Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Fog computing application of cyber-physical models of IoT devices with symbolic approximation algorithms

Deok-Kee Choi*

Abstract

Smart manufacturing systems based on cloud computing deal with large amounts of data for various IoT devices, resulting in several challenges, including high latency and high bandwidth usage. Since fog computing physically close to IoT devices can alleviate these issues, much attention has recently been focused on this area. Fans are nearly ubiquitous in manufacturing sites for cooling and ventilation purposes. Thereby, we built a fan system with an accelerometer installed and monitored the operating state of the fan. We analyzed time-series data transmitted from the accelerometer. We applied machine learning under streaming data analytics at the fog computing level to create a fan's cyber-physical model (CPM). This work employed the symbolic approximation algorithm to approximate the time series data as symbols of arbitrary length. We compared the performance of CPMs made with five time-series classification (TSC) algorithms to monitor the state of the fan for anomalies in real time. The CPM made with the BOSS VS algorithm, a symbol approximation algorithm, accurately determined the current state of the fan within a fog computing environment, achieving approximately 98% accuracy at a 95% confidence level. Furthermore, we conducted a posthoc analysis, running statistical rigor tests on experimental data and simulation results. The workflow proposed in this work would be expected to be utilized for various IoT devices in smart manufacturing systems.

Keywords: Fog computing, Cyber-physical model, Symbolic approximation algorithms, Smart manufacturing, IoT, Machine learning

Introduction

In cloud computing, smart manufacturing is built with various IoT devices. A managing intelligence observes the states of devices and takes necessary actions immediately to prevent disruption in manufacturing if any problems arise. This intelligence incorporated with domain-specific knowledge can be labeled as a cyber-physical model (CPM) [1]. However, there are some practical difficulties in creating CPM for real-world phenomena. For example, IoT devices are spatially densely installed in a manufacturing environment; thus, lessening device heat is essential in manufacturing. Although a fan is primarily used for cooling or ventilation, a study on cyber-physical modeling of fans has not drawn as much attention as installed devices. If the fan stops working; as a result, the device's temperature may rise and eventually fail to function correctly, causing significant interruptions to the entire manufacturing system. It is reported that a cooling fan is one of the top 10 failing components in electronic products [2]. Therefore, a study on the state of a cooling fan in smart manufacturing would be practically meaningful.

In general, the state of a fan can be monitored by analyzing data sent from sensors alongside devices on the site. The transmitted data from the sensor is Time Series (TS) with a temporal structure, so it requires non-conventional algorithms than those used for typical tabular data. A conventional data analytics algorithm, including machine learning, is suitable for dealing with

*Correspondence: dkchoi@dankook.ac.kr

Department of Mechanical Engineering, Dankook University, 152, Jukjeon-ro, Suji-gu, 16890 Yongin-si, Gyeonggi-do, Republic of Korea

pooled data, making it inappropriate to apply directly to TS. The pooled data is data that, unlike TS, does not consider the temporal sequence. As a non-conventional algorithm, Time Series Classification (TSC) algorithm [3] has been known as an algorithm that best fits TS for classification because it takes raw TS to keep its temporal structure intact. The role of the TSC algorithm is to compare TS sent from the sensor and classify it by predefined class. In this way, this algorithm helps to get the fan's state in real time. TSC algorithms can be regarded as the core of CPM, determining which of a set of designated classes in TS belongs. The classes for classification here are defined by multiple collections of data for machine learning.

TSC algorithms can be divided into three types according to how to interpret TS [4]. The first is the similarity-based technique, taking raw TS as input. However, this technique has disadvantages, such as relatively long computation time and weak scalability. The second type is conventional machine learning, disregarding the temporal structure of TS. However, in so doing, most of the information related to the time domain may be lost. A third type, the symbolic approximation technique, has recently drawn much attention, in which raw TS is converted into symbols of arbitrary length. This type's advantage is the dramatic reduction of high dimensionality and noise. The BOSS algorithm has been recognized as a state-of-the-art [5], which was adopted for creating the final CPM in this work. This algorithm runs a sliding window across each series, discretizes the window to form a word, develops a histogram of word counts over the dictionary, then constructs a classifier of the histograms. Once a TSC algorithm is chosen, the next would be having physical laws incorporated with CPM.

Thereby, in creating CPM, domain-specific knowledge plays a crucial role [6–9]. The complexity of phenomena occurring around the cooling fan can immediately go beyond our understanding in a real-world situation. Hence, there are barriers to establishing and solving physical laws or governing equations to determine the effect of each unknown element on the state of the fan. There are too many unknowns associated with physical interactions, even for a simple IoT device like the fan. Hence, setting up and directly solving such equations is not easy. We employed machine learning with TSC algorithms, a data-driven modeling approach, to resolve the complicated issues in this work. We set up a cooling fan system with an accelerometer installed for collecting experimental data. The physical elements related to the state of the fan can be acoustic noise, wind speed, rotation, vibration, counter-wind, and pressure change, as shown in Fig. 1. In order to set up data-driven modeling, devices for data generation are required, such as sensors. The larger the number of sensors mounted, the more accurate the classification model is. However, as the number of transmitted data increases and the uncertainties grow, making a lightweight CPM work efficiently over limited computing resources makes it challenging.

Upon considering the implementation of intelligence [10] onto a less powerful computing environment, CPM

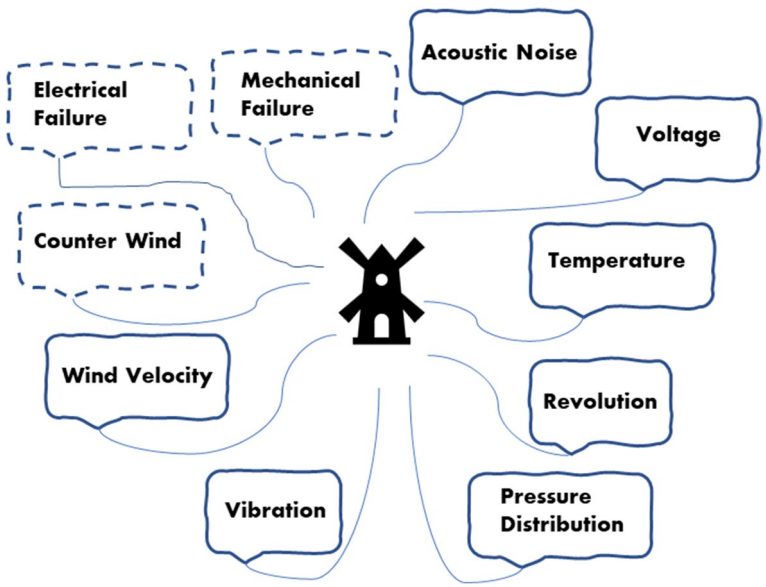

Fig. 1 A cyber-physical model is expected to sufficiently explain the effect of real-world elements such as acoustic noise, mechanical failure, temperate change, revolution (rpm), the pressure distribution of blades, vibration, counter-wind occurrence, and wind velocity, etc

must be sufficiently adequate and lightweight compared to that in cloud computing. Fog computing refers to a sublayer of an IoT system shown in Fig. 2. By CPM running in fog computing, we may expect IoT devices' operating states to be fully aware without relying heavily on cloud computing [11]. In cloud computing, efforts are being made to ensure that heavy loads are efficiently distributed as various applications require immediate attention in real-time [12, 13]. Thus, much research in recent years has been focused on smart manufacturing in fog computing [14, 15].

Several advantages of CPM created in this work can be addressed: Firstly, it can dramatically reduce the amount of data directly transmitted from the sensor to cloud computing. In other words, an intelligent CPM close to the sensor can self-determine the device's state and notify the administrator only at critical moments. Secondly, the types of sensors and data are as diverse and heterogeneous as the devices used in smart manufacturing. For example, the dispatch cycle of data sent by sensors is generally not the same. While there are sensors that send data more frequently, sensors send at considerable time intervals. Therefore, if intelligence can be materialized to account for this time interval, the problem of inconsistency over the data transfer cycle can also be solved spontaneously. This discrepancy is also directly related to latency and bandwidth in cloud computing. Therefore, intelligence operating on fog computing is vital, which can alleviate the heavy burden on cloud computing.

We demonstrated how to create an intelligent CPM for the cooling fan that worked with great accuracy, demanding the least amount of computing resources. As a result, we learned that in fog computing, which has a minimal computing resource compared to cloud computing, CPM could be used to ascertain the state of the fan with the high performance considering possible uncertainties in real-world situations. Therefore, the workflow proposed in this work may increase overall production efficiency by expanding the utilization of fog computing in smart manufacturing.

The paper is organized as follows: The Related work section discusses the related work. A detailed explanation of how to set up CPM, closely following Bayes' rule, is shown in CPM set up with Bayes' rule section. The Symbolic Fourier Approximation (SFA) section explains the symbolic approximation algorithm with sub-algorithms. Bag-of-SFA-Symbols VS section presents the BOSS algorithm and an extended BOSS VS algorithm. Results and discussion section explains the experimental implementation of the fan with the sensor, the exploratory data analysis, and multiple comparisons of CPM types with different algorithms. Lastly, the Conclusion section concludes this paper and discusses the further study.

Related work

This study aims to create a cyber-physical model (CPM) [16] that can accurately determine the operating state of the cooling fan as an IoT device in fog computing. Time series classification (TSC) algorithms that can correctly interpret the time series (TS) transmitted from the accelerometer attached to the fan are required. Domain-specific knowledge, which can describe the fan's characteristics, is also essential. A time-series T_i is defined as an ordered sequence $T_i = \{t_1, \ldots, t_n\}$. The univariate TS has only one feature ($d = 1$); on the other hand, the multivariate TS has several features ($d > 1$), that is $\forall j, t_j \in \mathcal{R}^d$, where d is the number of features.

