AIAA Propulsion and Energy 2020 Forum, August 24-28, 2020, VIRTUAL EVENT
DOI:10.2514/6.2020-3675

# Abrupt Fault Detection and Isolation for Gas Turbine Components Based on a 1D Convolutional Neural Network using Time Series Data

Junjie Zhao[1] and Yiguang Li[2]
*School of Aerospace, Transport and Manufacturing, Cranfield University, Bedfordshire, MK43 0AL, UK*

**The FDI step identifies the presence of a fault, its level, type, and possible location. Gas turbine gas-path fault detection and isolation can improve the availability and economy of gas turbine components. Data-driven FDI methods are studied in this paper. Some notable gas turbine FDI challenges include: insensitivity to operating conditions, robust separation of faults, noisy sensor readings and missing data, reliable fault detection in time-varying conditions, and the influence of performance gradual deterioration. For conventional ML methods, the problem with handling time series data is its volume and the associated computational complexity; therefore, the available information must be appropriately compressed via the transformation of high-dimensional data into a low-dimensional feature space with minimal loss of class separability. In order to improve the detection and isolation sensitivity, this paper develops a method for FDI based on CNNs. Work in this paper includes: (1) Defining the problem and assembling a dataset. (2) Preparing data for training, validation and test: data generation, feature engineering, data pre-processing, data formatting. (3) Building up the model. (4) Training and validating the model (evaluation protocol). (5) Optimizing: a. deciding the model size. b. regularizing the model by getting more training data, reducing the capacity of the network, adding weight regularization or adding dropout. c. tuning hyperparameters. (6) Evaluation.**

## I. Nomenclature

| | | |
|---|---|---|
| *ANN* | = | artificial neural network |
| *CNN* | = | convolutional neural network |
| *CS* | = | Computer Science |
| *DBN* | = | deep belief network |
| *D* | = | dimension for tensor (axis or rank), is different for vector. A vector is a 1D tensor |
| *DOD* | = | domestic object damage |
| *GPU* | = | graphics processing unit |
| *GRN* | = | gated relation network |
| *FDI* | = | fault detection and isolation |
| *FOD* | = | foreign object damage |
| *LSTM* | = | long short-term memory |
| *ML* | = | Machine Learning |
| *MLP* | = | multilayer perceptron |
| *1DCNN* | = | one dimensional convolutional neural network |
| *PNN* | = | probabilistic neural network |

[1] Ph.D. Student, Centre for Propulsion Engineering, junjie.zhao@cranfield.ac.uk.
[2] Reader, Centre for Propulsion Engineering, i.y.li@cranfield.ac.uk.

*RBN* = radial basis neural network
*RNN* = recurrent neural network
*SOM* = self-organizing maps
*SVM* = support vector machine

## II. Introduction

Gas turbine health monitoring is generally divided into three steps, namely, detection and isolation (FDI), diagnosis, and prognosis. The performance of gas turbines degrades over time because of deterioration mechanisms and fault events [1]. Gradual performance deterioration is not considered a fault, and it evolves on a much slower timescale than faults do. Fault diagnostic methods will not be required to diagnose gradual performance deterioration, but they should be designed to be robust with respect to these effects [2]. The FDI step identifies the presence of a fault, its level, type, and possible location [3] of abrupt events and failures such as FOD, DOD, bleed leaks & failures, variable geometry anomalies, actuator & instrumentation faults, and the like. If no abrupt change has been detected, long term engine module performance degradation because of fouling, erosion, and corrosion of turbomachinery blades and vanes [2] must be taken into account. Thermodynamic changes in efficiency and flow parameter for each major module (compressors and turbines) will provide a measure for the performance degradation in order to estimate when predefined limits of particular parameters are exceeded, which is referred to as prognosis [4].

In general, gas turbine gas path related faults are classified into three major categories, i.e. components faults (e.g. fan, compressor, and turbines), sensor faults (e.g. random, malfunction, drift, bias) [5, 6] and actuator faults (e.g. sluggish response and excessive dead band and hysteresis) [7]. Faults that appear instantaneously, but do not grow in magnitude over time, are considered abrupt faults. Faults that initiate and grow in magnitude over time are considered rapid faults [2].

The advanced gas turbine health monitoring method is classified into model-based method and data-driven method. As we all know, the complex and noisy working conditions hinder the construction of physical models, which make the modelling of complex dynamic systems very difficult. Most of these physics-based models cannot be updated with data measured on-line, which limits their effectiveness and flexibility. On the other hand, with significant development of sensors, sensor networks and computing systems, data-driven machine health monitoring models have become more and more attractive.

To extract useful knowledge and make appropriate decisions from Big Data, ML techniques have been regarded as a powerful solution. ML is a “data-driven” approach, that is, ML algorithms try to construct a “set of rules” or “blackbox” model of the “system” under analysis from data, and then use the model to predict the behavior of the system [8].

ML algorithms, for example, ANNs [9], SVM, KNN, BNN, Ensemble methods (Random Forest, etc.), and ELM are commonly-used diagnostics methods. Some comparisons haven been conducted for these basic ML algorithms in [8, 10, 11, 12]. ANNs, such as MLPs, RBN, PNN, AANN, SOM, and DNN are widely used in gas turbine diagnostics for performance simulation, fault detection, fault isolation, and fault identification.

In this study, a deep learning method is studied. As the most popular subfield of machine learning, deep learning is able to act as a bridge connecting big machinery data and intelligent machine health monitoring [13]. Originating from artificial neural networks, deep learning is a branch of machine learning which features multiple non-linear processing layers and tries to learn hierarchical representations of data. Deep learning methods are widely developed in machinery health monitoring, some of reviews are [13, 14]. Deep learning models have several variants such as Auto-encoders [15], DBN [16], Deep Boltzmann Machines, CNN [17], and RNN [18].

One method to enhance data-driven diagnostics is to change data types. Data-driven techniques for health monitoring of gas turbine engines either use snapshot data at a time instant from selected sensor observations or a window of time series data from various sensors [3]. The data volume is less when snapshot type data are handled and as a consequence, the computational expense is low also. However, by using only snapshot data, statistical changes in the information acquired may not be adequately captured, eventually resulting in missed detection of faults; this problem can be alleviated by using a window of time series data. For conventional ML methods, the problem with handling time series data is its volume and the associated computational complexity; therefore, the available information must be appropriately compressed via transformation of high-dimensional data into a low-dimensional feature space with minimal loss of class separability [3].

CNNs are successfully used in pattern recognition and image processing [19]. A CNN is good at learning local features, and it is robust with respect to the feature shift, scale, and distortion. 1DCNN, the one-dimensional version of the 2D convnets will be developed to deal with time-series data, which can be thought of as a 1D grid taking samples at regular time intervals. The property of a 1DCNN makes it a viable solution for dealing with the FDI problem. Firstly,

the information about a component fault is contained in several related sensors. The abrupt information of an abrupt fault is contained in several adjacent sensor values rather than all the values in a sliced time-series. By perceiving local features and sharing weights, a CNN can accurately extract key information from adjacent sensor values. Secondly, when processing time series data, a CNN produces a sort of timeline that shows when different features appear in the input. If we move an event later in time in the input, the exact same representation of it will appear in the output, just later in time [19]. As a result, it can effectively discern the impact of noise and ensure some degree of shift invariance. In this paper, a 1DCNN is developed for gas turbine FDI. Furthermore, the reason why 1DCNN is suitable for component fault detection is analyzed and visualized in detail. 1DCNN is evaluated in the context of a model-based application of a gas turbine component FDI.

## III. 1DCNN Methodology

### A. 1DCNN Architecture

A CNN consists of an input and an output layer, as well as multiple hidden layers, as is shown in Fig. 1. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is followed by additional layers such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The final convolution, in turn, often involves backpropagation in order to more accurately weight the end product.

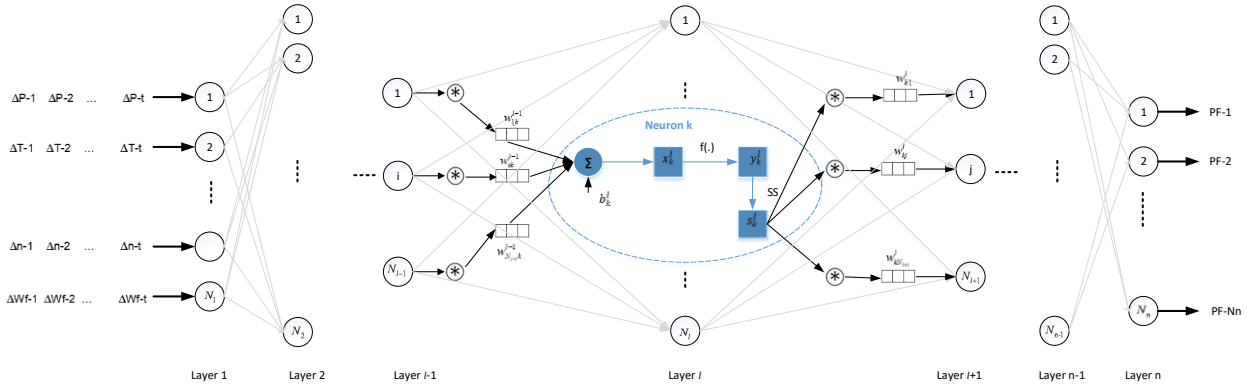


**Fig. 1 1DCNN architecture.**

An example of convolution (without kernel flipping) applied to a 1-D tensor is shown in Fig. 2.

