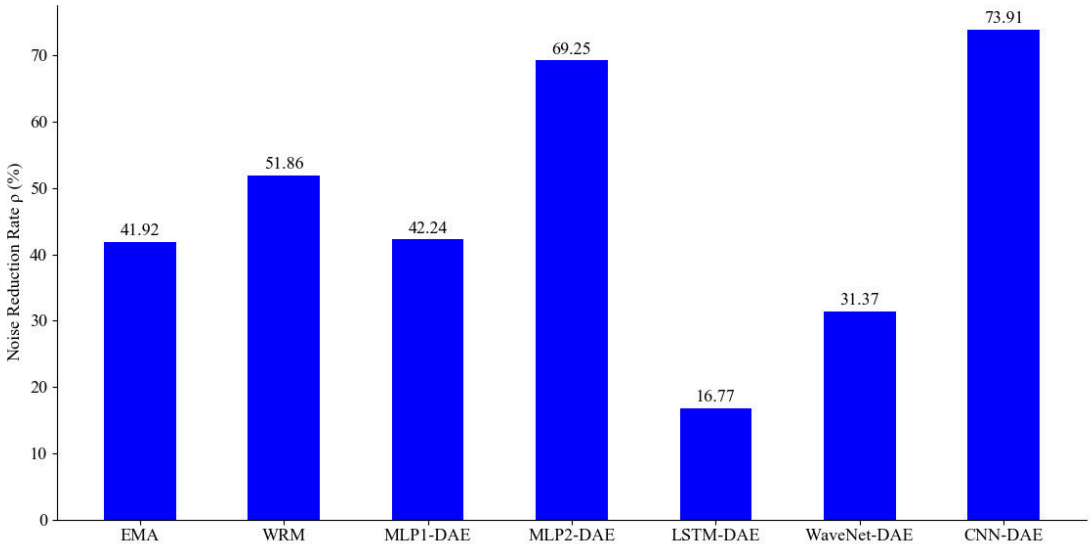


Fig. 15. Noise reduction rate $\rho(\%)$ of different filters for the test data set.

- The EMA filter can reduce noise but also distort the edges in the signal due to the MA process, which results in import feature loss for diagnostics.
- The WRM filter can remove the noise while preserving important features in health signals. A primary disadvantage of the WRM filter is that it induces a time delay, which will result in high Detection Latency.
- While all the other noise filtering methods studied here use 2D tensors, MLP1-DAE is the only method that uses 1D tensors. It tends to smooth out the changes, especially when the change magnitudes are comparable with the noise magnitudes in time-series signals.
- MLP2-DAE can remove noise in the data and perform better at preserving important changes in health signals than the other methods except CNN-DAE. However, like the WRM filter, a disadvantage of MLP2-DAE is the time delay due to using future data points, which results in increased Detection Latency.

- Noise filtering methods based on sequence modelling algorithms, i.e., LSTM-DAE and WaveNet-DAE, are not suitable for this research's noise filtering problem.
- The proposed CNN-DAE method performs best among all these methods in removing noise and preserving important features for diagnostics. It can address the time delay problem by introducing causal convolution.

4.2. Blind Test Case Results

A case study using the blind test case data from NASA was conducted to evaluate the influence of the noise filtering process on the overall diagnostic performance. A single flat MLP classifier is developed for fault classification that includes detection and isolation. The initial training data set described in Section 3.2 is used for training and validating the MLP classifier. The initial training data set and the blind test case data from NASA are processed with the proposed data processing method. Blind test case diagnostic assessments of the MLP classifiers with the proposed CNN-DAE method were submitted to NASA for evaluation. The evaluation results of the proposed CNN-DAE method are shown in Fig. 16 of Appendix A and explained in the following subsections.

4.2.1. Detection Performance Metrics

Table 4 presents the True Positive Rate (TPR), False Positive Rate (FPR), False Alarm Rate (FAR), and Detection Latency metrics concerning all fault types, fault evolution rates, and fault magnitudes.

Table 4 Detection performance.