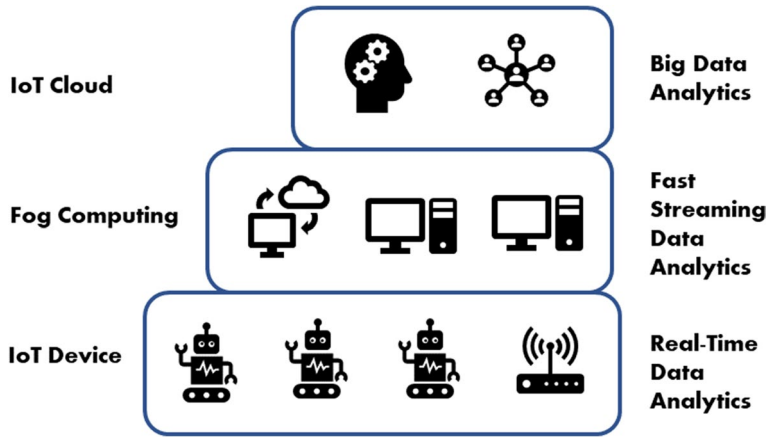

Fig. 2 Three levels of IoT system. As the top-level, IoT employing a cloud system associated with big data analytics. Fog computing resides in the middle, equipped with fast streaming data analytics. At the bottom level, IoT devices such as sensors are located with consecutive temporal data requiring real-time data analytics

Although research on TSC has drawn much attention for years, most results are related to accuracy improvement and not scalability. However, recent IoT sensor-driven applications have to deal with precisely these scaled data in classification, making methods unimportant that do not scale [17, 18]. In particular, TSC working in fog computing has to meet stringent limits over the usage of resources. Therefore, TSC algorithms should be lightweight, fast, and accurate in a fog computing context. Next, the three most widely used TSC algorithms are discussed.

Firstly, One-Nearest-Neighbor and Dynamic Time Warping (1-NN DTW) [19] is a TSC algorithm that defines the similarity among multiple TS by calculating the distance of each element of TS for comparison. The principle of the algorithm is that the farther away the two TS data are in distance, the less alike they are. Thus, from the perspective of TSC, it is easy to determine to which class a given TS belongs. The algorithm has been used for years because it is straightforward and intuitive. Moreover, this algorithm is still used as a baseline to validate the performance of newly proposed algorithms; however, as the number of data increases, the amount of computation and the memory required grow, making it challenging to utilize in fog computing.

The second type of TSC includes machine learning techniques such as KNN [20], artificial neural networks (ANN) [21], and Long Short-Term Memory (LSTM) [22]. However, those algorithms may not be considered genuine TSC from a strict perspective because they do not consider TS's temporal structure. Therefore, valuable information on the time domain might be lost. In addition, this algorithm is commonly called a black box type because it is intricate to understand the process by which the results are derived.

Thirdly, the symbolic approximation algorithm has attracted much attention recently. The algorithm has brought a breakthrough in dealing with large amounts of TS. The fundamental principle of the algorithm is to divide continuous TS into appropriate intervals and replace it with symbols of arbitrary length. Bag-Of-SFA-Symbols (BOSS) [23, 24], Symbolic Aggregate approximation (SAX) [25, 26], and ExtrAction for time SEries cLassification (WEASEL) [27] have been most famous. WEASEL builds on the bag-of-patterns (BOP) approach. It activates a sliding window over TS and extracts features in each window fed into TSC on machine learning training. Still, we found that the BOP method tends to take longer in computation than the BOSS algorithm [28]. SAX is a symbolic approximation method used in many applications. In SAX, Piecewise Aggregate Approximation (PAA) is utilized to discretize the mean value of each segment. However, this approach considers only the mean value of each segment; thereby, it often may fail to distinguish different time series with similar mean values, resulting in relatively less accurate classification. Furthermore, it has the disadvantage of constantly recalculating means for new data. The BOSS algorithm is based on Symbolic Fourier Approximation (SFA) [29], and SFA consists of Discrete Fourier Transform (DFT) for approximation and Multiple Coefficient Binning (MCB) for quantization. BOSS is an algorithm running a sliding window across each series, discretizing the window to form a word, developing a histogram of word counts over the dictionary, then constructing a classifier in the form of the histograms.

As for the subject of TSC, two contributions, which we made through this work, can be presented as follows: The first is about extending algorithms into ones that can deal with multivariate TS. Most of the previously published studies on TSC relate to the univariate TS. However, TS transmitted from IoT devices is more likely multivariate TS [30]. For example, this work transmitted multivariate TS data from a multi-channel acceleration sensor mounted on the fan. TSC handling the multivariate TS is much more complex than that for the univariate TS because they also need to find correlations between multiple TS data at each time element. We employed BOSS, WEASEL, and other algorithms to cope with the issue and modified algorithms to address multivariate TS. Eventually, the BOSS algorithm, recognized as state of the art in the field, resulted in the best CPM for accuracy and scalability in this work. The second is of use of real-world data mixed with noise in experiments. In papers published on TSC algorithms [31, 32], most results were obtained using publicly available data in the UCR machine learning repository [33]. In order to compare and analyze the various algorithms, it is common to verify the results using well-known data. However, there is an issue that domain-specific knowledge, which can account for real-world phenomena, may not be directly applicable to CPM because using the data provided by the UCR repository is not directly experimented with by researchers. Thereby, it is believed that using data with much noise obtained in real-world situations might yield somewhat different results, which is also pointed out in literature as a concerning issue [34]. Therefore, we established an experimental system in this study and considered real-world data that might be observed in actual manufacturing instead.

CPM should be created for a comprehensive account of real-world phenomena using as much information as possible about the fan, commonly referred to as domain-specific knowledge. The traditional way is to obtain information through experiments, and numerical simulations [35]. Computer simulations of a fan performance require

physical laws or government requirements related to fluid dynamics. Coggiola et al. [36] employed the computational fluid dynamics that interprets the flow equations: Steady and Unsteady Reynolds-averaged Navier-Stokes equations [37]. However, since these equations are given in partial differential equations with many variables for inputs, it is not easy to solve even numerically and require considerable computing resources [38]. In addition, multiple-input variables are needed to solve the equations, for example, a fan's velocity, pressure, temperature, and humidity. For these input variables, sensors capable of measuring corresponding physical elements are required. In other words, different sensors for flow velocity, pressure change, temperature variance, and humidity measurement are needed, respectively. Furthermore, as the number of sensors increases, the creation of CPM can become more involved.

Thus, efforts to create CPM recently have received significant attention by actively exploiting data from sensors. For example, to comprehend the effect of the vibration of a fan, another physical law that can relate the vibration and flow through a fan is required, which may not be known to date. Consequently, traditional methods have encountered many complications that cannot be quickly answered. Oh et al. [39] monitored the state of a cooling fan by applying three parameters: acoustic noise mission, shaft rotational speed, and current. However, it requires at least three different sensors. For better accuracy of models, other parameters need to be sought that can affect the state of the fan, and it might take quite some time to find out. Jin et al. [40] used vibration signals to determine the fan's state and find similarity by calculating Mahalanobis distance (MD) for TS data to compare with a baseline. However, this method has the disadvantage of being computationally intensive and weak to scalability as a TSC, similar to the 1-NN DTW introduced earlier. Once a data-driven modeling approach is employed rather than the conventional one, domain-specific knowledge must be assimilated into CPM somehow. Practically, domain-specific knowledge could be implemented by determining the type or number of sensors mounted on the fan. Therefore, machine learning techniques can be utilized to analyze data to find essential features. Consequently, much domain-specific knowledge is needed when building a model. This study installed an accelerometer that detects the fan's movement to identify the fan's state. Thus, some domain-specific knowledge may come from the accelerometer. The next task is how we attained a lightweight CPM working in fog computing.

As the type and number of connected devices increases, cloud computing becomes more challenging to manage. Such problems include latency, bandwidths, and scalability [41]. If the latency between cloud computing and the device is high, it is not easy to cope with real-time events. In addition, specific IoT devices often transmit more data than others, which poses a significant challenge in cloud computing bandwidth. Moreover, scalability issues cannot be overlooked because large amounts of data must be processed at a time. Thus, it would be more convenient in many ways to implement intelligence in fog computing near devices so that we can be fully aware of the situation without relying heavily on cloud computing. Accordingly, much research in recent years has been focused on smart manufacturing applications in fog computing: IoT deployment in fog computing [42], IoT data scheduling in fog computing [43], IoT task scheduling in fog computing [44], and IoT tasks offloading in edge-cloud environments [45]. However, those research is not for specific real-world systems. In this study, taking advantage of machine learning with TSC algorithms, we could develop an efficient and lightweight CPM of the fan system with minimal sensors installed. We demonstrated that this model worked smoothly in fog computing.

We obtained two promising results in this work: The first was reducing the high dimensionality of incoming TS from the sensor using the BOSS algorithm. Second, the CPM with the BOSS algorithm acted as a low-pass filter, diminishing the noise significantly. An intelligent system for the operating state of the cooling fan was explained in detail. We demonstrated that such intelligence as CPM works smoothly in fog computing to accurately determine the device's state by analyzing the data transmitted through the IoT device or sensor in real time. Next, a detailed discussion on setting up CPM, closely following Bayes' rule, is presented.

CPM set up with Bayes' rule

In this study, the CPM was sought to classify the fan's state (or class) solely based on the sensor data. The class set $\mathbf{C} = \{C_1, \cdots, C_K\}$ with $K = 3$ classes that are normal, counter-wind, and mechanical failure states of the fan. The CPM can be expressed as a function $f : \mathbf{T} \rightarrow \mathbf{C}$, where $\mathbf{T} = \{T_1, \ldots, T_N\}$, and in probabilistic context, it can be expressed as a joint probability distribution $p(\mathbf{C}, \mathbf{T})$. However, finding the mutual relationship between $\mathbf{C}$ and $\mathbf{T}$ in an actual situation to complete the joint probability is not straightforward. For example, after observing only the TS transmitted from the IoT sensor installed in the fan, it is challenging to determine the state of the fan at once. Therefore, to alleviate this difficulty, we employed a technique of reducing the unknowns using a marginalization based on the Bayes' rule [46], which is handy to describe whole processes here logically. With

the rule, the maximization over a class C on the posterior $p(\mathbf{C}|\mathbf{T})$ is equivalent to the function $f(\mathbf{T})$:

$$\begin{aligned} f(\mathbf{T}) &= \underset{C\in\mathbf{C}}{\operatorname{argmax}}\, p(\mathbf{C}|\mathbf{T}) \\ &= \underset{C\in\mathbf{C}}{\operatorname{argmax}} \frac{p(\mathbf{C},\mathbf{T})}{\int p(\mathbf{T}|\mathbf{C})p(\mathbf{C})d\mathbf{C}} \\ &\propto \underset{C\in\mathbf{C}}{\operatorname{argmax}}\, p(\mathbf{T}|\mathbf{C})p(\mathbf{C}) \end{aligned} \tag{1}$$

The denominator of Eq. (1) is not related to class C, as a result, the CPM function $f(\mathbf{T})$ can be approximated into the multiplication of the likelihood $p(\mathbf{T}|\mathbf{C})$ and the prior $p(\mathbf{C})$. This function is internally composed of histograms for symbols, so it retains the nature of probability, a fascinating result. Because most machine learning models learned from the data are given in the form of linear or nonlinear algebraic equations, not a probability. If the classification results of a CPM are presented as probabilities, we could interpret the result in a probabilistic way; thus, the higher the probability, the more probable the state of the fan becomes. In addition, we compared the classification results for the five different CPMs by performing rigorous statistical tests of accuracy and scalability.