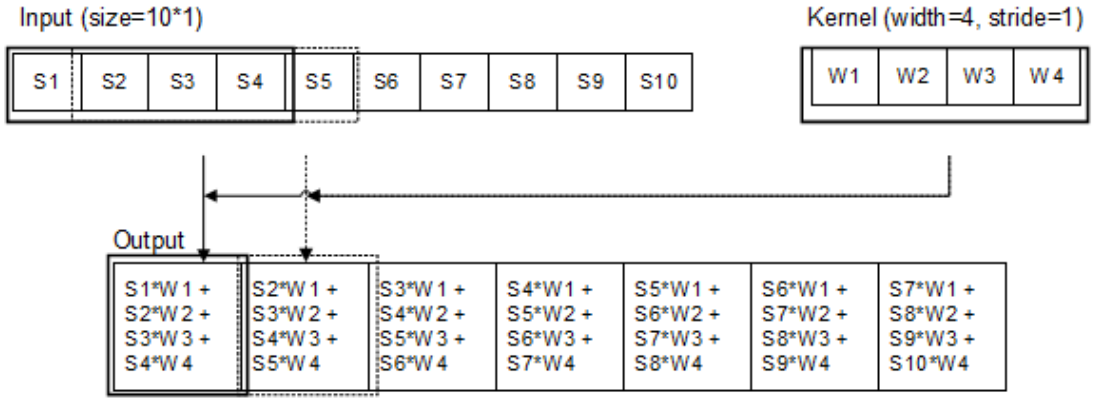


**Fig. 2 An example of convolution operation.**

An example of max pooling is shown in Fig. 3.

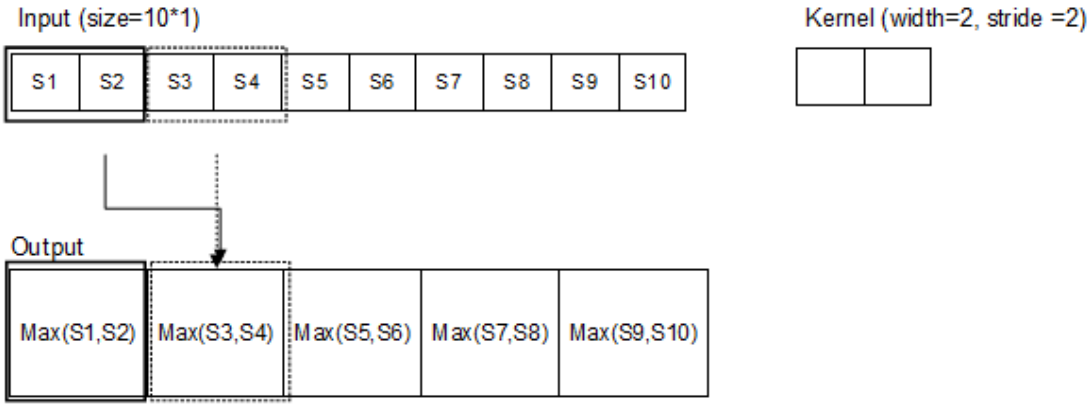


**Fig. 3 An example of max pooling.**

Each linear activation is run through a nonlinear activation function, such as the rectified linear activation function. Three activation functions, Sigmoid, tanh, and ReLU, are show in Fig. 4.

$$Sigmod(x) = \frac{1}{1+e^{-x}} \qquad \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad f(x) = \max(0, x)$$

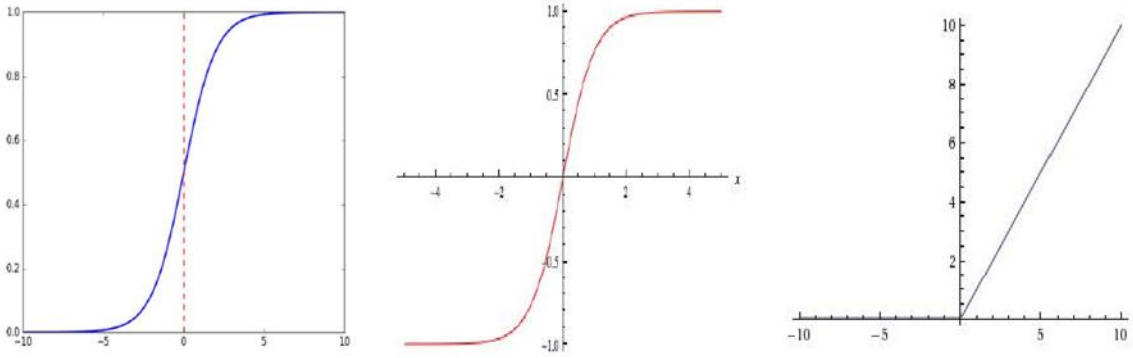

**Fig. 4 Three common activation functions.**

## B. Training of Convolutional Neural Networks

For the 1DCNN in this research, a typical hidden layer contains one or few functions from the convolution, pooling and activation functions. Other functions, such as fully connected and normalization functions, can also be included in a hidden layer. A typical hidden layer is shown in Fig. 5.

The final output of the kth neuron at layer l is $s_k^l$, therefore, the subsampled version of the intermediate output is $y_k^l$. In each CNN-layer, 1D forward propagation (1D-FP) is expressed as follows:

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} conv1D(w_{ik}^{l-1}, s_i^{l-1}) \tag{1}$$

Where conv1D (.,.) is a regular 1-D convolution without zero padding on the boundaries, $x_k^l$ is the input, $b_k^l$ is the bias of the kth neuron at layer l, and $s_i^{l-1}$ is the output of the ith neuron at layer l-1. $w_{ik}^{l-1}$ is the kernel (weight) from the ith neuron at layer l–1 to the kth neuron at layer l. To accomplish BP training, there are three more elements that are stored for each neuron: the delta error $\Delta_k^l$, downsampled delta error $\Delta S_k^l$, and, finally, the derivative of the intermediate output $f'(x_k^l)$.

Let l = 1 and l = NL be the input and output layers, respectively. For an input vector p, and its corresponding output vector, let [t1,…, tNL] be the target class vector. The MSE in the output layer can then be expressed as

$$\mathrm{E} = \mathrm{E}\left(y_1^L, \ldots, y_{N_L}^L\right) = \frac{1}{N_L}\sum_{i=1}^{N_L}(y_i^L - t_i)^2 \tag{2}$$

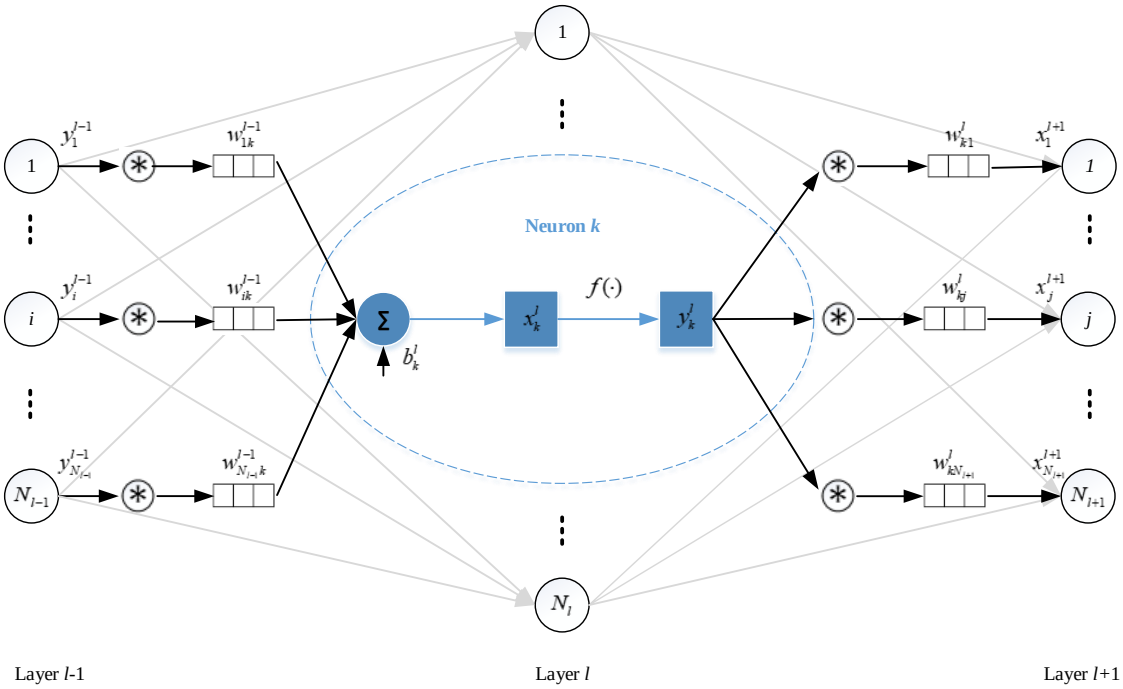


**Fig. 5 A typical hidden layer consists of convolution, pooling and activation function.**

Consequently, the iterative flow of the BP for the 1D signals in the training set can be stated as follows:

1) Initialize weights and biases (e.g., randomly, ~U (-0.1, 0.1)) of the network.
2) 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 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: Update the weights and biases by the (accumulation of) sensitivities scaled with the learning factor ε.