Diagnostic method	FPR [%]	FAR	TPR [%]	Detection Latency
MLP with MLP1-DAE	0.08110	1232	36.4	3.42
MLP with MLP2-DAE	0.08601	1162	62.8	19.50
MLP with EMA	0.03935	2541	45.9	3.90
MLP with WRM	0.09600	1440	54.2	8.39
MLP with LSTM-DAE	0.06272	1594	48.6	3.37
MLP with WaveNet-DAE	0.05104	1959	46.1	3.88
MLP with CNN-DAE	0.07303	1369	65.7	2.69

FAR is the inverse of FPR and is presented in the third column of Table 4. It reflects the average number of flights required to generate a false alarm. A target FAR of 1000 flights or greater is specified in reference [8] to maintain uniformity in the diagnostic methods. All these methods satisfy the FAR target. With the same MLP classifier for fault diagnostics, the MLP with the CNN-DAE method provides the highest TPR of 65.7%, which is a 2.9% improvement over the method with the second-highest TPR score (the MLP with MLP2-DAE method). In most cases, some latency is associated with the correct detection of a fault resulting in missed detections within the first few flights after the fault occurs. As shown in the fifth column of Table 4, the Detection Latency of the proposed diagnostic method is 2.69, which exhibits superior diagnostic latency compared to the other methods. In conclusion, the proposed MLP with the CNN-DAE method performs best on fault detection among all the compared methods by detecting fault events more accurately with the shortest Detection Latency.

4.2.2. Kappa Coefficient

Table 5 presents the average Kappa Coefficient results concerning all fault types, fault evolution rates, and fault magnitudes. Kappa Coefficient reflects the overall fault classification performance. With the same MLP classifier for fault diagnostics, MLP with the CNN-DAE method produces an overall Kappa Coefficient of 0.731. The Kappa Coefficient for abrupt faults is 0.824, and the Kappa Coefficient for rapid faults is 0.704. MLP with CNN-DAE method produces the highest Kappa Coefficient.

Table 5 Kappa Coefficient.

Diagnostic method	Overall Kappa	Kappa for abrupt fault	Kappa for rapid fault
MLP with MLP1-DAE	0.460	0.573	0.420
MLP with MLP2-DAE	0.677	0.705	0.647
MLP with EMA	0.561	0.674	0.518
MLP with WRM	0.626	0.713	0.594
MLP with LSTM-DAE	0.481	0.482	0.457
MLP with WaveNet-DAE	0.548	0.665	0.502
MLP with CNN-DAE	0.731	0.824	0.704

4.2.3. Discussion

Table 6 summarises the blind test case study results, which provides the ranking of the diagnostic methods for each evaluation metric. With the same MLP classifier for fault diagnostics, the MLP with the CNN-DAE method proposed in this paper ranks first in TPR, Detection Latency, and Kappa Coefficient evaluation metrics. As a reflection of fault classification performance, the average Kappa Coefficient is enhanced to 0.731, and the Kappa Coefficient for abrupt faults is above 0.80. It clearly illustrates the importance of the noise filtering process on the gas path diagnostic performance. Detection Latency for the MLP with MLP2-DAE method and MLP with WRM method are 19.50 and 8.39, which are much higher than the other noise filtering methods. The reason for the high Detection Latency is time delay due to the use of future data points. MLP1-DAE provides a noise reduction rate of 42.24% and ranks fourth among the studied methods; however, the average Kappa Coefficient evaluation for MLP with MLP1-DAE method ranks seventh. The MLP with MLP1-DAE method has the worst diagnostic performance, which proves the worse performance of MLP1-DAE on preserving critical features in the signal compared to the other studied noise filtering methods.

Table 6 Diagnostic method ranking for each metric.

Diagnostic method	TPR	Detect latency	Overall Kappa
MLP with MLP1-DAE	7th	3rd	7th
MLP with MLP2-DAE	2nd	7th	2nd
MLP with EMA	6th	5th	4th
MLP with WRM	3rd	6th	3rd
MLP with LSTM-DAE	4th	2nd	6th
MLP with WaveNet-DAE	5th	4th	5th
MLP with CNN-DAE	1st	1st	1st

Table 7 summarises the blind-test-case metric results from other known diagnostic methods using the ProDiMES software for evaluation. Regularized Extreme Learning Machines-Sparse Representation Classification (RELM-SRC) method [33] has the best diagnostic performance, which provides a kappa coefficient of 0.685. The kappa coefficient of the proposed method in this paper is 0.731, which is 0.046, larger than the kappa coefficient provided by the RELM-SRC method.