Once the CPM is complete, labeling a new TS data $Q = \{q_1, \cdots, q_m\}$ and $\forall i,\ q_i \in \mathcal{R}^d$ for $i = 1 \ldots m$ with d being the feature can be performed shown in Eq. (2).

$$\begin{aligned} \texttt{label}(\mathbf{Q}) &= \underset{C_k\in\mathbf{C}}{\operatorname{argmax}}\, p(\mathbf{C}|\mathbf{Q}) \\ &= \underset{C_k\in\mathbf{C}}{\operatorname{argmax}}\, p(\mathbf{Q}|\mathbf{C})p(\mathbf{C}) \end{aligned} \tag{2}$$

That is, `label`$(\mathbf{Q})$ is a formula for the classification process. It is noted that $p(\mathbf{C}|\mathbf{Q})$ can be interpreted as the entity with newly arrived data $\mathbf{Q}$ plugged into $p(\mathbf{C}|\mathbf{T})$. Internally, this process is implemented in vector multiplication, and mathematical details of the BOSS algorithm will be explained in the following sections.

An entire procedure of creating a CPM here is schematically diagrammed in Fig. 3, divided into three phases: observation, machine learning, and status update. In the observation phase, streaming data is stored for $3T$ sec. Once it passes $3T$ sec, Machine Learning trains the model for $2T$ sec. It should be noted that even while learning the model, the new data is still coming in to be stored. After the training is finished, a CPM conducts classification over new data for T sec. It is important to note that Machine Learning and Status Update must complete within $3T$ sec; otherwise, this process can be out of work.

Symbolic Fourier Approximation (SFA)

This section introduces Symbolic Fourier Approximation (SFA) used in the BOSS algorithm. SFA consists of Discrete Fourier Transformation (DFT) and Multiple Coefficient Binning (MCB).

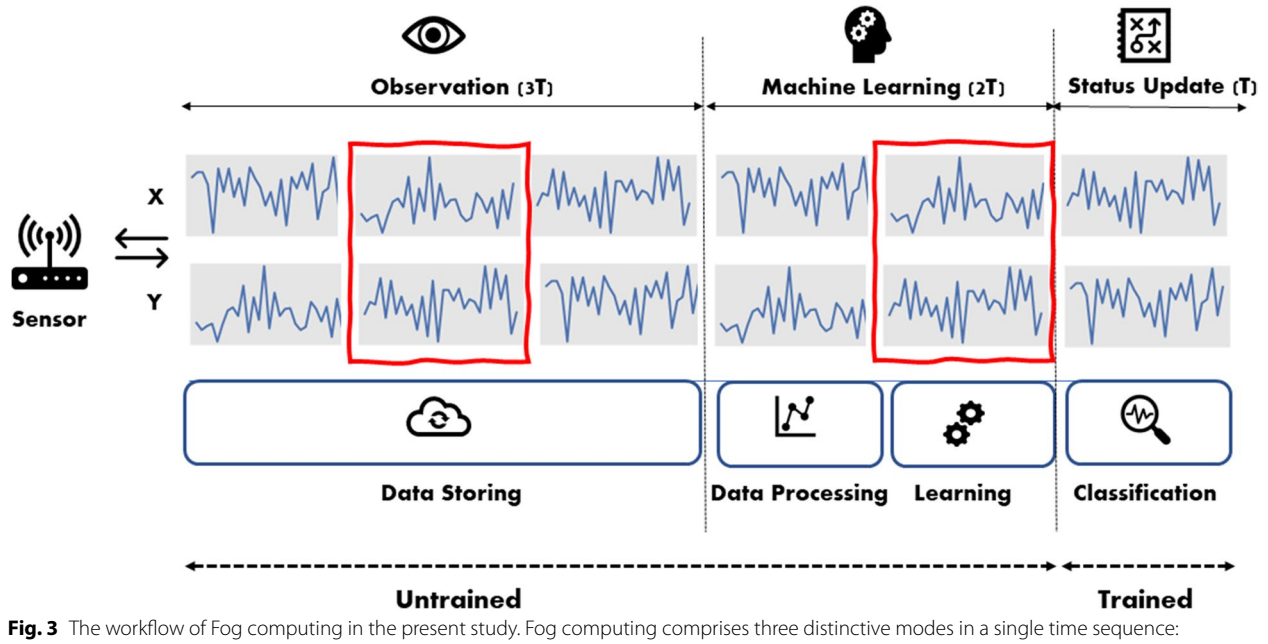

Fig. 3 The workflow of Fog computing in the present study. Fog computing comprises three distinctive modes in a single time sequence: observation, machine learning, and status update. The observation phase executes streaming data storing. The Machine Learning phase does data processing and learning the model. The status update phase carries out a classification of a fan status with the trained model. A red box refers to a sliding window corresponding to a process of the workflow on the timeline. For example, the box which is on the second from the left indicates data storing. Likewise, the fifth box from the left is of learning the model

Discrete Fourier Transformation (DFT)

Discrete Fourier Transform (DFT) extracts Fourier coefficients from each time series T:

$$\texttt{DFT}(T) = \{a_1, b_1, \ldots, a_m, b_m\} \tag{3}$$

where a_i and b_i are the real and the imaginary element of Fourier coefficients. Figure 4 shows that low-pass filtering and smoothing of a sample of acceleration in x-axis upon Discrete Fourier Transform (DFT), where DFT result is obtained by taking first two Fourier coefficients in Eq. (3).

Multiple Coefficient Binning (MCB)

Next, the Multiple Coefficient Binning (MCB) quantization is carried out with training data. $\mathbf{M}$ matrix is constructed using the Fourier transform of N training time series with the first l of Fourier coefficients being equivalent to an SFA word of length l as defined in Eq. (4).

$$\begin{aligned}\mathbf{M} &= \begin{pmatrix} \texttt{DFT}(T_1) \\ \vdots \\ \texttt{DFT}(T_N) \end{pmatrix} \\ &= \begin{pmatrix} (a_1, b_1)_1 & \ldots & (a_{l/2}, b_{l/2})_1 \\ \vdots & \vdots & \vdots \\ (a_1, b_1)_N & \ldots & (a_{l/2}, b_{l/2})_N \end{pmatrix}\end{aligned} \tag{4}$$

where M_j being $j-$th column of $\mathbf{M}$ matrix for all of N training data. M_i is then divided into intervals of c and is sorted by value and then divided into c bins of equi-depth policy. That is, the $i-$th row of $\mathbf{M}$ corresponds to the Fourier transform of the ith time series T_i. With the columns M_j for $j = 1, \ldots, l$, and an alphabet space $\mathcal{A}^l$ of size c, the breakpoints $\beta_j(0) < \cdots < \beta_j(c)$ for each column M_j are generated. Each bin is labeled by applying the ath symbol of the alphabet $\mathcal{A}^l$ to it. For all combination of (j, a) with $j = 1, \ldots, l$ and $a = 1, \ldots, c$, the labeling $symbol(a)$ for M_j can be done by

$$[\beta_j(a-1), \beta_j(a)] \approx symbol(a) \tag{5}$$

It is noted that this process in Eq. (5) applies to all training data.

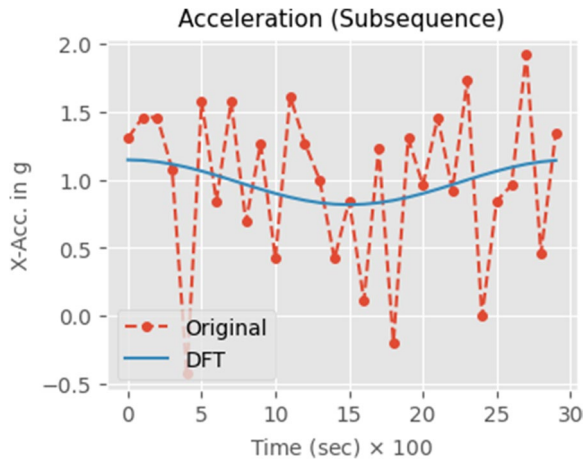

Fig. 4 Low-pass filtering and smoothing of a sample of acceleration in x-axis upon Discrete Fourier Transform (DFT). In this plot, DFT result is obtained by taking only first two Fourier coefficients

SFA working example

SFA word can be obtained from $\texttt{SFA}(T) = s_1, \ldots, s_l$ with DFT where $\texttt{DFT}(T) = t'_1, \ldots, t'_l$ and t's are transformed time series with Fourier transform. That is, $\texttt{SFA}: \mathcal{R}^l \rightarrow \mathcal{A}^l$, where $\mathcal{A}^l$ is the alphabet set of which size is c. For a working example, in Fig. 5, we set $l = 1$ and $c = 2$. Six samples as shown Fig. 5a are randomly selected from the experimental data. The data then is transformed via DFT, resulting in the Fourier coefficients for each sample. A vector of the Fourier coefficient values of the first sample reads (-1.352, 5.043) as shown in Fig. 5b. Next, MCB is conducted with an alphabet set $\mathcal{A}^1 = \{\mathbf{aa}, \mathbf{ab}, \mathbf{ba}, \mathbf{bb}\}$ as shown in Fig. 5c. Thereby, an SFA word of the first sample is mapped into a word **ab** shown in Fig. 5d. Likewise, the other samples can be transformed into their respective SFA words.