## IV. Data Generation

### A. Engine Model

For this project, an engine model which has similar thermodynamic properties of LM2500+ was chosen, as shown in Fig. 6 and Table 1.

First, the engine is modelled and simulated by the Cranfield University gas turbine performance simulation and diagnostics software, PYTHIA. Then, design point and off-design point adaptation are conducted to obtain accurate engine model.

**Table 1 Specification of the engine model comparing to LM2500+.**

| | LM2500+ | Model in this study |
|---|---|---|
| Power Output | 31.9 MW | 30 MW |
| Net Efficiency | 38.8 % | 39.8 % |
| Exhaust Temperature | 525 °C | 569 °C |
| Exhaust Gas Flow | 85.9 kg/s | 81.96 kg/s |
| Pressure Ratio | 23.1 | 21 |

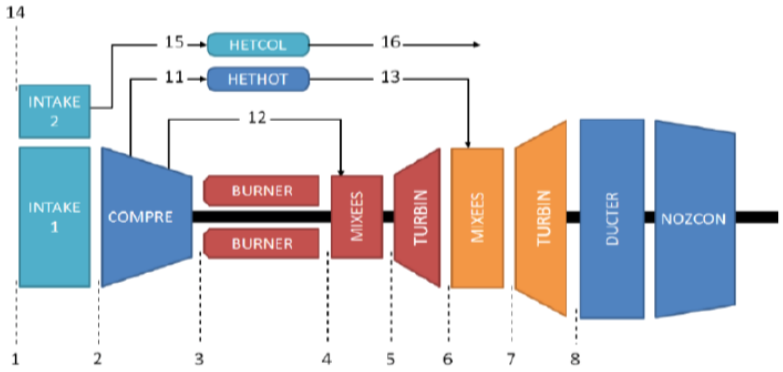


**Fig. 6 The engine model configuration.**

## B. Measurements Selection

The measurements selection method developed in Ref. [20] is used.

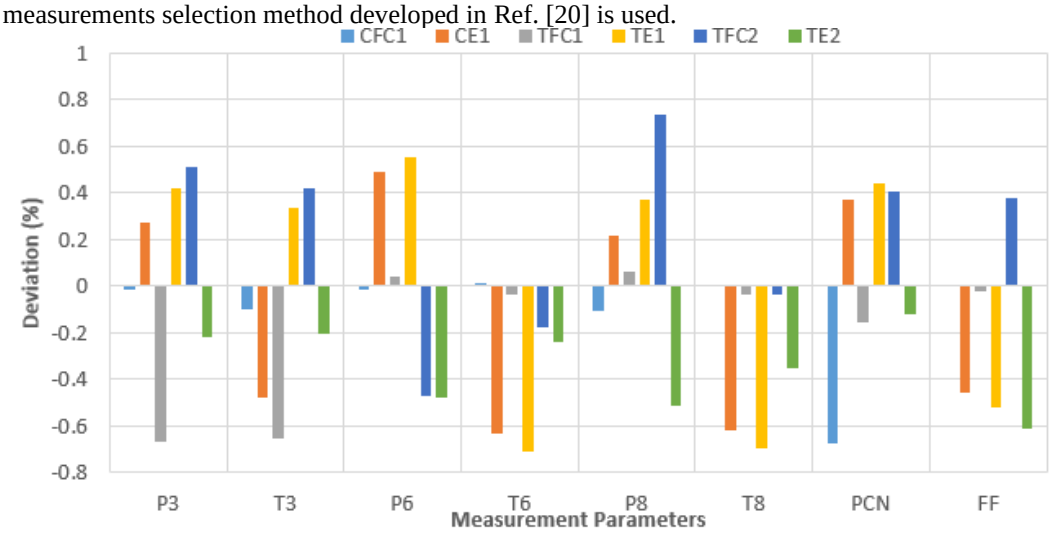


**Fig. 7 Sensitivity Analysis Bar Chart.**

**Table 2 Correlations in the measurements.**

| | P3 | T3 | P6 | T6 | P8 | T8 | PCN | FF |
|---|---|---|---|---|---|---|---|---|
| P3 | 1 | 0.7079 | 0.2035 | -0.4888 | 0.6594 | -0.3798 | 0.639 | -0.0043 |
| T3 | 0.7079 | 1 | -0.17 | 0.057 | 0.407 | 0.1417 | 0.3395 | 0.3413 |
| P6 | 0.2035 | -0.17 | 1 | -0.5072 | 0.2115 | -0.5035 | 0.2961 | -0.396 |
| T6 | -0.4888 | 0.057 | -0.5072 | 1 | -0.4134 | 0.9829 | -0.5943 | 0.7417 |
| P8 | 0.6594 | 0.407 | 0.2115 | -0.4134 | 1 | -0.2407 | 0.6704 | 0.296 |
| T8 | -0.3798 | 0.1417 | -0.5035 | 0.9829 | -0.2407 | 1 | -0.509 | 0.8525 |
| PCN | 0.639 | 0.3395 | 0.2961 | -0.5943 | 0.6704 | -0.509 | 1 | -0.1706 |
| FF | -0.0043 | 0.3413 | -0.396 | 0.7417 | 0.296 | 0.8525 | -0.1706 | 1 |

The sensitivity analysis revealed the most sensitive sensors to a given health parameter fault as shown in Fig. 7 Sensitivity Analysis Bar Chart. In the correlation matrix in Table 2, a relatively high element value denotes a correlation between the two corresponding sensors.

In conclusion, the measurement set selected for diagnostics is shown in Table 3.

**Table 3 Measurements set for diagnostics.**

| Ambient and Operating Condition Parameters | Measurement Setting |
|---|---|
| Ambient temperature | Compressor outlet total pressure(P3) |
| | Compressor outlet total temperature(T3) |
| Ambient pressure | Turbine1 outlet total pressure(P6) |
| | Turbine1 outlet total pressure(T6) |
| Relative Humidity | Turbine2 outlet total pressure(P8) |
| | Turbine2 outlet total temperature(T8) |
| Shaft power | Compressor relative rotational speed(PCN) |
| | Fuel flow rate(FF) |

### C. Patterns Generation with Engine Model in PYTIHA

The modules are developed to produce simulated "snap-shot" engine measurements, with relevant noise levels, as if collected from a gas turbine engine at a constant frequency. Engine operating conditions, component deterioration profiles and gas path faults should be specified before generating data with the engine model.

Gas turbine operating conditions cannot be constant most of the time because of variations in ambient conditions. In order to accommodate ambient condition variation, ambient temperature changes between -45 deg. Celsius to +45 deg. Celsius with up to 3% change in ambient pressure were considered.

**Table 4 Sensors and accuracy ranges [21].**

| Sensor | Unit | Required accuracy (%) |
|---|---|---|
| P3 | bar | ± 0.1 |
| T3 | K | ± 0.4 |
| P6 | bar | ± 0.1 |
| T6 | K | ± 0.4 |
| P8 | bar | ± 0.1 |
| T8 | K | ± 0.4 |
| PCN | r/min | ± 0.03 |
| FF | kg/s | ± 1.0 |

A gas turbine engine will naturally degrade over its lifetime of use because of fouling, erosion, and corrosion of turbomachinery blades and vanes. FDI methods will not be required to diagnose gradual performance deterioration, but they should be designed to be robust with respect to these effects [2]. Gradual performance deterioration evolves on a much slower timescale than faults do. In this research, faults are assumed to occur under healthy conditions (without performance deterioration).

**Table 5 Selected Fault Patterns.**

| Fault description | Fault magnitude (absolute value) | | | | | | Noise multiplier | No. Patterns |
|---|---|---|---|---|---|---|---|---|
| | Component 1 deviations | | Component 2 deviations | | Component 3 deviations | | | |
| | Efficiency (%) | Flow capacity (%) | Efficiency (%) | Flow capacity (%) | Efficiency (%) | Flow capacity (%) | | |
| No fault | / | / | / | / | / | / | 1 | 620 |
| C fault | 0 to 3.5 | 0 to 7 | | | | | 1 | 620 |
| CT fault | | | 0 to 3.5 | 0 to 7 | | | 1 | 620 |
| PT fault | | | | | 0 to 3.5 | 0 to 7 | 1 | 620 |
| C+CT fault | 0 to 3.5 | 0 to 7 | 0 to 3.5 | 0 to 7 | | | 1 | 720 |
| C+PT fault | 0 to 3.5 | 0 to 7 | | | 0 to 3.5 | 0 to 7 | 1 | 720 |
| CT+PT fault | | | 0 to 3.5 | 0 to 7 | 0 to 3.5 | 0 to 7 | 1 | 720 |
| C+CT+PT fault | 0 to 3.5 | 0 to 7 | 0 to 3.5 | 0 to 7 | 0 to 3.5 | 0 to 7 | 1 | 640 |

In this research, single and multiple abrupt component faults are studied. The fault type, the fault magnitude, and sensor noise level should be specified in order to generate a unique fault pattern, as shown in Table 5. The method in [21] is used to generate the fault patterns. The isentropic efficiency range was divided into several severity segments and each level was combined with different ratios of flow capacity drops. The gas path parameters and their corresponding noise values are given in Table 4. After being combined with white noise of Gaussian distribution, 5280 fault patterns (620 NF + 4660 component faults) were generated, as shown in Fig. 8.