Table 7 Blind test case results from other known diagnostic methods.

Diagnostic method	TPR [%]	Detect latency	Overall Kappa
Weighted Least Squares[8]	44.7	4.86	0.588
Probabilistic Neural Network (PNN)[8]	44.7	4.86	0.590
Extended Kalman Filter[8]	50.9	4.02	0.627
Generalized Observer[8]	51.9	4.24	0.617
PNNs[9]	48.5	4.70	0.595
k-Nearest Neighbour[9]	48.5	4.70	0.605
PNN and Adaptive Engine Model Fusion[9]	48.5	4.70	0.595
Support Vector Machine (SVM)[34]	52.5	3.30	0.660
Upper and Lower Singleton Type-2 Fuzzy Logic System (ULST2-FLS)[35]	52.2	4.45	0.647
Regularized Extreme Learning Machines- Sparse Representation Classification (RELM-SRC)[33]	55.7	3.30	0.685

5. CONCLUSIONS

In the context of Intelligent Engines, this study proposes a novel CNN-DAE method for aircraft engine gas path health signal denoising. The proposed method is evaluated with NASA's ProDiMES software.

The conclusions drawn from this study are as follows:

- The proposed denoising method can effectively remove the noise in health signals by reconstructing the denoised data from the noisy data. The proposed denoising method is superior in denoising performance to other optional denoising methods in the open literature.

- The proposed method can accommodate time-series data. The convolution operation can adequately capture changes in the signals that evolve with time. The causal convolution can solve the problem of using future data points.
- The MLP with CNN-DAE diagnostic method presents the best performance compared to other known diagnostic methods. Kappa Coefficient of the proposed diagnostic method is 0.731 and is at least 0.046 higher than the other diagnostic methods. It is proved that the proposed CNN-DAE denoising method can preserve the important changes in health signals for enhanced diagnostic information, which significantly improves diagnostic accuracy.

Overall, the proposed method can accommodate time series without using future data points, remove noise for improved denoising accuracy and preserve the important changes in health signals for enhanced diagnostic information. The proposed method can potentially contribute to intelligent condition monitoring systems by effectively exploiting historical information for improved denoising and diagnostic performance, which will enhance the availability, reliability, and efficiency of Intelligent Engines.

As an ML method, the proposed method has limitations in terms of the need for a large amount of labelled training data, case-dependency and offline training for the applications in aircraft engine diagnostics. Fortunately, emerging technologies such as Digital Twin, Intelligent Engine, and Incremental Online Learning provide potential opportunities for ML applications. As for future work, more studies to address these application limitations will be worth developing.

NOMENCLATURE

AANN	=	Auto-Associative Neural Network
BP	=	Back Propagation
C-MAPSS	=	Commercial Modular Aero-Propulsion System Simulation
CNN	=	Convolutional Neural Network
CNN-DAE	=	Convolutional Neural Network Denoising Autoencoder
CWIM	=	Center Weighted Idempotent Median
D	=	Dimension (Axis or Rank) for Tensor
DOD	=	Domestic Object Damage
EFS	=	Engine Fleet Simulator
EMA	=	Exponential Moving Average
FIR	=	Finite Impulse Response
FMH	=	FIR Median Hybrid
FOD	=	Foreign Object Damage
FP	=	Forward Propagation
FPR	=	False Positive Rate
HPC/LPC	=	High/Low Pressure Compressor
HPT/LPT	=	High/Low Pressure Turbine
IIR	=	Infinite Impulse Response
LSTM	=	Long Short-Term Memory Network
LSTM-DAE	=	Long Short-Term Memory Network Denoising Autoencoder
MA	=	Moving Average
MAE	=	Mean Absolute Error
MCR	=	Misclassification Rate
ML	=	Machine Learning
MLP	=	Multilayer Perceptron
NN	=	Neural Network
1DCNN	=	One-Dimensional Convolutional Neural Network
1D-FP	=	One-Dimensional Forward Propagation
PNN	=	Probabilistic Neural Network
ProDiMES	=	Propulsion Diagnostic Method Evaluation Strategy
RELM-SRC	=	Regularized Extreme Learning Machines-Sparse Representation Classification
ReLU	=	Rectified Linear Activation Function
RNN	=	Recurrent Neural Network
RM	=	Recursive Median
Seq2Seq	=	Sequence-to-Sequence Learning