Bag-of-SFA-Symbols VS

The Bag-Of-SFA-Symbols VS (BOSS VS) represents the time series representation with the structure-based representation of the bag-of-words model. The sequence of SFA words for six samples in Fig. 5d reads as follows:

$$\mathbf{S} = \{\mathbf{ab}, \mathbf{ba}, \mathbf{bb}, \mathbf{aa}, \mathbf{aa}, \mathbf{bb}\} \tag{6}$$

The values that count the appearance of SFA words in Eq. (6) are expressed upon numerosity reduction:

$$B : \mathbf{aa} = 2,\ \mathbf{ab} = 1,\ \mathbf{ba} = 1,\ \mathbf{bb} = 2 \tag{7}$$

It is noted that SFA words in Eq. (6) now results in the BOSS histogram shown in Eq. (7). Therefore, the BOSS model B can be regarded as a random variable, that is, $B : \mathbf{S} \rightarrow \mathcal{N}$. The probability mass function $p(B)$ can be addressed by $p(B = \mathbf{aa}) = 1/3$, $p(B = \mathbf{ab}) = 1/6$, $p(B = \mathbf{ba}) = 1/6$, and $p(B = \mathbf{bb}) = 1/3$. This provides us with quite important information about the structure of the samples, which structure is being used as features for machine learning.

BOSS VS model is an extended BOSS model. A time series $T = \{t_1, \ldots, t_n\}$ of length n is divided into sliding windows of length of w is $S_{i,w}$, where $w \in \mathcal{N}$. The SFA word is defined as $\texttt{SFA}(S_{i,w}) \in \mathcal{A}^l$, with

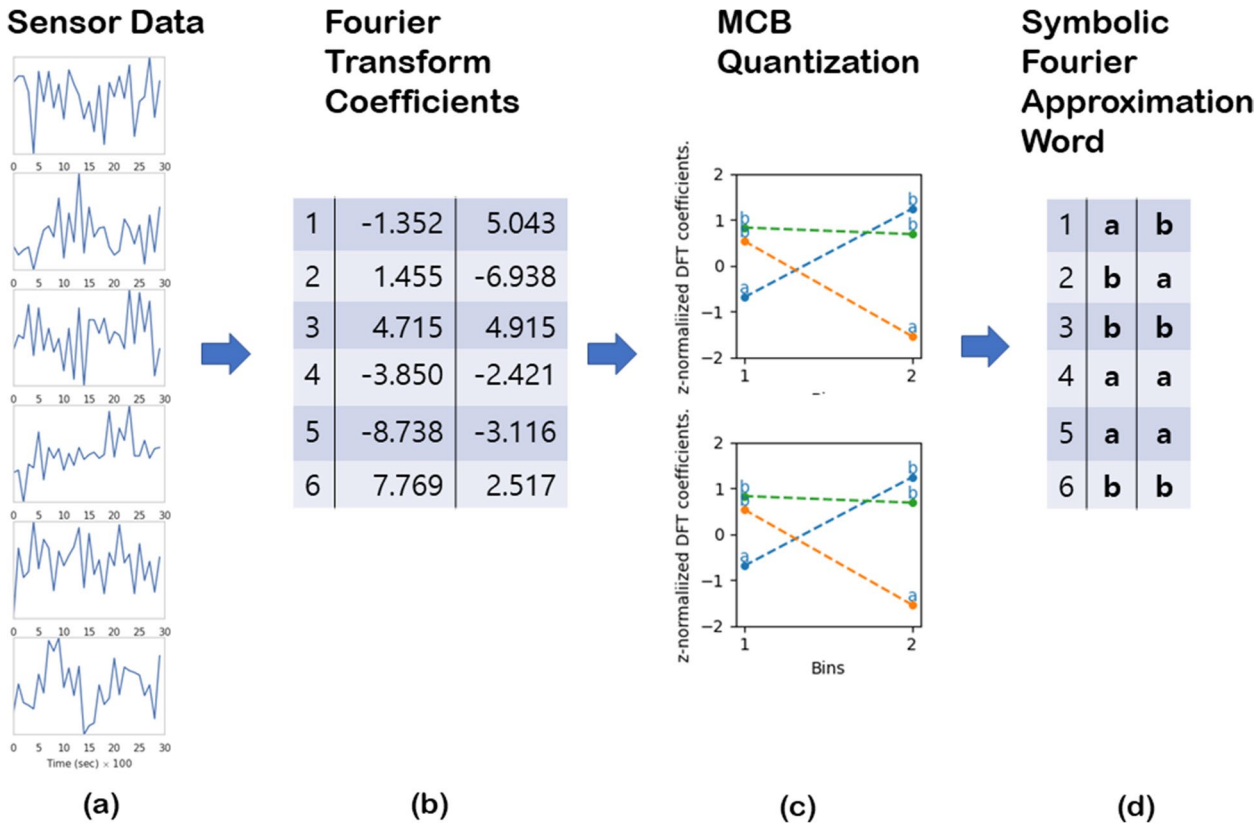

Fig. 5 A pictorial diagram of symbolic Fourier approximation (SFA) procedure: **a** Incoming sensor data of six time-series, **b** The data is then transformed via Fourier transform, **c** The Fourier coefficients are quantized via Multiple Coefficient Binning (MCB), and **d** Each time series has been mapped into its respective SFA word

$i = 1, 2, \ldots, (n - w + 1)$, where $\mathcal{A}$ is the SFA word space and $l \in \mathcal{N}$ is the SFA word length. The BOSS histogram $B(\mathbf{S}) : \mathcal{A}^l \rightarrow \mathcal{N}$. The number in the histogram is the count of appearance of an SFA word within T upon numerosity reduction. BOSS VS model allows frequent updates, such as fast streaming data analytics. As shown in Fig. 6a and b, the BOSS VS model operates sliding windows unto each time series resulting in multiple windowed subsequences. Next, each subsequence is transformed into the SFA words shown in Fig. 6c. All of the subsequences eventually result in the BOSS histogram shown in Fig. 6d. However, since the BOSS histogram itself is not suitable for performing multiple matrix calculations, it is vectorized through Term Frequency Inverse Document Frequency (TF-IDF) algorithm shown in Fig. 6e.

Term frequency inverse document frequency

The BOSS VS model employs Term Frequency Inverse Document Frequency (TF-IDF) algorithm to weight each term frequency in the vector. This assigns a higher weight to signify words of a class. The term frequency `tf` for SFA words $\mathbf{S}$ of a time series T within class $\mathbf{C}$ is defined as

$$\texttt{tf}(\mathbf{S}, \mathbf{C}) = \begin{cases} 1 + \log\left(\sum_{T \in \mathbf{C}} B(\mathbf{S})\right), & \text{if } \sum_{T \in \mathbf{C}} B(\mathbf{S}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where $B(\mathbf{S})$ is the BOSS histogram in Eq. (7). The inverse document frequency `idf` is given by

$$\texttt{idf}(\mathbf{S}, \mathbf{C}) = \log \frac{|\mathbf{C}|}{|\{\mathbf{C} | T \in \mathbf{C} \cap B(\mathbf{S}) > 0\}|} \tag{9}$$

In this study, for classification purposes, we employed three different states of a running fan, which is presented as a set of classes (states) $\mathbf{C}$. The elements of the set are $\mathbf{C} = \{C_1, C_2, C_3\}$. It is noted that each element C_k for $k = 1, 2, 3$ represents a certain state of the fan. As for the human-readable format, we have assigned name-tags to each class such as $C_1 =$ "*Normal*", $C_2 =$ "*Counter Wind*", and $C_3 =$ "*Mechanical Failure*", respectively. The inverse document frequency indicates the frequency of an SFA word in a class C_k. Therefore, in this study, the numerator

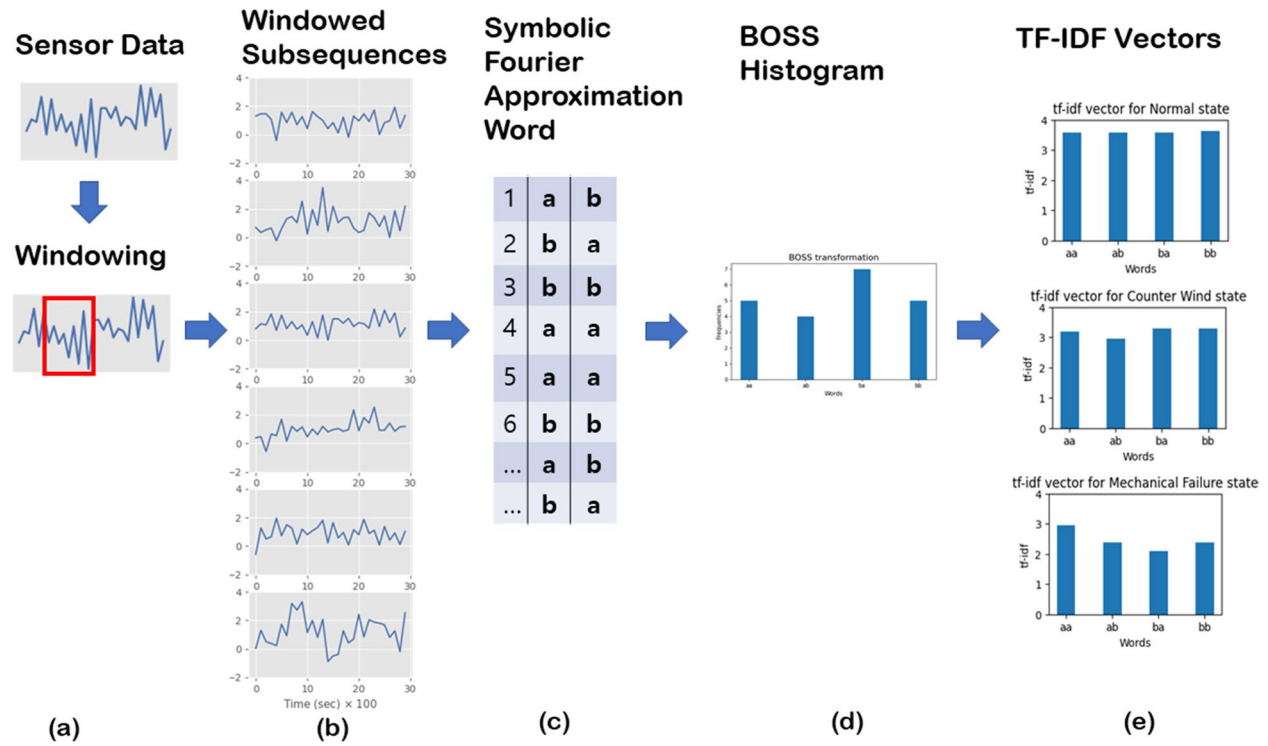

Fig. 6 BOSS model and BOSS VS: **a** Samples are being scanned with a sliding window, **b** multiple windowed subsequences are generated, **c** all of the subsequences are transformed into SFA words, **d** SFA words are summarized in the form of BOSS histogram (BOSS model), and **e** the BOSS histogram is vectorized through Term Frequency Inverse Document Frequency (TF-IDF) model, which finally results in TF-IDF vectors for training data

of Eq. (9) of $|\mathbf{C}|$ denotes a numeric value 3. Multiplying Eq. (8) by Eq. (9), the `tfidf` of an SFA word $\mathbf{S}$ within class $\mathbf{C}$ is defined as

$$\begin{aligned}\texttt{tfidf}(\mathbf{S},\mathbf{C}) &= \texttt{tf}(\mathbf{S},\mathbf{C}) \cdot \texttt{idf}(\mathbf{S},\mathbf{C}) \\ &= \left[1 + \log\left(\sum_{T\in\mathbf{C}} B(\mathbf{S})\right)\right] \cdot \\ &\log \frac{|\mathbf{C}|}{|\{\mathbf{C}|T \in \mathbf{C} \cap B(\mathbf{S}) > 0\}|}\end{aligned} \tag{10}$$