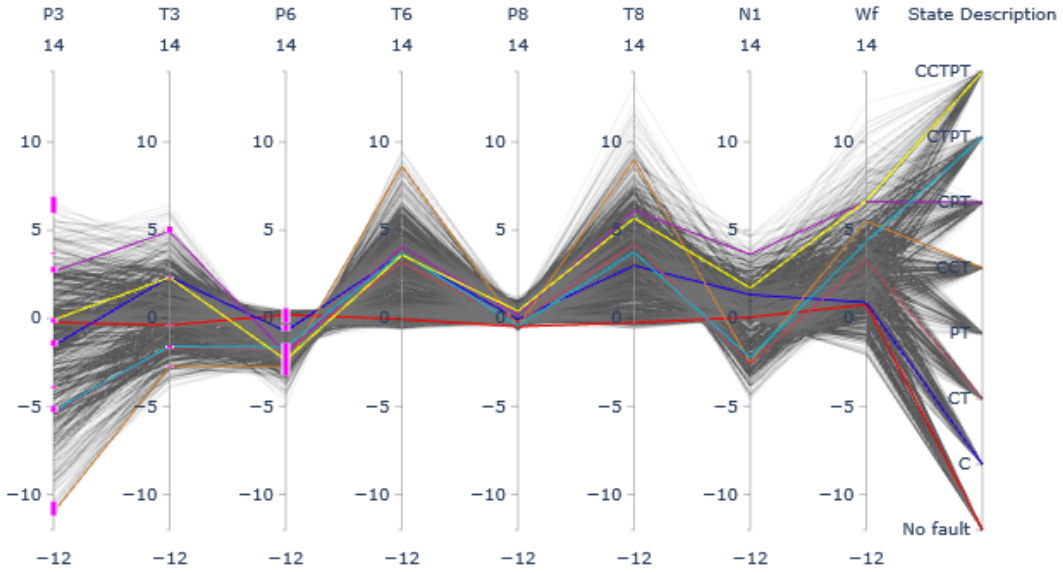


**Fig. 8 Fault patterns.**

In order to generate windows of time series data, the sampling rate, the length of window, the time of fault initiation and the fault evolution rate should be defined after obtaining the fault patterns. A fault sample is shown in Fig. 9. The time series data from selected sensor observations are shown in Fig. 10. As a result, 5280 time series samples with the abrupt fault occurring in the middle will be obtained.

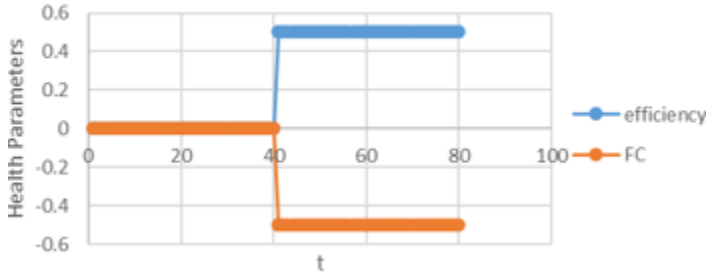


**Fig. 9 An example of Compressor Turbine abrupt fault.**

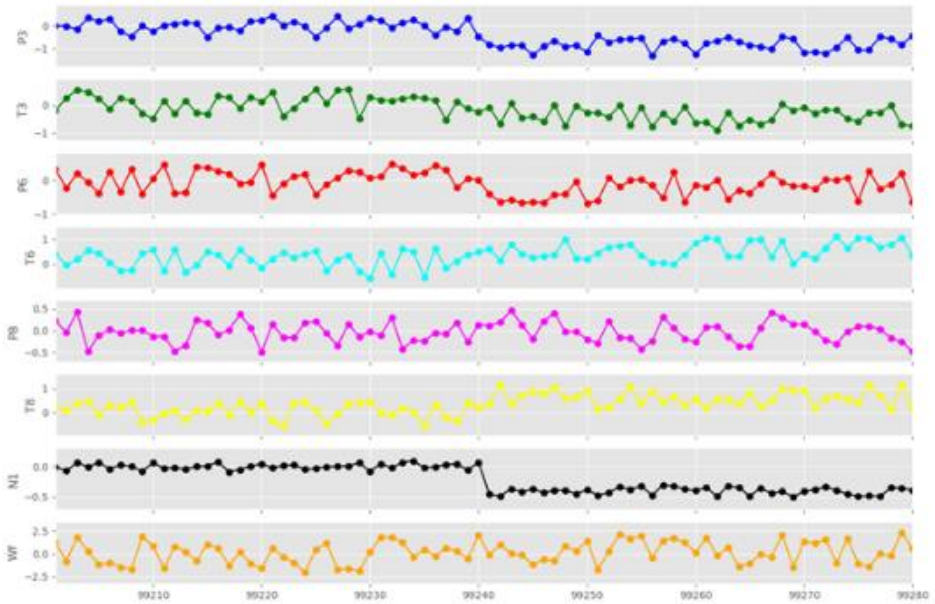


**Fig. 10 The corresponding measurements of Compressor Turbine abrupt fault.**

### D. Data Preprocessing

*1. Data correction*

The parameter correction approach in [21] is adapted to calculate correction exponents for individual parameters. The results obtained for an operating point from the optimization and the associated corrected parameters are presented in Table 6.

**Table 6 Parameter correction exponents and equations for the engine measurements.**

| Parameter | a | b | Corrected parameter |
|---|---|---|---|
| $P_3$ | 0 | 1 | $P_{3C} = P_3/ \delta$ |
| $T_3$ | 0.94 | 0 | $T_{3C}= T_3/ \theta^{0.94}$ |
| $P_6$ | 0 | 1 | $P_{6C}=P_6/\delta$ |
| $T_6$ | 0.85 | 0 | $T_{6C}= T_6/ \theta^{0.85}$ |
| $P_8$ | 0 | 0 | $P_{8C} = P_8/ \delta$ |
| $T_8$ | 1 | 0 | $T_{6C}= T_6/ \theta$ |
| PCN | 0.5 | 0 | $PCNC=PCN/ \theta^{0.5}$ |
| FF | 0.63 | 1 | $FFC = FF/ (\theta^{0.63} \delta)$ |

The corrected parameters are then normalized as shown in the following equation with the delta value introduced into the network for training or testing or diagnosis. The gas-path measurement deviations (M) are computed from the established baseline engine parameters (Mb).

$$\Delta M(\%) = \frac{M - Mb}{Mb} \times 100 \tag{3}$$

*2. Data augmentation*

As shown in Fig. 11, a sliding window method is developed for data augmentation. As a result, 5280*40 = 21120 time series samples are obtained.

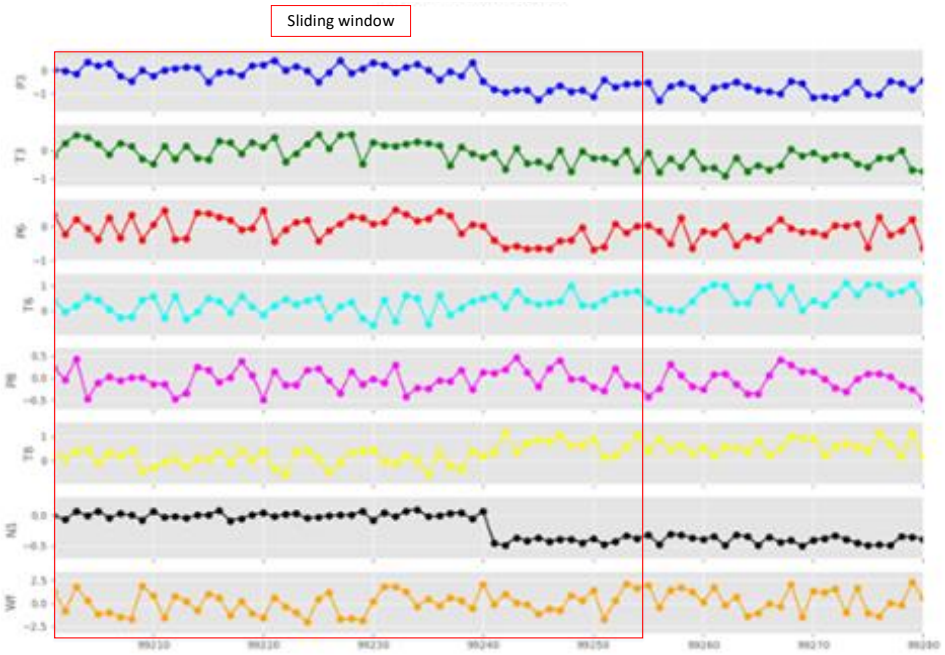


**Fig. 11 Data augmentation with sliding windows.**

*3. Data visualization*

Data visualization is conducted to extract features from the data samples and map the features using visual charts. It is essential to analyze massive amounts of information and make data-driven decisions.

Autoencoders [19] are developed and used for dimensionality reduction or feature learning.

Each fault pattern has 8 measurements. In order to visualize and analyze these fault patterns, a denoising autoencoder as shown in Fig. 12 is developed to project 8-dimensional vectors representing fault patterns to 2- or 3-dimensional vectors. The autoencoder is composed of 5 fully connected layers, with the 3rd layer as a bottleneck with a linear activation function. After trial and error, the structure of 2D autoencoder was found to be 8:15:2:15:8. The 3D autoencoder is almost identical except it has 3 neurons in the bottleneck layer.

Autoencoders were trained on 5280 fault patterns. These vectors of activation were shuffled. Noise in the autoencoders was added to the input vectors and it was drawn from a normal distribution with mean 0 and standard deviation 0.1. The network was trained with the Adam optimizer and the mean-squared error was minimized.