SVM	=	Support Vector Machine
TPR	=	True Positive Rate
ULST2-FLS	=	Upper and Lower Singleton Type-2 Fuzzy Logic System
VBV	=	Variable Bleed Valve
VSV	=	Variable Stator Vane
WaveNet-DAE	=	WaveNet Denoising Autoencoder
WRM	=	Weighted Recursive Median

REFERENCES

[1] Lipowsky, H., Staudacher, S., Bauer, M., and Schmidt, K. J., 2010, "Application of Bayesian Forecasting to Change Detection and Prognosis of Gas Turbine Performance," J. Eng. Gas Turbines Power, **132**(3), pp. 1–8.

[2] Sogut, M. Z., Yalcin, E., and Karakoc, T. H., 2017, "Assessment of Degradation Effects for an Aircraft Engine Considering Exergy Analysis," Energy, **140**, pp. 1417–1426.

[3] Volponi, A. J., and Tang, L., 2016, "Improved Engine Health Monitoring Using Full Flight Data and Companion Engine Information," SAE Int. J. Aerosp., **9**(1), pp. 91–102.

[4] Simon, D. L., 2010, *NASA/TM—2010-215840: Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) User's Guide*, NASA.

[5] Ganguli, R., 2012, *Gas Turbine Diagnostics: Signal Processing and Fault Isolation*, London: CRC Press.

[6] Chen, Y. Z., Zhao, X. D., Xiang, H. C., and Tsoutsanis, E., 2021, "A Sequential Model-Based Approach for Gas Turbine Performance Diagnostics," Energy, **220**(11), pp. 1–20.

[7] Kim, S., 2021, "A New Performance Adaptation Method for Aero Gas Turbine Engines Based on Large Amounts of Measured Data," Energy, **221**(11), pp. 1–15.

[8] Simon, D. L., Borguet, S., Léonard, O., and Zhang, X., 2014, "Aircraft Engine Gas Path Diagnostic Methods: Public Benchmarking Results," J. Eng. Gas Turbines Power, **136**(4), pp. 1–10.

[9] Koskoletos, A. O., Aretakis, N., Alexiou, A., Romesis, C., and Mathioudakis, K., 2018, "Evaluation of Aircraft Engine Gas Path Diagnostic Methods through ProDiMES," J. Eng. Gas Turbines Power, **140**(12), pp. 1–13.

[10] Uday, P., and Ganguli, R., 2010, "Jet Engine Health Signal Denoising Using Optimally Weighted Recursive Median Filters," J. Eng. Gas Turbines Power, **132**(4), pp. 1–8.

[11] Ganguli, R., 2002, "Noise and Outlier Removal from Jet Engine Health Signals Using Weighted FIR Median Hybrid Filters," Mech. Syst. Signal Process., **16**(6), pp. 967–978.

[12] Ganguli, R., 2003, "Jet Engine Gas-Path Measurement Filtering Using Center Weighted Idempotent Median Filters," J. Propuls. Power, **19**(5), pp. 930–937.

[13] Raikar, C., and Ganguli, R., 2017, "Denoising Signals Used in Gas Turbine Diagnostics with Ant Colony Optimized Weighted Recursive Median Filters," Ina. Lett., **2**(3), pp. 133–143.

[14] Borguet, S., Leonard, O., and Dewallef, P., 2016, "Regression-Based Modeling of a Fleet of Gas Turbine Engines for Performance Trending," J. Eng. Gas Turbines Power, **138**(2), pp. 1–9.

[15] Bengio, Y., 2009, *Technical Report 1312: Learning Deep Architectures for AI*, Foundations and Trends in Machine Learning.

[16] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J., 1985, "A Learning Algorithm for Boltzmann Machines," Cogn. Sci., **9**(1), pp. 147–169.