The result of $\texttt{tfidf}(\mathbf{S},\mathbf{C})$ on three states is displayed in Fig. 6e. It is noted that a high weight in Eq. (10) is obtained by a high term frequency in the given class.

Classification

Classification of new data Q can be carried out using the cosine similarity metric `CosSim`:

$$\texttt{CosSim}(\mathbf{Q},\mathbf{C}) = \frac{\sum_{\mathbf{S}\in\mathbf{Q}} \texttt{tf}(\mathbf{S},\mathbf{Q}) \cdot \texttt{tfidf}(\mathbf{S},\mathbf{C})}{\sqrt{\sum_{\mathbf{S}\in\mathbf{Q}} \texttt{tf}^2(\mathbf{S},\mathbf{Q})}\sqrt{\sum_{\mathbf{S}\in\mathbf{C}} \texttt{tfidf}^2(\mathbf{S},\mathbf{C})}} \tag{11}$$

It is noted that in Eq. (11) $\texttt{tf}(\mathbf{S},\mathbf{Q})$ is of the term frequency of $\mathbf{Q}$ as shown in Fig. 7b, which is the BOSS histogram of $\mathbf{Q}$. Then, $\texttt{CosSim}(\mathbf{Q},\mathbf{C})$ is calculated in Eq. (11). Upon maximizing the cosine similarity, a query $\mathbf{Q}$ is thus classified into the class C_k as shown in Eq. (12):

$$\texttt{label}(\mathbf{Q}) = \arg\max_{C_k\in\mathbf{C}}(\texttt{CosSim}(\mathbf{Q}, C_k)) \tag{12}$$

In conclusion, the BOSS VS algorithm comprises two notions: Bag-of-words and TF-IDF. What makes the BOSS VS different from other algorithms is a way of taking features of data. This algorithm does not construct a loss function like other machine learning algorithms but uses Bag-of-Words instead. With time series being transformed into sequences of symbols, Bag-of-words approaches are then used to extract features from these sequences. Time series are presented as histograms with designated symbols. And then, each histogram is transformed into TF-IDF vectors for classification. We have discussed building a model with quite a complexity; thereby, we sorted out the procedure step by step with a lookup table. Table 1 displays the lookup table for the probabilistic models and corresponding algorithms.

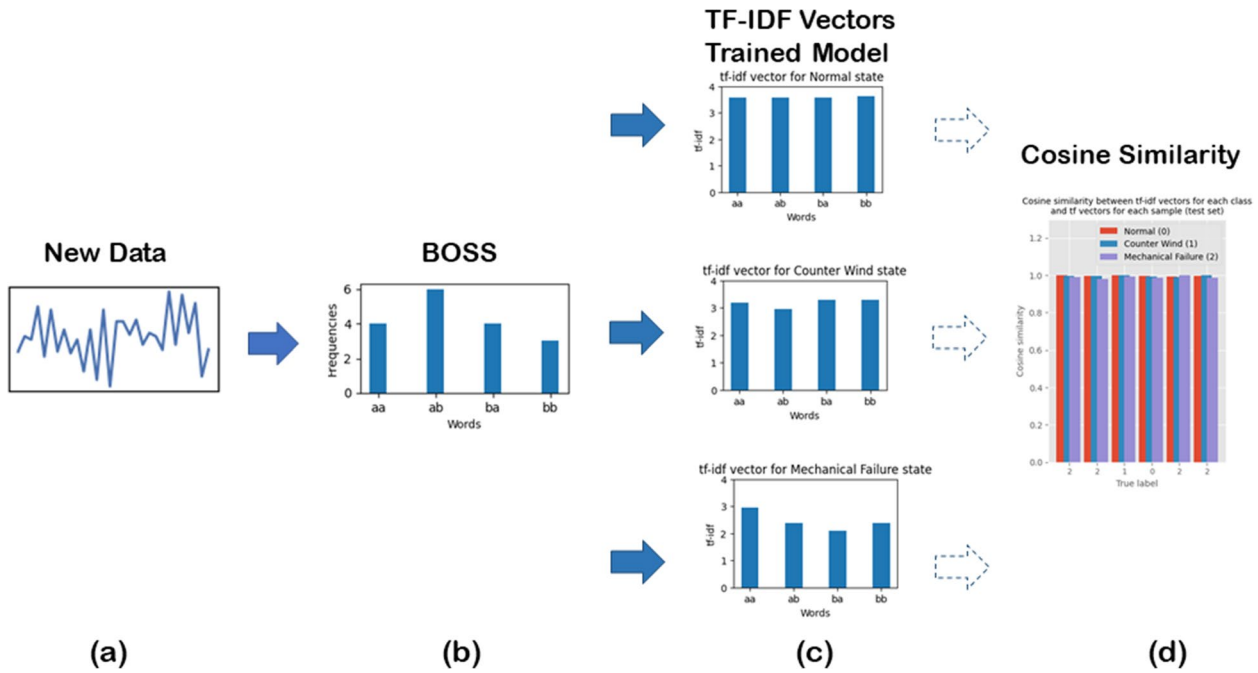

Fig. 7 Schematic diagram of classification with the cosine similarity: **a** New data for query is first transformed into SFA words, **b** the SFA words of the new data is transformed into the BOSS histogram, **c** the trained model in the form of tf-idf algorithm is given, and **d** the classification is carried out through calculating the cosine similarity between the trained model and the query

Results and discussion

Experiments

The experimental apparatus is a three-blade fan on the wheels with a low-power digital accelerometer made in Analog Device (ADXL345 GY-80) as shown in Fig. 8. The dimension of the apparatus the width of 18.5 mm, length of 12.3 mm, and height of 30 mm. We considered three of the most probable states of the fan we can think of in a real-world situation for classification. The normal state where the fan runs without any noticeable event (see left pane in Fig. 8). The counter-wind state occurs when the counter-wind intermittency against the fan (see center pane in Fig. 8). The mechanical failure state where one of the blades is broken off (see right pane in Fig. 8). The average rotational speed of the fan was 114 rpm in the normal state, 41 rpm in the counter-wind state, and 107 rpm in the mechanical-failure state, respectively. Each data set was collected at a sampling rate of 100 Hz for 3 seconds from the accelerometer. For example, we built 2,700 data sets (900 data sets for each fan state). Thus, we made balanced data, and each state has the same amount of data. For example, 900 samples for each state of the fan were collected via x and y channels, so the number of data points sums 900 samples $\times$ 2 channels $\times$ 300 $\times$ 3 states $= 1,620,000$. It took 2 hours and 15 minutes to collect 1,620,000 data points at each measurement. The experiment data sets are shown as a set of time series along with mean and standard deviation into three states in Fig. 9.

For the learning model, we divided the whole data into training and test data with a ratio of 9:1. That is, 270 data sets out of 2,700 were assigned to test data. We performed 10-fold cross-validation over five scenarios, respectively. The cross-validation results are shown in Table 6. For comparison, we listed two different accuracies: non-cross-validation and cross-validation. In scenario II, where 900 data sets are used for machine learning, one of five models, the 1-NN DTW, crashed during the simulations. Such an issue may be caused by limited computing resources in the fog-computing

Table 1 Lookup table for the probabilistic models and corresponding algorithms

Note	Model	Prior	Trained Model	Transformed	Classifier
Probability	$p(\mathbf{T}\|\mathbf{C})$	$p(\mathbf{C})$	$p(\mathbf{C}\|\mathbf{T})$	$p(\mathbf{Q}\|\mathbf{C})$	$p(\mathbf{C}\|\mathbf{Q})$
Algorithm	`tf`$(\mathbf{S},\mathbf{C})$	`idf`$(\mathbf{S},\mathbf{C})$	`tfidf`$(\mathbf{S},\mathbf{C})$	`tf`$(\mathbf{Q},\mathbf{C})$	`CosSim`$(\mathbf{Q},\mathbf{C})$

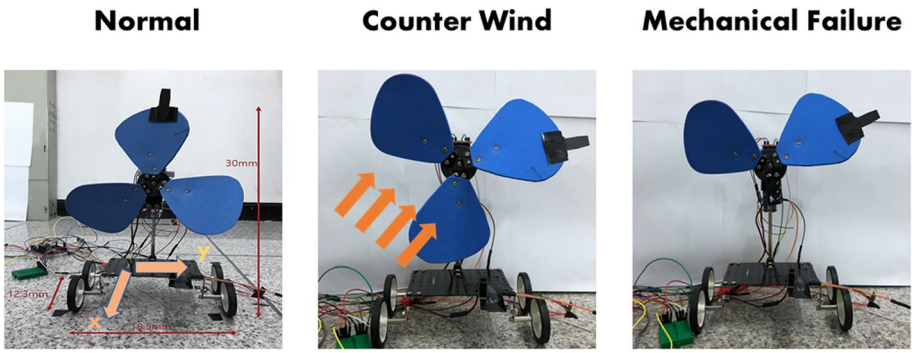

Fig. 8 Photos of the three-blade fan in the three states: Normal state (left), Counter-wind state (center), and Mechanical failure state (right). The counter-wind state indicates the state where the counter-wind being blown by another fan in front of the fan. The mechanical failure refers to the state in which one of the blades having been removed off

facility. Thus, the cross-validation over the 1-NN DTW model could not go forward for larger data sets (listed as N/A in Table 6).