By feeding a fault pattern to the well-trained autoencoder we obtain the activations on the bottleneck. These lower-dimensional bottleneck activations are presented in Fig. 13.

As we can see from the Fig. 13, the samples are not linearly separable. The furthest points are associated with maximum severities, whereas the closest points correspond to low-level faults. As the fault magnitude increases, the overlapping possibilities of the adjacent classes increase. As the number of fault components increases, the overlapping possibilities of the adjacent classes increase. The conventional nonlinear ML classifiers can be used to classify the fault types.

There are 21120 time series samples. Each time series sample has 40 time steps and each time step has 8 sensor measurements. A denoising autoencoder is developed to project 8-dimensional vectors at each time step to 2- or 3-dimensional vectors. Autoencoders are trained on vectors for all single time steps of each sample.

By feeding a sequence of time series data corresponding to a fault sample to the autoencoder we obtain the activations on the bottleneck. We refer to this lower-dimensional bottleneck activations sequence as an example path. In Fig. 14, example paths visualization is presented. Each point in the visualization represents each 2D activation from the autoencoder for a single time step and for one example. The color scale represents the time step (from 0 to 40) and black lines are connecting points from a single example path.

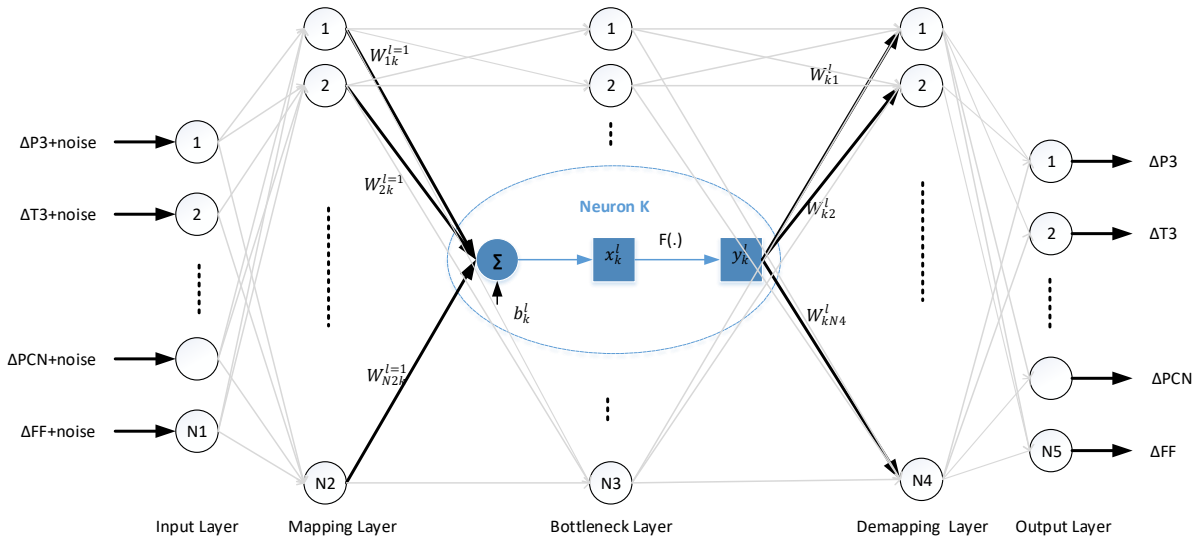


**Fig. 12 A denoising autoencoder for fault pattern visualization.**

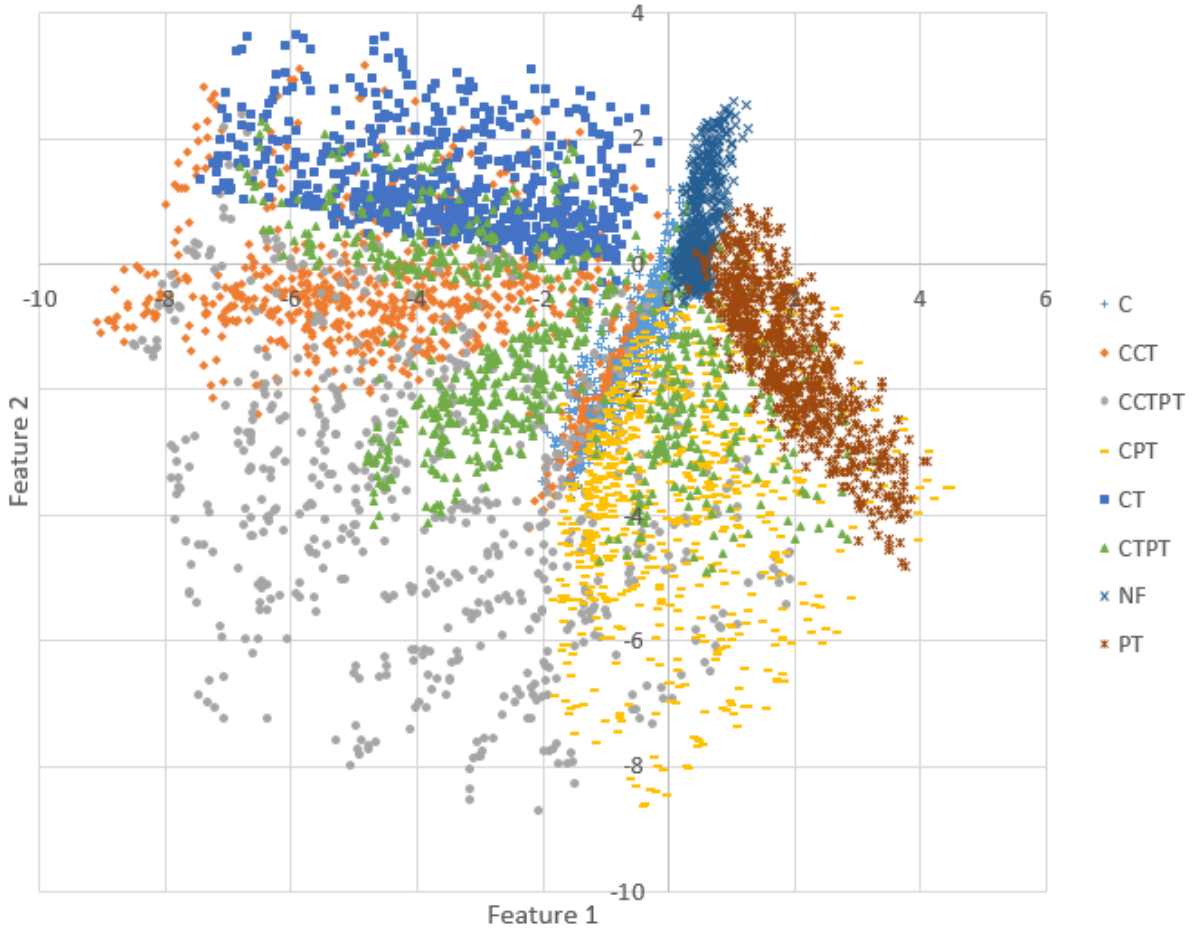


**Fig. 13 Fault pattern 2- dimensional visualization results.**

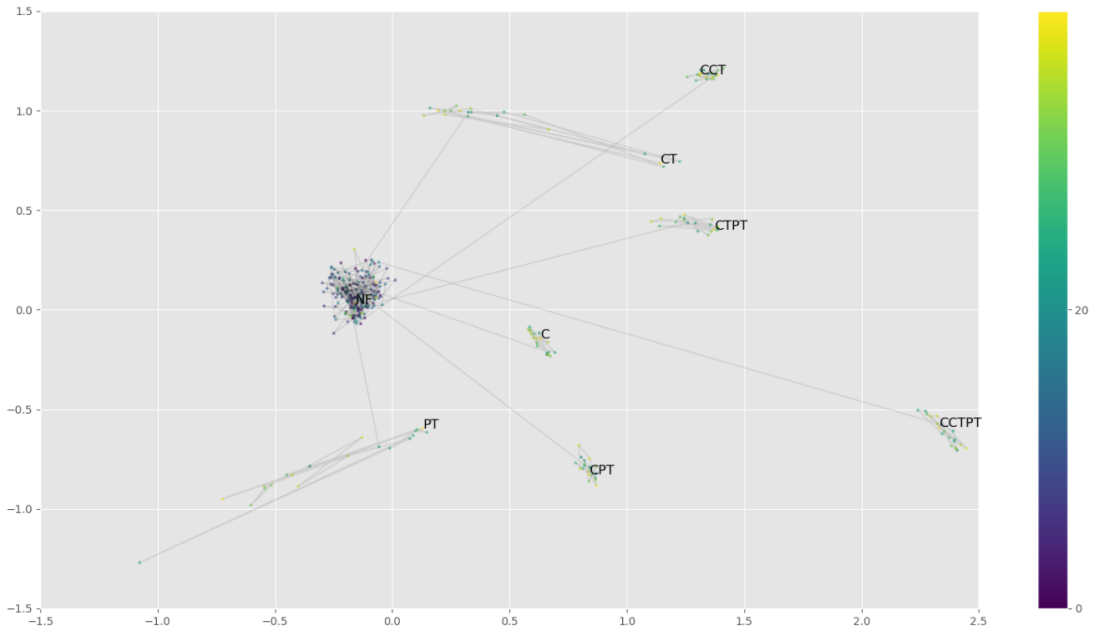


**Fig. 14 Visualization result of 8 time series input samples.**

## V. Experiments

### A. Developing Environments

Our experiments were conducted on a PC with intel Core i7 CPU and we employed a NVIDIA on Quadro P600 GPU processor running on Windows 10. A deep learning tool, Keras, was used to develop the ML methods.