[17] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. A., 2008, "Extracting and Composing Robust Features with Denoising Autoencoders," *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008, pp. 1096–1103.

[18] Kramer, M. A., 1992, "Autoassociative Neural Networks," Comput. Chem. Eng., **16**(4), pp. 313–328.

[19] Guo, T. H., Saus, J., Lin, C. F., and Ge, J. H., 1996, "Sensor Validation for Turbofan Engines Using an Autoassociative Neural Network," *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Diego, CA, 29-31 July 1996, pp. 1–8.

[20] Lu, P. J., Zhang, M. C., Hsu, T. C., and Zhang, J., 2001, "An Evaluation of Engine Faults Diagnostics Using Artificial Neural Networks," J. Eng. Gas Turbines Power, **123**(2), pp. 340–346.

[21] Zedda, M., and Singh, R., 1998, "Fault Diagnosis of a Turbofan Engine Using Neural Networks: A Quantitative Approach," *Proceedings of the 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Cleveland, OH, USA, 13-15 July 1998.

[22] Ogaji, S. O. T., Singh, R., and Probert, S. D., 2002, "Multiple-Sensor Fault-Diagnoses for a 2-Shaft Stationary Gas-Turbine," Appl. Energy, **71**(4), pp. 321–339.

[23] Li, D., Zhou, J., and Liu, Y., 2021, "Recurrent-Neural-Network-Based Unscented Kalman Filter for Estimating and Compensating the Random Drift of MEMS Gyroscopes in Real Time," Mech. Syst. Signal Process., **147**, pp. 1–20.

[24] Li, H., Yang, Z., and Yan, W., 2022, "An Improved AIC Onset-Time Picking Method Based on Regression Convolutional Neural Network," Mech. Syst. Signal Process., **171**(January), pp. 1–20.

[25] Bai, S., Kolter, J. Z., and Koltun, V., 2018, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," arXiv, pp. 1–14.

[26] Barzegar, V., Laflamme, S., Hu, C., and Dodson, J., 2021, "Ensemble of Recurrent Neural Networks with Long Short-Term Memory Cells for High-Rate Structural Health Monitoring," Mech. Syst. Signal Process., **164**(September), pp. 1–15.

[27] Chen, Z., Xia, T., Li, Y., and Pan, E., 2021, "A Hybrid Prognostic Method Based on Gated Recurrent Unit Network and an Adaptive Wiener Process Model Considering Measurement Errors," Mech. Syst. Signal Process., **158**, pp. 1–21.

[28] Zhao, J., and Li, Y., 2020, "Abrupt Fault Detection and Isolation for Gas Turbine Components Based on a 1d Convolutional Neural Network Using Time Series Data," *Proceedings of AIAA Propulsion and Energy 2020 Forum*, Virtual, Online, 24-28 August 2020, pp. 1–19.

[29] Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., and Inman, D. J., 2021, "1D Convolutional Neural Networks and Applications: A Survey," Mech. Syst. Signal Process., **151**, pp. 1–21.

[30] Goodfellow, I., Bengio, Y., and Courville, A., 2016, *Deep Learning*, MIT Press.

[31] Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K., 2016, "WaveNet: A Generative Model for Raw Audio," arxiv, pp. 1–15.

[32] Feurer, M., and Hutter, F., 2019, "Chapter 8: Hyperparameter Optimization," *Automated Machine Learning - Methods, Systems, Challenges*, Springer, pp. 233–317.

[33] Pérez-Ruiz, J. L., Tang, Y., and Loboda, I., 2021, "Aircraft Engine Gas-Path Monitoring and Diagnostics Framework Based on a Hybrid Fault Recognition Approach," Aerospace, **8**(8), pp. 1–26.

[34] Loboda, I., Pérez-Ruiz, J. L., and Yepifanov, S., 2018, "A Benchmarking Analysis of a Data-Driven Gas Turbine Diagnostic Approach," *Proceedings of ASME Turbo Expo 2018 Turbomachinery Technical Conference and Exposition*, Oslo, Norway, 11-15 June 2018, pp. 1–13.