In this study, the operating state of the fan is classified into three categories: normal state, counter-wind state, and mechanical failure state. In Fog computing, CPM is learned by machine learning using data from acceleration sensors and five different TSCs and analyzing the newly input data to distinguish the state of the fan. Fog computing in this study was Intel Core i3-8100 CPU 3.60 GHz at 4 GB memory on Ubuntu 20.04.1 LTS.

Exploratory data analysis

Exploratory Data Analysis (EDA) is carried out to identify data property. In Fig. 9, the raw data, its rolling mean, and the standard deviation are overlaid. Since raw data contains much noise, it is necessary to filter it out for a better analysis. The rolling mean is one such filtering tool. The standard deviation can be used for estimating the variance of data. In addition, we need to know how many trends, repetition over intervals, white noise, and other uncertainty are. We should never take these data characteristics lightly because they can determine the authenticity of the experiment. We employed the Augmented Dickey-Fuller (ADF) test with the null hypothesis of whether it was stationary. The test results of p-value ≤ 0.05 as shown in Table 2 for the three states with time series from experimental data; therefore, we can reject the null hypothesis at a 95% confidence level. Thus, we may affirm that the experimental data was stationary.

Comparison of models

We employed five different models: WEASEL MUSE, BOSS VS, random forest (RF), logistic regression (LR), and one-nearest neighbor DTW (1-NN DTW). Table 3 describes the characteristics of five models according to temporal structure (Temporal), low-pass filtering (Filter), transformation (Transform), and critical features (Features). Only 1-NN DTW keeps the temporal structure of data, and the others do not consider the order of data points over time. Algorithms for feature extraction are χ^2 test for WEASEL MUSE, Term-Frequency Inverse Document Frequency algorithm for BOSS VS, Entropy for RF, the cost function for LR, and Euclidean distance for 1-NN DTW.

Classification with BOSS VS

Table 4 shows the numerical expression of the trained model $p(\mathbf{C}|\mathbf{T})$ in Table 1, which is the result of vector $\texttt{tfidf}(\mathbf{S}, \mathbf{C})$ calculated using training data. The symbolic algorithm SFA converts the whole training data to $\mathbf{S} = \{\mathbf{aa}, \mathbf{ab}, \mathbf{ba}, \mathbf{bb}\}$. For example, the features of the normal state (C_1) are **aa**, **ab**, **ba**, and **bb** with the numerical values (3.5649, 3.5649, 3.5649, 3.6390) as displayed in Table 4. For the counter-wind state (C_2), the value reads (3.1972, 2.9459, 3.3025, 3.3025), which is clearly distinguished from those of the normal state.

Table 5 shows the classifier $p(\mathbf{C}|\mathbf{Q})$ in Table 1 for $\mathbf{Q}$. For example, the first sample Q_1 is predicted as the normal state because of the largest value of 0.9990 throughout the column to which it belongs. In the same fashion, the classification is performed for the remaining time series, such as the counter-wind state for Q_2, the mechanical failure state for Q_5, etc.

Post-Hoc analysis

Often in many studies, the results tend to be presented without statistical rigor. However, it is essential to check if it was statistically significant before further discussion, called Post-Hoc analysis.

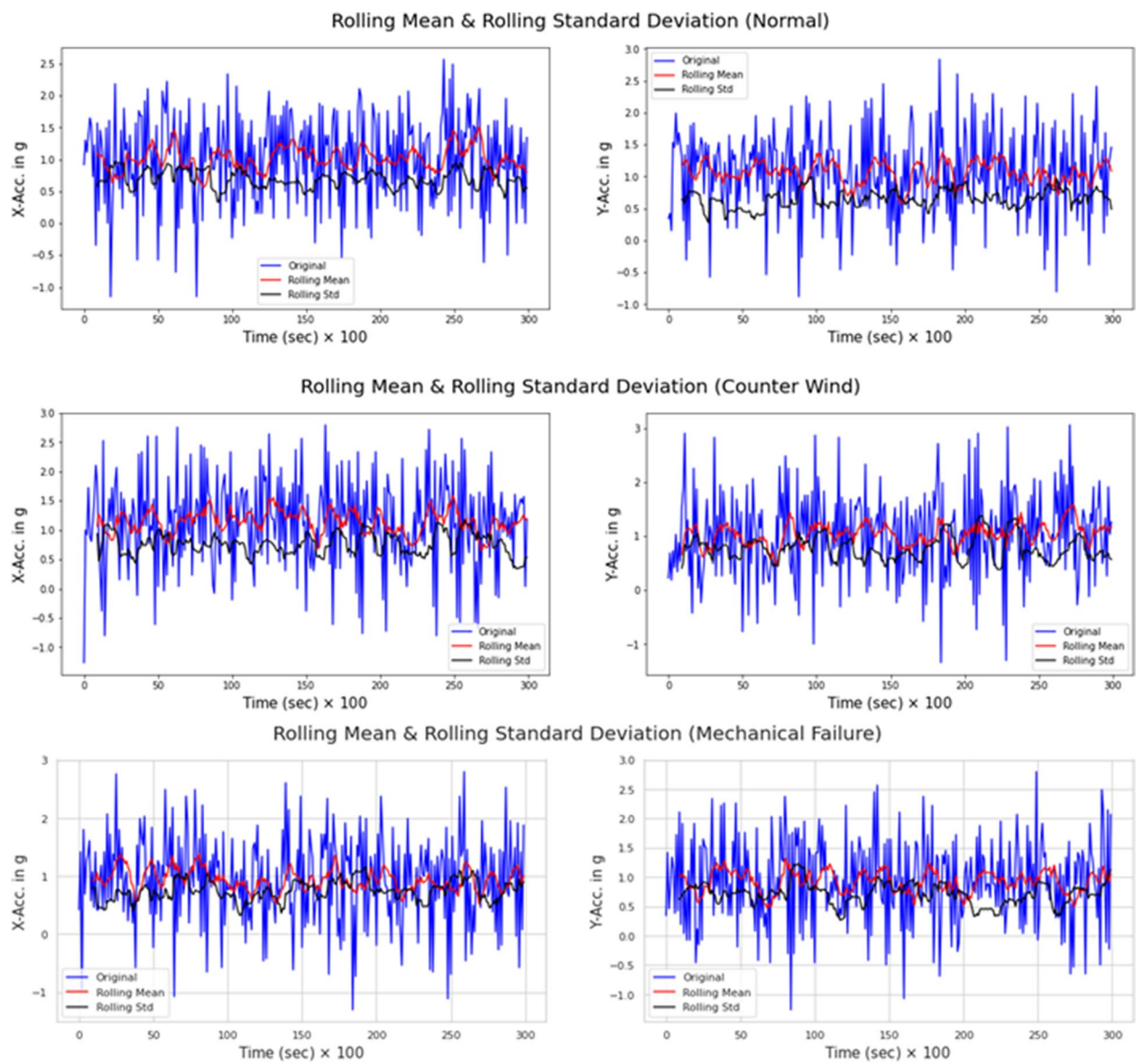

Fig. 9 Experimental time series data for three states of the fan: Normal state (top row), counter-wind state (middle row), and mechanical failure state (bottom row). Raw data from the accelerometer overlaid with the rolling mean and standard deviation. Each row represents both *x* (left) and *y* (right) acceleration in *g* unit

As shown in Table 6, the BOSS VS model indicates the highest accuracy for all data sizes. In addition, even though the number of data is increased from 180 k to 1.6 million, it shows a bit change in accuracy, so we may conclude that the BOSS VS model is not significantly affected by the data size. The smaller the number of data used, the shorter the run time, but on the other hand, the model tends to be overfitted to the data. For example, in scenario I, we used 180,000 data points; for the BOSS VS model, the accuracy turned out to be 100%. This result indicates overfitting, where we used too little data for training. If the number of data is increased, the time for preprocessing and calculation also increases accordingly. Machine learning models tend to suffer from overfitting one way or another for several reasons. The reviewer correctly pointed out that using a proper validation method is essential. Furthermore, the most apparent reason might be lacking data. In machine learning communities, it is well known that too little data for simulation does not bring significant results. In this study, we experienced overfitting when only 300 data sets were used for simulation. As we used more data, such overfitting does not

Table 2 Augmented Dickey-Fuller (ADF) Test Results: The results show that all of six time-series are found to be stationary of being statistically significant owing to p-value ≤ 0.05 (95% confidence level), which indicates evidence against the null hypothesis H_0

Time Series	ADF statistic	*P*-value	Critical Value (5%)
x-acc (Normal)	-4.629	1.1×10^{-4}	-2.871
y-acc (Normal)	-6.137	8.1×10^{-8}	-2.871
x-acc (Counter Wind)	-6.486	1.2×10^{-8}	-2.871
y-acc (Counter Wind)	-5.839	3.8×10^{-7}	-2.871
x-acc (Mechanical Failure)	-4.577	1.4×10^{-4}	-2.871
y-acc (Mechanical Failure)	-4.459	2.3×10^{-4}	-2.871

Table 3 Comparison of characteristics of five models via conducting normalization (Norm.), keeping temporal structure (Temporal), carrying out low-pass filtering (Filter), executing transformation (Transform), and key features (Features)

	Temporal	Filter	Transform	Features
WEASEL MUSE	No	Yes	Yes	χ^2 test
BOSS VS	No	Yes	Yes	Term Frequency
RF	No	No	No	Entropy
LR	No	No	No	Cost
1-NN DTW	Yes	No	No	Distance

Table 4 The trained model $p(\mathbf{C}|\mathbf{T})$ with equivalent of TF-IDF vector $\texttt{tfidf}(\mathbf{S}, \mathbf{C})$ for the training data. $\mathbf{S} = \{\mathbf{aa}, \mathbf{ab}, \mathbf{ba}, \mathbf{bb}\}$ is the SFA words and $\mathbf{C} = \{C_1, C_2, C_3\}$ is three states of the fan

Class	C_1	C_2	C_3
aa	3.5649	3.1972	2.9459
ab	3.5649	2.9459	2.3862
ba	3.5649	3.3025	2.0986
bb	3.6390	3.3025	2.3862

appear anymore. Therefore, more data can be a cure for the issue in this particular case.