### B. Conventional ML algorithms

Tensors are used as the data structure for ML systems [22]. The training samples of conventional ML algorithms can only be 2D tensors (matrices) with shape (samples, features). A single input sample is 1D tensor (vector), which means there can only be one 0D tensor (scalar) for an input layer neuro.

An initial investigation was undertaken to determine the relative difficulty in detecting and isolating fault types in the system using available ML tools. More specifically, could a MLP accurately learn to classify the difference between a healthy system and a faulty system and isolate the faulty components. MLP is a feed-forward neural network type supervised learning algorithm consisting of input and output layers with one or more hidden layers in between.

*1. MLP1 method*

The data used for MLP1 is the 5280 fault pattern data, by randomly divided it into three sub-samples (70% for training and 15% each for validation and testing).

A grid search method is used to select the optimal hyperparameters. The results are shown in Table 7.

A four layer network using all 8 inputs with three nodes in the first hidden layer and two nodes in the second hidden layer was able to achieve 90.09 % classification accuracy.

This experiment demonstrated that the raw inputs available in the fault pattern data set provide enough information to indicate the health of the system and that a MLP network is very capable of distinguishing faulty components using the available data.

The fault pattern data provides enough information to indicate the health state based on the health parameters the change at a single timestep. However, the fault pattern data cannot provide the required information about a fault. For a fault event, continuous variables representing the fault initiation, grows and magnitudes are needed.

*2. MLP2 method*

The training samples of conventional ML algorithms can only be 2D tensors (matrices) with shape (samples, features). A single input sample is a 1D tensor (vector), which means only one 0D tensor (scalar) can be used for an input layer neuro. However, time series data are 3D tensors with shape (samples, timesteps, features). A single input sample is a 2D tensor (matrix).

**Table 7 MLP1 model optimization.**

| Hyperparameters | Searching ranges | Optimal result |
|---|---|---|
| Layers number | (3, 5, 1) | 4 |
| Units per hidden layer | (1,1000,1) | 8-33-12-8 |
| Optimizer | (rmsprop, adam) | adam |
| Learning rate of the optimizer | default | default |
| Activations | (relu, sigmoid) | relu |
| Last-layer activation | softmax | softmax |
| Loss function | categorical_crossentropy | categorical_crossentropy |
| Epochs | 200 | 200 |
| Batch size | (1, 2000, 1) | 166 |
| Evaluation protocols | (hold-out validation, K-fold cross-validation, iterated K-fold validation) | hold-out validation |

The data that we would like to feed into our network is 2-dimensional (40x8) data. Unfortunately, MLP is not able to process multi-dimensional input data. Therefore we need to "flatten" the data for the input layer into the neural network. Instead of feeding a 2D 40x8 matrix we will feed in a list of 320 values. The reshaped 0D tensor has two different types. The first type is reshaped from time series data with the shape of (40, 8), which means measurements at a timestep are listed in the 0D tensor and then the next timestep. The first type is reshaped from time series data with the shape of (8, 40), which means measurements from a sensor are listed into the 0D tensor and then the next sensor. An example is shown in 15.

The two type samples have the same shape of (211200, 320). Using the same method with the MLP for fault patterns classification, MLPs are built and optimized. MLPs with the shape of 320-100-10-8 are able to achieve 94.1 % classification accuracy for both of the two type samples.

Experimentation showed that the two type samples produce exactly same result. Which means the MLP are not sensitive to the order of time at all. One of the 320 input values is given to a node in the input layer randomly.

### C. FDI using 1DCNN

Time series data are 3D tensors with shape (samples, timesteps, features). A single input sample is a 2D tensor (matrix).

The CNN architecture is shown as Fig. 16. Based on some well-known CNN architectures and cross-validation results, four convolution layers, one max-pooling layer, one global average pooling layer, and one fully-connected layer are used in this paper.

Convolutions are usually performed after padding the feature maps. Padding means adding some values to the edge of the input matrix. On the one hand, it can increase the size of feature maps so that it is a useful way to increase the model depth. On the other hand, padding in feature maps can better use the border information of feature maps, which is beneficial for the detection performance. No-padding means there is no padding operation. A typical padding method in CNNs is zero-padding.

Bayesian Optimization is used for the hyper parameters optimization. As is shown in Fig. 17, all the hyper parameters are decided after optimization.

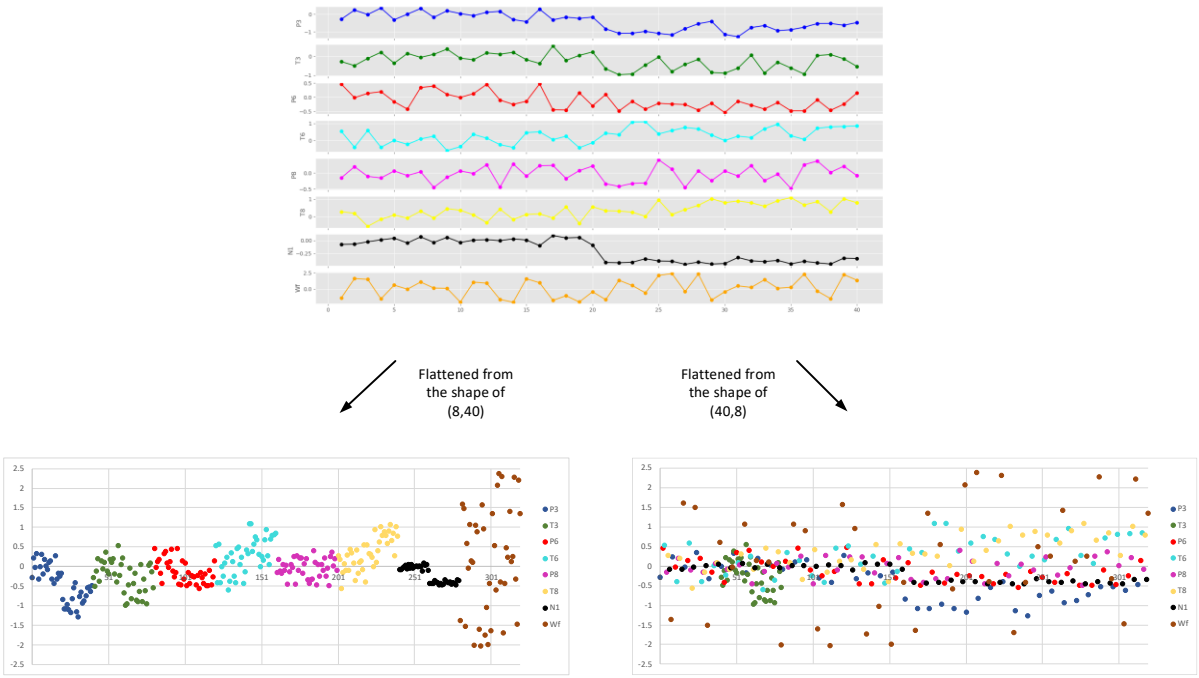


**Fig. 15 Two types of flattened time series samples.**

**Table 8 1DCNN model optimization.**

| Hyperparameters | Searching ranges | Optimal result |
|---|---|---|
| Convolutional Layers number | (1, 6, 1) | 3 |
| Pooling Layers number | (1, 6, 1) | 1 |
| Dropout layer rate | (0.0, 0.9, 0.1) | 0.26 |
| Conv1d_1 filter number | (50, 200, 1) | 58 |
| Conv1d_1 kernel size | (2, 10, 1) | 7 |
| Conv1d_2 filter number | (50, 200, 1) | 161 |
| Conv1d_2 kernel size | (2, 10, 1) | 7 |
| Conv1d_3 filter number | (50, 200, 1) | 58 |
| Conv1d_3 kernel size | (2, 10, 1) | 7 |
| Maxpooling1d pool_size | (2, 3, 1) | 2 |
| Optimizer | (rmsprop, adam) | adam |
| Learning rate of the optimizer | default | default |
| Activations | (relu, sigmoid) | relu |
| Last-layer activation | softmax | softmax |
| Loss function | categorical_crossentropy | categorical_crossentropy |
| Epochs | 1000 | 1000 |
| Batch size | (1, 2000, 1) | 166 |
| Padding | (no-padding, zero-padding) | zero-padding |
| Evaluation protocols | (hold-out validation, K-fold cross-validation, iterated K-fold validation) | hold-out validation |