[35] Calderano, P. H. S., Ribeiro, M. G. C., Amaral, R. P. F., Vellasco, M. M. B. R., Tanscheit, R., and de Aguiar, E. P., 2019, "An Enhanced Aircraft Engine Gas Path Diagnostic Method Based on Upper and Lower Singleton Type-2 Fuzzy Logic System," J. Brazilian Soc. Mech. Sci. Eng., **41**(2), pp. 1–14.

APPENDIX

Appendix A. The Blind Test Evaluation Results from NASA

Abrupt Fault Cases (all)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detection Latency	Classify Latency
Fan	0.82	0	0	0	0	0	0	9E-04	0	0	0	0	0	0	0	0	0	2E-02	0.16	82%	1.2	1.0
LPC	0	0.63	0	0	0	0	2E-02	0	0	4E-03	0	0	0	0	0	0	0	0	0.34	63%	1.9	2.0
HPC	0	0	0.91	0	0	0	0	0	4E-03	0	0	0	0	0	0	0	0	0	0.09	91%	0.6	0.7
HPT	0	0	0	0.99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01	99%	0.1	0.1
LPT	0	9E-04	0	0	0.96	0	9E-04	0	0	0	0	0	0	0	0	0	2E-03	2E-02	0.02	96%	0.2	0.4
VSV	0	0	0	0	0	0.96	0	0	1E-02	0	0	0	0	0	0	0	3E-03	0	0.02	96%	0.2	0.3
VBV	0	7E-02	0	0	0	0	0.64	0	0	3E-03	0	1E-02	0	0	0	0	2E-03	0	0.37	54%	2.0	2.3
Nf	4E-02	0	0	0	0	0	0	0.81	0	0	0	0	0	0	0	0	0	2E-02	0.13	81%	0.6	0.4
Nc	0	0	5E-04	0	0	1E-02	0	0	0.27	0	0	0	1E-02	0	0	0	1E-03	0	0.70	27%	4.1	4.3
P24	0	3E-02	0	0	0	0	3E-03	0	0	0.82	0	0	0	0	0	3E-02	0	0	0.12	82%	0.9	1.0
Ps30	2E-03	0	6E-03	0	0	0	2E-03	3E-03	0	0	0.61	4E-03	0	0	0	0	0	2E-03	0.37	61%	2.1	2.3
T24	0	0	0	0	0	0	7E-03	0	0	0	8E-04	0.86	0	0	0	0	3E-02	0	0.10	86%	0.6	0.8
T30	0	0	0	0	0	0	0	0	1E-02	0	0	0	0.70	6E-04	0	0	0	0	0.29	70%	1.1	1.1
T48	0	0	0	0	0	0	0	0	0	3E-03	0	0	0	0.88	9E-04	0	0	0	0.12	88%	0.8	0.9
WF36	5E-03	0	0	0	0	0	0	8E-03	0	0	0	0	0	4E-03	0.83	7E-03	0	0	0.14	83%	0.9	0.9
P2	0	1E-02	0	0	0	1E-03	3E-03	3E-03	0	2E-02	0	0	0	5E-03	0	0.62	0	6E-04	0.44	52%	2.0	2.0
T2	0	0	0	0	0	5E-03	3E-03	0	0	0	0	8E-03	0	0	0	0	0.82	4E-03	0.16	82%	0.8	0.7
Pamb	1E-02	0	0	0	5E-03	0	0	9E-04	0	0	0	0	0	0	0	0	6E-03	0.08	0.90	8%	4.7	5.0
No Fault	4E-05	1E-04	7E-05	1E-05	0	4E-06	8E-05	5E-05	1E-05	3E-05	4E-05	3E-05	2E-05	1E-05	2E-05	6E-05	3E-05	1E-04	0.99927	99.927%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.741	0.259	1.2
No Fault	7.3E-04	0.99927	N/A

Kappa Coefficient
0.82

False Alarm Rate		
Once per	1369	flights

(a) Abrupt faults

Rapid Fault Cases (all)