Table 6 does not tell whether the difference in the accuracy of each model is statistically meaningful. Another ambiguity arises in the results of the run time. Thus, we carried out the ANOVA (Analysis of Variance) test that provides statistical significance among differences. Table 7 shows the ANOVA test result for accuracy with $F = 60.8$, and p-value ≤ 0.05 at a 95% confidence level. This result indicates that the null hypothesis is rejected. Therefore, we can say that the mean values for the accuracy of each model differ significantly. In addition, the difference in run time for each model is statistically significant, with $F = 4.58$, and p-value $= 0.008$. In conclusion, we can confirm the simulation over five scenarios for accuracy and run time with five statistically significant models.

However, the ANOVA test results shown in Table 7 alone cannot tell which model is different in accuracy and run time from others. Thereby, another test should be carried out to see which model is significantly different from the others. We employed Tukey's Honest Significant Difference test (Tukey Test) for all pairwise comparisons while controlling multiple comparisons. In this study, the suitability of models was sought statistically in two aspects: accuracy and scalability.

Accuracy

The Tukey test result, which is multiple comparisons of accuracy from five models, is summarized in Table 8. Two cases, 1-NN DTW vs. WEASEL MUSE and LR vs. RF, are not statistically significant upon a 95% confidence level. This result implies that two pairs may have a remarkable similarity in making poor predictions. On the contrary, all pairwise comparisons with the BOSS VS model are proven statistically significant at a 95% confidence level. Figure 10 shows yet another aspect of the trend of accuracy over run time for all five models, where the BOSS VS model outputs a far outstanding performance in accuracy and run time. As a result,

Table 5 The classifier $p(\mathbf{C}|\mathbf{Q})$ with equivalent to Cosine similarity between the trained model $p(\mathbf{C}|\mathbf{T})$ for each class and new samples $\mathbf{Q} = \{Q_1, \ldots, Q_6\}$ as a query. $\mathbf{C} = \{C_1, C_2, C_3\}$ is three states of the fan. The similarity results in the prediction for the new samples. The maximum value of the cosine similarity for each sample is boldfaced

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
Normal	**0.9990**	0.9958	0.9987	**0.9963**	0.9943	0.9970
Counter Wind	0.9964	**0.9977**	**0.9988**	0.9942	0.9909	**0.9991**
Mechanical Failure	0.9908	0.9791	0.9924	0.9855	**0.9985**	0.9868

Table 6 Comparison of performance of five models according to five scenarios: The number of data points is increased from 180k up to 1.6 Million. As for Fog computing facility: Intel Core i3-8100 CPU 3.60GHz at 4GB memory, and the OS is Ubuntu 20.04.1 LTS. BOSS VS model shows excellent scalability while keeping the highest accuracy among the other models. N/A denotes that simulation data is unavailable due to computation crashes caused by hardware limitations in this Fog computing facility. The 10-fold cross-validation accuracy is shown in the 95% confidence level

Scenario	Data points (data sets)	Model	Time (sec)	Accuracy (non CV)	Cross Validation Accuracy
I	180,000 (300)	WEASEL MUSE	0.39	0.20	0.46 ± 0.17
		BOSS VS	0.46	1.00	0.98 ± 0.04
		RF	0.44	0.56	0.70 ± 0.12
		LR	0.25	0.63	0.57 ± 0.14
		1-NN DTW	10.40	0.26	0.35 ± 0.06
II	540,000 (900)	WEASEL MUSE	1.00	0.33	0.49 ± 0.08
		BOSS VS	1.42	0.97	0.98 ± 0.02
		RF	1.16	0.82	0.85 ± 0.06
		LR	2,700	0.65	0.64 ± 0.06
		1-NN DTW	89.54	0.37	N/A
III	900,000 (1,500)	WEASEL MUSE	1.810	0.34	0.56 ± 0.08
		BOSS VS	2.456	0.98	0.98 ± 0.01
		RF	1.961	0.90	0.89 ± 0.06
		LR	7.153	0.74	0.70 ± 0.09
		1-NN DTW	246.794	0.40	N/A
IV	1,080,000 (1,800)	WEASEL MUSE	2.193	0.36	0.54 ± 0.06
		BOSS VS	2.966	0.98	0.98 ± 0.01
		RF	2.472	0.87	0.89 ± 0.03
		LR	12.921	0.76	0.70 ± 0.05
		1-NN DTW	352.067	0.41	N/A
V	1,620,000 (2,700)	WEASEL MUSE	3.851	0.37	0.55 ± 0.03
		BOSS VS	8,100	0.98	0.98 ± 0.01
		RF	3.910	0.92	0.91 ± 0.02
		LR	32.183	0.71	0.73 ± 0.02
		1-NN DTW	793.22	0.38	N/A

Table 7 ANOVA test result of accuracy and run time among five models

	sum_{sq}	df	F	PR(>F)
Algorithms (Accuracy)	1.64	4.0	60.80	6.56×10^{-11}
Residual	0.13	20.0	-	-
Algorithms (Run time)	346268.91	4.0	4.58	0.008631
Residual	377628.33	20.0	-	-

98% accuracy was obtained at a 95% confidence level by adopting the model with the BOSS VS algorithm.

Scalability

In general, a scalable model shows consistently good performance despite increasing the amount of data. Multiple comparisons of run time for five models are summarized in Table 9. All pairwise cases for the 1-NN DTW model vs. the other models are significant. Thus, we may conclude that the 1-NN DTW model is far less scalable. Figure 10 shows the scalability of models. Except for the 1-NN DTW model, the other models keep relatively small changes in the run-time subject to increasing data size. In Fig. 11, the scalability comparison of five models is shown. As the amount of data increases, the 1-NN-DTW model shows the worst scalability. We observed that the BOSS VS model performs excellent scalability with the best accuracy among the other models. Figure 12 shows the comparison of the 95% confidence interval (CI) of the accuracy of each model using experimental data of different sizes. The accuracy of the BOSS VS model fell into $CI = 0.9872 \pm 0.0073$, of which statistical behavior is much better compared to $CI = 0.8205 \pm 0.1319$ in the second-place RF model. Moreover, the deviation of 0.0073 of the BOSS VS model is quite small compared to 0.1319 of the RF model. This

Table 8 Multiple Comparison of Means of Accuracy - Tukey HSD test

Group1	Group2	meandiff	p-adj	lower	upper	reject
1-NN DTW	BOSS VS	0.614	0.001	0.458	0.770	True
1-NN DTW	LR	0.328	0.001	0.172	0.484	True
1-NN DTW	RF	0.447	0.001	0.292	0.603	True
1-NN DTW	WEASEL MUSE	-0.049	0.862	-0.205	0.106	False
BOSS VS	LR	-0.286	0.001	-0.441	-0.130	True
BOSS VS	RF	-0.166	0.032	-0.322	-0.011	True
BOSS VS	WEASEL MUSE	-0.663	0.001	-0.819	-0.508	True
LR	RF	0.119	0.187	-0.036	0.274	False
LR	WEASEL MUSE	-0.377	0.001	-0.533	-0.222	True
RF	WEASEL MUSE	-0.497	0.001	-0.652	-0.341	True

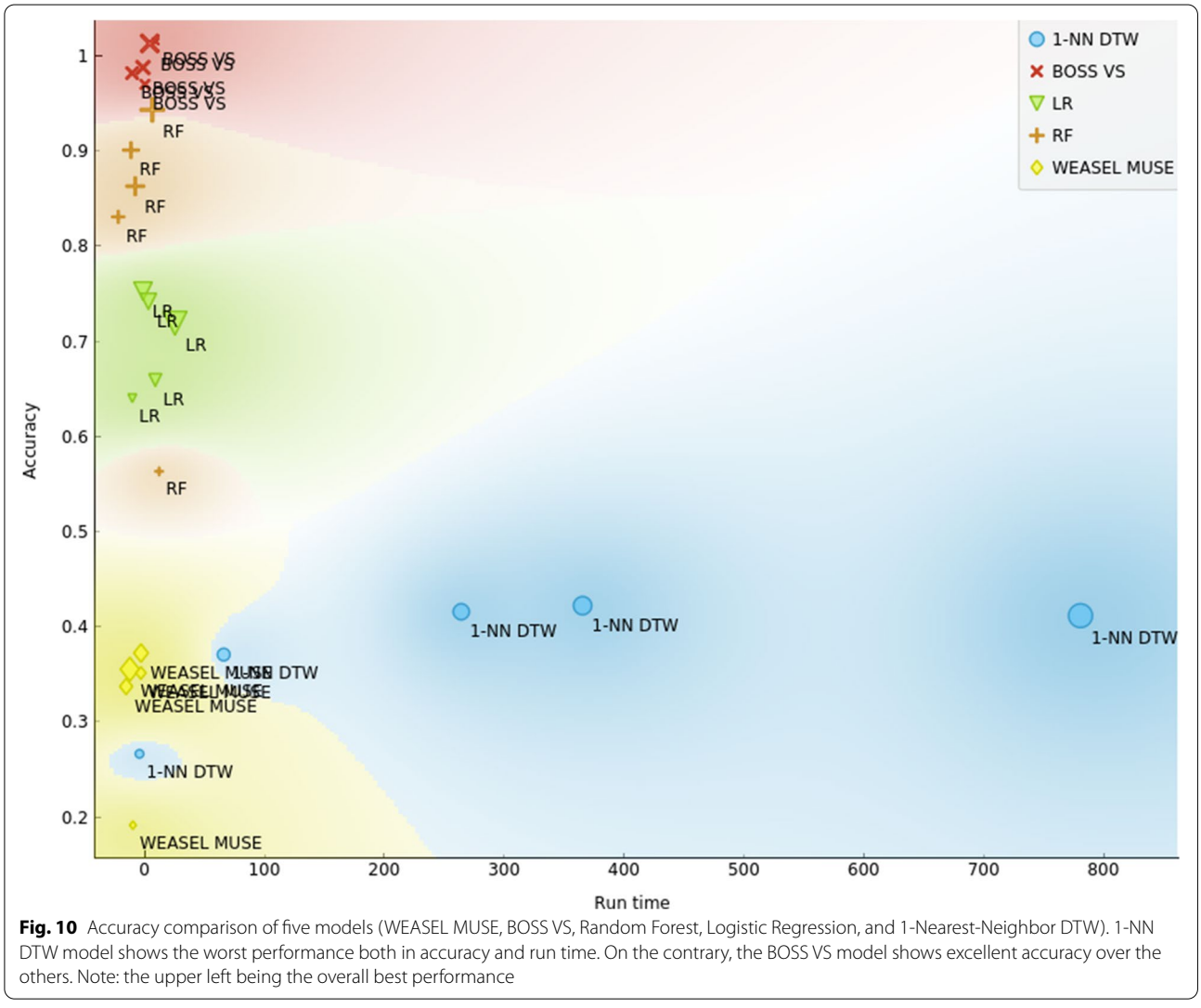

Fig. 10 Accuracy comparison of five models (WEASEL MUSE, BOSS VS, Random Forest, Logistic Regression, and 1-Nearest-Neighbor DTW). 1-NN DTW model shows the worst performance both in accuracy and run time. On the contrary, the BOSS VS model shows excellent accuracy over the others. Note: the upper left being the overall best performance