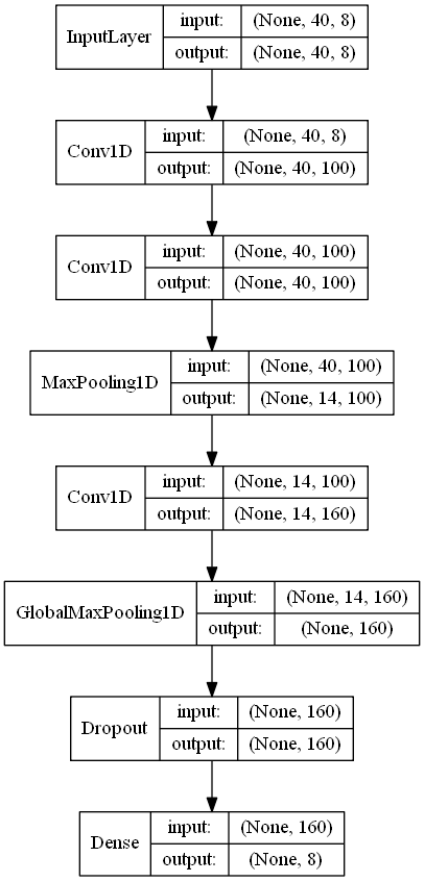


**Fig. 16 1DCNN architecture.**

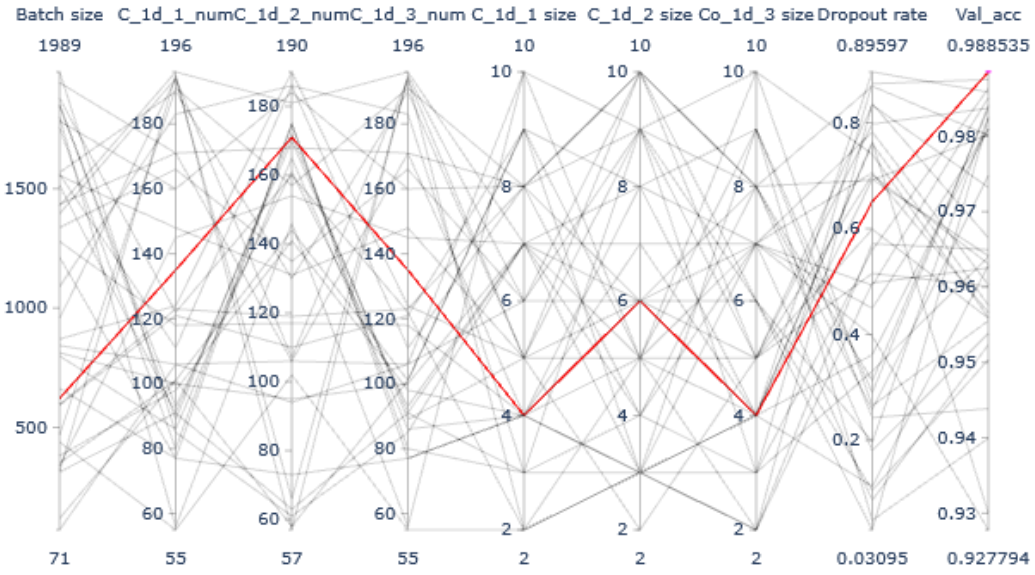


**Fig. 17 Results of Hyper Parameters Optimization.**

## D. Results

*1. Evaluation Metrics*

The evaluation metrics are presented in Appendix 1. The results are shown in Fig. 18 for easy understanding.

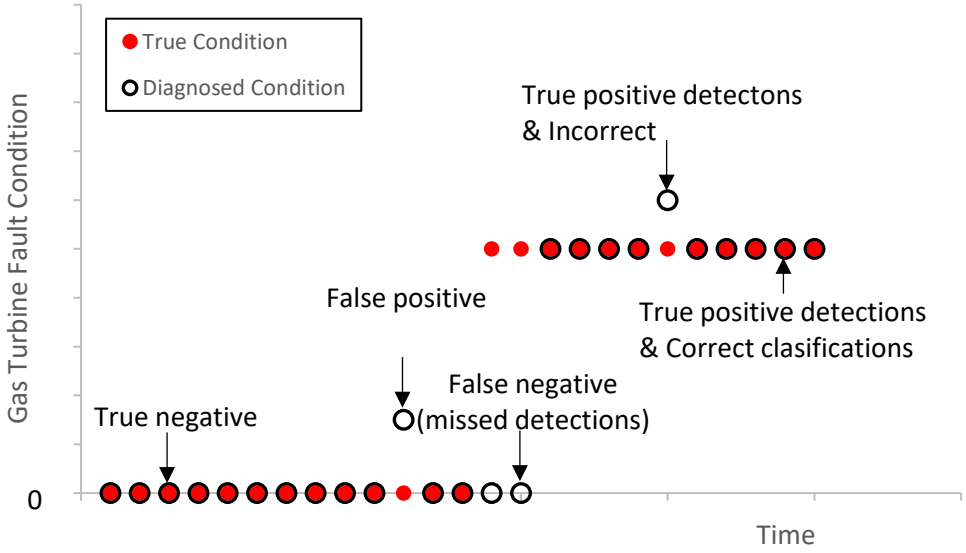


**Fig. 18 An example of fault FDI result.**

*2. Evaluation Results*

**Table 9 1DCNN-based algorithm metrics.**

Abrupt Fault Cases (all)

Confusion Matrix
Predicted State

| True State | No Fault | C | T | PT | CCT | CPT | CTPT | CCTPT | Accuracy | Detect Latency | Classify Latency |
|---|---|---|---|---|---|---|---|---|---|---|---|
| No Fault | 0.9997 | 1E-02 | 6E-04 | 3E-04 | 0 | 2E-03 | 4E-03 | 0 | 99.97% | N/A | N/A |
| C | 7E-03 | 0.9976 | 2E-02 | 0 | 3E-03 | 3E-03 | 4E-04 | 0 | 99.76% | 0.64 | 0.21 |
| CT | 1E-02 | 2E-02 | 0.9963 | 2E-02 | 0 | 3E-02 | 0 | 0 | 99.63% | 0.80 | 0.36 |
| PT | 5E-03 | 0 | 6E-03 | 0.9970 | 0 | 9E-04 | 2E-04 | 4E-03 | 99.70% | 0.75 | 0.31 |
| CCT | 0 | 0 | 0 | 0 | 0.9921 | 7E-03 | 6E-04 | 0 | 99.21% | 1.26 | 0.22 |
| CPT | 1E-03 | 1E-02 | 1E-02 | 5E-03 | 4E-03 | 0.9908 | 7E-04 | 2E-04 | 99.08% | 1.75 | 0.30 |
| CTPT | 0 | 0 | 0 | 0 | 0 | 2E-03 | 0.9752 | 0 | 97.52% | 2.55 | 0.56 |
| CCTPT | 0 | 0 | 0 | 8E-04 | 0 | 0 | 5E-03 | 0.9630 | 96.30% | 2.74 | 0.30 |

Decision Matrix
Predicted State

| True State | Fault | No Fault | Detect Latency |
|---|---|---|---|
| Fault | 0.9991 | 9E-04 | 1.10 |
| No Fault | 3E-04 | 0.9997 | N/A |

| Kappa Coefficient |
|---|
| 0.9726 |

The kappa coefficient is an overall evaluation of the FDI performance. For the case shown in Table 9, Table 11, and Table 10, the kappa coefficient for 1DCNN algorithm is 0.9726, which is 0.0615 higher than the MLP2 algorithm. Missed detection, false alarm rate and time latency are three key factors for the FDI system. The missed detection rate for 1DCNN algorithm is 0.00009. The false alarm rate is 0.00003. The detection latency for 1DCNN is 1.10. All these three factors are great improvement on the MLP algorithms.

**Table 10 Comparisons among the studied algorithms.**

| Algorithm | TPR | TNR | Detection Latency | Kappa |
|---|---|---|---|---|
| MLP1 | 90.11±0.23% | 89.67±0.37% | 2.53 | 0.9033 |
| MLP2 | 92.17±0.54% | 90.12±0.36% | 2.34 | 0.9111 |
| 1DCNN | 99.91 ±0.07% | 99.97±0.02% | 1.10 | 0.9726 |

**Table 11 MLP2-based algorithm metrics.**

Abrupt Fault Cases (all)

Confusion Matrix
Predicted State

| True State | No Fault | C | T | PT | CCT | CPT | CTPT | CCTPT | Accuracy | Detect Latency | Classify Latency |
|---|---|---|---|---|---|---|---|---|---|---|---|
| No Fault | 0.96 | 7E-03 | 1E-02 | 5E-03 | 0 | 1E-02 | 3E-03 | 7E-08 | 96% | N/A | N/A |
| C | 9E-03 | 0.88 | 5E-02 | 0 | 1E-02 | 6E-02 | 0 | 0 | 88% | 2.57 | 1.12 |
| CT | 5E-03 | 9E-02 | 0.82 | 3E-02 | 0 | 6E-02 | 0 | 0 | 82% | 2.76 | 1.23 |
| PT | 3E-02 | 0 | 4E-02 | 0.84 | 0 | 6E-02 | 0 | 3E-02 | 84% | 2.12 | 1.88 |
| CCT | 0 | 9E-03 | 0 | 0 | 0.98 | 1E-02 | 0 | 0 | 98% | 3.11 | 1.95 |
| CPT | 6E-03 | 3E-02 | 3E-02 | 1E-02 | 1E-02 | 0.90 | 0 | 8E-03 | 90% | 3.57 | 2.12 |
| CTPT | 3E-03 | 0 | 0 | 0 | 0 | 3E-03 | 0.99 | 0 | 99% | 1.78 | 1.43 |
| CCTPT | 0 | 0 | 0 | 1E-02 | 0 | 1E-02 | 0 | 0.98 | 98% | 3.21 | 1.33 |

Decision Matrix
Predicted State

| True State | Fault | No Fault | Detect Latency |
|---|---|---|---|
| Fault | 0.9217 | 0.0783 | |
| No Fault | 0.0988 | 0.9012 | N/A |

| Kappa Coefficient |
|---|
| 0.9111 |

## VI. Conclusion

Work conducted to date includes: (1) An engine model similar to GE LM2500+ was developed. (2) Data for training, validation and test were generated under different operating conditions. White noise of Gaussian distribution was combined in order to simulate the sensor noise. (3) Data preprocessing: to avoid the effects of the ambient condition variation on the gas-path measurement deviations, all the measurement parameters were corrected to standard operating conditions SLS conditions. A sliding window method was developed to augment the time series data. (4) Data visualization was conducted to extract features from the data samples and map the features using a method based on a denoising autoencoder. It is essential to analyze massive amounts of information and make data-driven decisions. (5) A 1DCNN based method with four convolution layers, one max-pooling layer, one global average pooling layer, and one fully-connected layer were developed. It is able to achieve 98.57±0.02 % overall accuracy after 166 seconds of training on Quadro P600 GPU processor running on Windows 10. (6) Comparison: the training samples of conventional ML algorithms can only be 2D tensors (matrices) with shape (samples, features). Time series data which are 3D tensors with shape (samples, timesteps, features) should be "flattened" into 2D tensors (matrices) with shape (samples, timesteps × features). A MLP with the shape of 320-100-10-8 was developed using the flattened data. It is able to achieve 90.12±0.36 % overall accuracy.