Confusion Matrix

Predicted State

True State	Fan	LPC	HPC	HPT	LPT	VSV	VBV	Nf	Nc	P24	Ps30	T24	T30	T48	WF36	P2	T2	Pamb	No Fault	Accuracy	Detection Latency	Classify Latency
Fan	0.68	0	0	0	0	0	0	0	0	0	5E-04	0	0	0	0	2E-03	0	2E-02	0.32	68%	4.4	4.6
LPC	0	0.44	0	0	0	0	1E-02	0	0	0	0	0	0	0	0	3E-03	0	0	0.54	44%	6.3	6.3
HPC	0	0	0.73	0	0	0	0	0	3E-03	0	4E-04	0	0	0	0	0	0	0	0.27	73%	3.8	3.8
HPT	0	0	0	0.84	0	0	0	0	0	0	2E-03	4E-03	0	0	0	0	0	0	0.15	84%	2.3	2.4
LPT	0	6E-04	0	0	0.78	0	0	0	0	0	3E-03	0	0	0	0	6E-04	2E-03	3E-02	0.19	78%	2.8	3.3
VSV	0	0	0	0	0	0.79	0	0	3E-02	0	0	0	0	0	0	0	4E-03	0	0.18	79%	2.7	3.2
VBV	0	3E-02	0	0	0	0	0.45	0	0	0	0	5E-03	0	0	0	0	4E-03	0	0.51	45%	5.6	5.7
Nf	3E-02	0	0	0	0	0	1E-03	0.65	0	0	2E-03	0	0	0	0	5E-04	0	1E-02	0.31	65%	3.9	3.9
Nc	0	0	0	0	0	6E-03	0	0	0.15	0	0	0	4E-03	0	0	0	0	0	0.84	15%	9.4	9.4
P24	0	1E-02	0	0	0	0	8E-03	0	0	0.68	0	0	5E-04	7E-03	5E-04	1E-02	5E-04	0	0.28	68%	3.5	3.9
Ps30	0	0	3E-03	0	2E-03	0	5E-04	0	0	0	0.41	2E-03	0	0	0	0	0	5E-04	0.58	41%	6.3	6.3
T24	0	5E-04	0	1E-03	0	0	1E-02	0	0	0	5E-03	0.64	0	0	0	0	4E-02	0	0.31	64%	4.0	4.0
T30	0	0	0	2E-03	0	0	0	0	1E-02	0	0	0	0.56	4E-03	0	0	0	0	0.42	56%	4.7	4.8
T48	0	0	0	0	0	0	0	0	2E-03	5E-03	0	0	0	0.68	2E-03	0	0	0	0.31	68%	4.0	4.1
WF36	3E-03	0	0	0	0	0	0	6E-03	0	0	0	0	0	4E-04	0.64	2E-03	0	0	0.35	64%	4.5	4.5
P2	0	3E-03	3E-03	0	0	0	0	9E-03	0	1E-02	0	0	0	4E-04	1E-03	0.35	0	8E-04	0.62	35%	5.6	7.1
T2	0	0	5E-04	0	0	8E-03	1E-03	0	0	5E-04	0	2E-02	0	0	0	0	0.65	1E-03	0.32	65%	4.0	4.2
Pamb	2E-02	0	0	0	2E-03	0	0	3E-03	0	0	3E-03	0	0	0	0	0	2E-03	0.07	0.90	7%	9.2	8.8
No Fault	4E-05	1E-04	7E-05	1E-05	0	4E-06	8E-06	5E-05	1E-05	3E-05	4E-05	3E-05	2E-05	1E-05	2E-05	6E-05	3E-05	1E-04	0.99927	99.927%	N/A	N/A

Decision Matrix

Predicted State

True State	Fault	No Fault	Detection Latency
Fault	0.582	0.418	4.5
No Fault	7.3E-04	0.99927	N/A

Kappa Coefficient
0.70

(b) Rapid faults

Fig. 16. Evaluation metrics for the MLP classifier with the proposed CNN-DAE method.

2022-10-31

Convolutional neural network denoising autoencoders for intelligent aircraft engine gas path health signal noise filtering

Zhao, Junjie

American Society of Mechanical Engineers

Zhao J, Li Y-G, Sampath S (2023) Convolutional neural network denoising autoencoders for intelligent aircraft engine gas path health signal noise filtering. Journal of Engineering for Gas Turbines and Power, Volume 145, Issue 6, June 2023, Article number 061013, Paper number GTP-22-1234 https://doi.org/10.1115/1.4056128