The results of this paper indicate that the fault detection and isolation for gas turbine components based on 1DCNN is much better than the typical methods. It is proved that the modification of related sensors readings in time series are the key features which can distinguish between different fault classes appearing in different locations, and the convolution operation successfully preserves the key features. Then with the pooling operation, the shift and noise of the key features is eliminated. CNN extracts the information between sensors time series and eliminates the impact of feature shift and noise.

## Appendix

Appendix 1 Evaluation metrics

1. True Positive Rate (number of correct fault detections divided by the number of fault cases)
2. False Negative Rate (number of incorrect no fault detections divided by the number of fault cases)
3. False Positive Rate (number of incorrect fault detections divided by the number of no fault cases)
4. True Negative Rate (number of correct no fault detections divided by the number of no fault cases)
5. Correct Classification Rate (number of correct classifications of a fault divided by the number of cases of that fault)
6. (Incorrect) Misclassification Rate (number of incorrect classifications of a fault divided by the number cases of that fault)
7. Detection latency
8. Classification latency
9. Kappa Coefficient: the function cohen_kappa_score computes Cohen's kappa statistic. This measure is intended to compare labelings by different human annotators, not a classifier versus a ground truth. The kappa score is a number between -1 and 1. Scores above 0.8 are generally considered good agreement; zero or lower means no agreement (practically random labels). The Kappa Coefficient, denoted here as κ, is calculated from the elements of the un-normalized confusion matrix, C, as shown in the equation below. The two subscript indices represent the row and column corresponding to individual confusion matrix elements

$$k = \frac{N(corrected\ classified) - N(expected\ corrected\ by\ chance))}{N(total) - N(xpected\ corrected\ by\ chance)} \tag{4}$$

Where

$$N(corrected\ classified) = \sum_{p=1}^{n} C_{pp} \tag{5}$$

$$N(total) = \sum_{p=1}^{n} \sum_{q=1}^{n} C_{pq} \tag{6}$$

$$N(xpected\ corrected\ by\ chance) = \sum_{p=1}^{n} \{ \sum_{q=1}^{n} \frac{C_{pq}}{N(total)} . \sum_{q=1}^{n} C_{pq} \} \tag{7}$$

## References


[1] Lipowsky, H., Staudacher, S., Bauer, M., and Schmidt, K. J. "Application of Bayesian Forecasting to Change Detection and Prognosis of Gas Turbine Performance." *Proceedings of the ASME Turbo Expo*, Vol. 1, No. March 2010, 2009, pp. 587–596. https://doi.org/10.1115/GT2009-59447.

[2] Simon, D. L. "Propulsion Diagnostic Method Evaluation Strategy (ProDiMES) User's Guide." NASA TM-2010–21584, 2010.

[3] Sarkar, S., Jin, X., and Ray, A. "Data-Driven Fault Detection in Aircraft Engines with Noisy Sensor Measurements." *Journal of Engineering for Gas Turbines and Power*, Vol. 133, No. 8, 2011, pp. 1–10. https://doi.org/10.1115/1.4002877.

[4] Volponi, A. J., and Tang, L. "Improved Engine Health Monitoring Using Full Flight Data and Companion Engine Information." *SAE International Journal of Aerospace*, Vol. 9, No. 1, 2016, pp. 91–102. https://doi.org/10.4271/2016-01-2024.

[5] De Bruijn, B., Nguyen, T. A., Bucur, D., and Tei, K. "Benchmark Datasets for Fault Detection and Classification in Sensor Data." *Proceedings of the 5th International Confererence on Sensor Networks*, 2016, pp. 185–195. https://doi.org/10.5220/0005637901850195.

[6] Jombo, G., Zhang, Y., Griffiths, J. D., and Latimer, T. "Automated Gas Turbine Sensor Fault Diagnostics." *Proceedings of ASME Turbo Expo 2018 Turbomachinery Technical Conference and Exposition*, Oslo, Norway, 2018, p. V006T05A003. https://doi.org/10.1115/gt2018-75229.

[7] Sarkar, S., Rao, C., and Ray, A. "Estimation of Multiple Faults in Aircraft Gas-Turbine Engines." *Proceedings of the American Control Conference*, Vol. 223, 2009, pp. 216–221. https://doi.org/10.1109/ACC.2009.5159981.

[8] Jaw, L. C., and Lee, Y. "Engine Diagnostics in the Eyes of Machine Learning." *Proceedings of ASME Turbo Expo 2014: Turbine Technical Conference and Exposition*, Düsseldorf, Germany, 2014.

[9] Loboda, I. "Neural Networks for Gas Turbine Diagnosis." *Intech*, 2016, pp. 195-220. https://doi.org/http://dx.doi.org/10.5772/57353.

[10] Loboda, I., and Olivares Robles, M. A. "Gas Turbine Fault Diagnosis Using Probabilistic Neural Networks." *International Journal of Turbo and Jet Engines*, Vol. 32, No. 2, 2015, pp. 175–191. https://doi.org/10.1515/tjj-2014-0019.

[11] Amare, D. F., Aklilu, T. B., and Gilani, S. I. "Gas Path Fault Diagnostics Using a Hybrid Intelligent Method for Industrial Gas Turbine Engines." *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, Vol. 40, No. 12, 2018, pp. 1–17. https://doi.org/10.1007/s40430-018-1497-6.

[12] Batayev, N. "Gas Turbine Fault Classification Based on Machine Learning Supervised Techniques." *Proceedings of 14th International Conference on Electronics Computer and Computation, ICECCO 2018*, 2019, pp. 206–212. https://doi.org/10.1109/ICECCO.2018.8634719.

[13] Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., and Gao, R. X. "Deep Learning and Its Applications to Machine Health Monitoring." *Mechanical Systems and Signal Processing*, Vol. 115, 2019, pp. 213–237. https://doi.org/10.1016/j.ymssp.2018.05.050.

[14] Khan, S., and Yairi, T. "A Review on the Application of Deep Learning in System Health Management." *Mechanical Systems and Signal Processing*, Vol. 107, 2018, pp. 241–265. https://doi.org/10.1016/j.ymssp.2017.11.024.

[15] Luo, H., and Zhong, S. "Gas Turbine Engine Gas Path Anomaly Detection Using Deep Learning with Gaussian Distribution -Autoencoder." *Proceedings of 2017 Prognostics and System Health Management Conference, Harbin*, 2017. https://doi.org/10.1109/PHM.2017.8079166.

[16] Tamilselvan, P., and Wang, P. "Failure Diagnosis Using Deep Belief Learning Based Health State Classification." *Reliability Engineering and System Safety*, Vol. 115, 2013, pp. 124–135. https://doi.org/10.1016/j.ress.2013.02.022.

[17] Liu, J., Liu, J., Yu, D., Kang, M., Yan, W., Wang, Z., and Pecht, M. G. "Fault Detection for Gas Turbine Hot Components Based on a Convolutional Neural Network." *Energies*, Vol. 11, No. 8, 2018. https://doi.org/10.3390/en11082149.

[18] Heimes, F., Heimes, F. O., and Systems, B. A. E. "Recurrent Neural Networks for Remaining Useful Life Estimation Recurrent Neural Networks for Remaining Useful Life Estimation." *Proceedings of 2008 International Conference on Prognostics and Health Management*, November 2008, pp. 227–234., 2016. https://doi.org/10.1109/PHM.2008.4711422.

[19] Ian Goodfellow, Yoshua Bengio, Courville, A. *Deep Learning*, Forest Sciences, 2009.
[20] Jasmani, M. S., Li, Y.-G., and Ariffin, Z. "Measurement Selections for Multicomponent Gas Path Diagnostics Using Analytical Approach and Measurement Subset Concept." *Journal of Engineering for Gas Turbines and Power*, Vol. 133, No. 11, 2011, p. 111701. https://doi.org/10.1115/1.4002348.
[21] Ogaji, S. O. T. "Advanced Gas-Path Fault Diagnostics for Stationary Gas Turbines" Ph.D. Dissertation, Cranfield University, 2003.
[22] Ketkar, N. *Deep Learning with Python*, Manning Publications Co., New York, 2017.