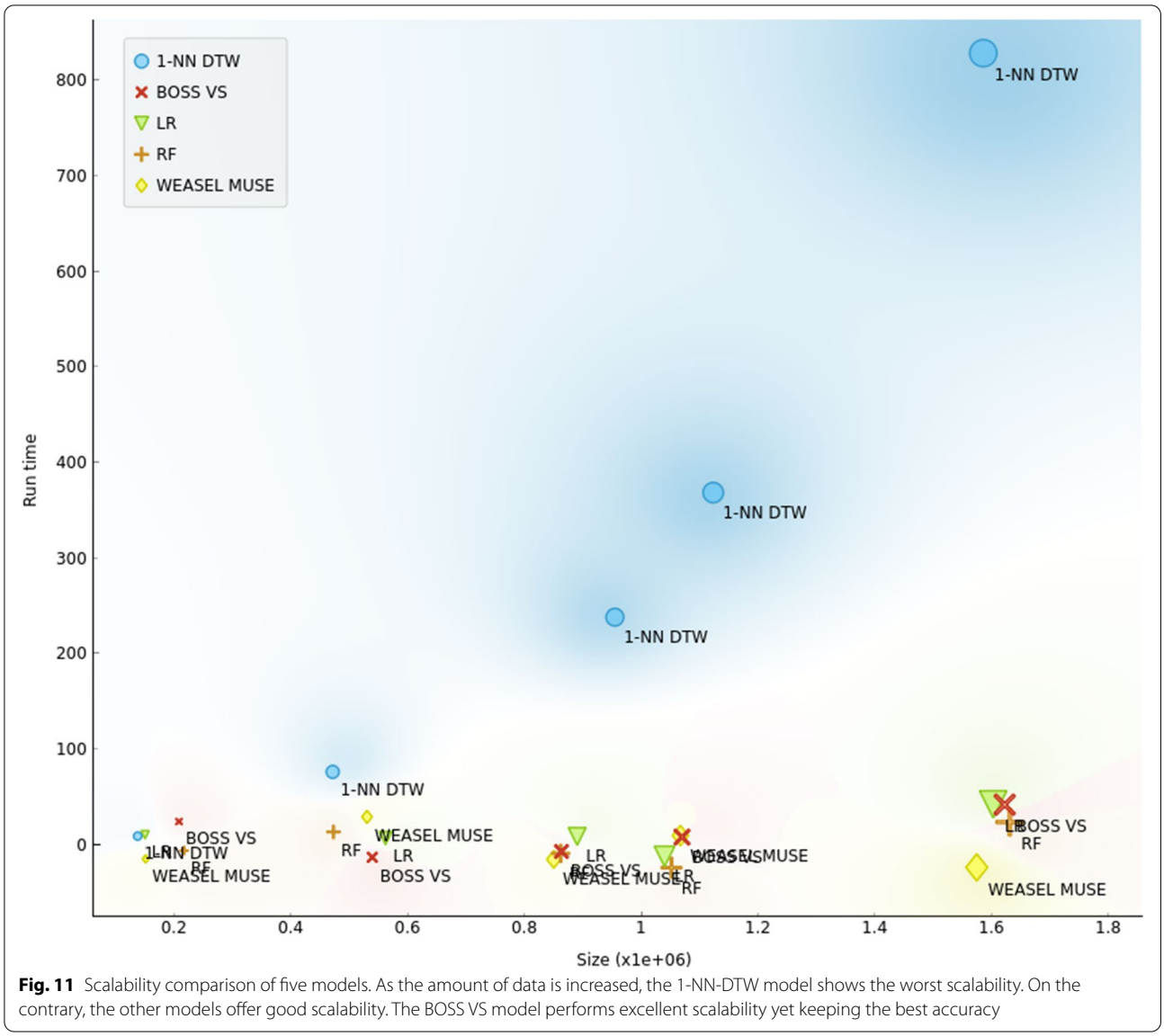

Fig. 11 Scalability comparison of five models. As the amount of data is increased, the 1-NN-DTW model shows the worst scalability. On the contrary, the other models offer good scalability. The BOSS VS model performs excellent scalability yet keeping the best accuracy

Table 9 Multiple Comparison of Means of run time - Tukey HSD

Group1	Group2	meandiff	p-adj	lower	upper	reject
1-NN DTW	BOSS VS	-296.01	0.020	-556.08	-35.93	True
1-NN DTW	LR	-287.39	0.025	-547.47	-27.32	True
1-NN DTW	RF	-296.41	0.020	-556.49	-36.34	True
1-NN DTW	WEASEL MUSE	-296.55	0.020	-556.62	-36.48	True
BOSS VS	LR	8.61	0.900	-251.45	268.68	False
BOSS VS	RF	-0.405	0.900	-260.47	259.66	False
BOSS VS	WEASEL MUSE	-0.542	0.900	-260.61	259.53	False
LR	RF	-9.02	0.900	-269.09	251.05	False
LR	WEASEL MUSE	-9.15	0.900	-269.23	250.91	False
RF	WEASEL MUSE	-0.137	0.900	-260.21	259.93	False

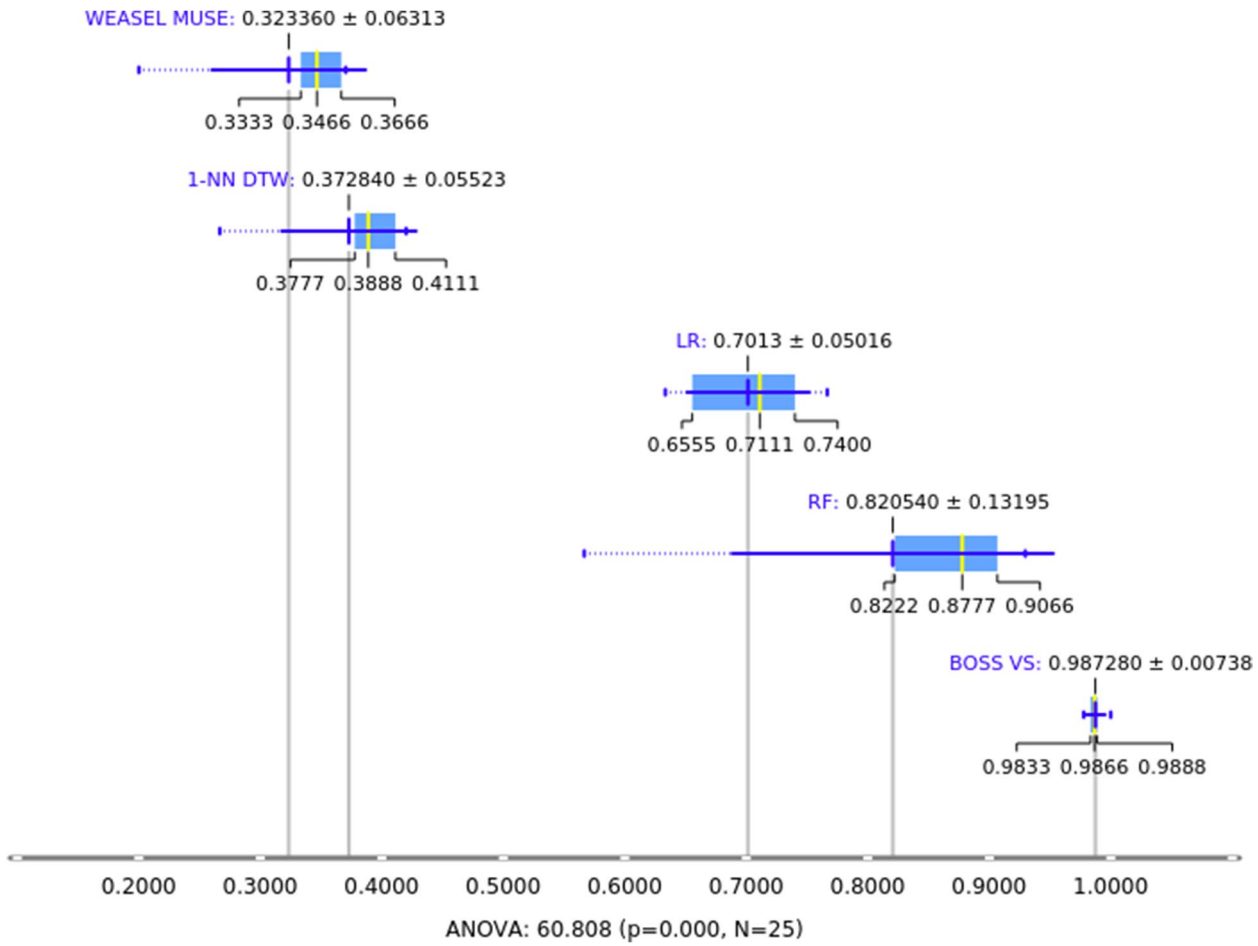

Fig. 12 The result of comparing the 95% confidence interval (CI) of the accuracy of five models using five scenarios of data size. This illustrates the scalability of each model's performance in classification. The accuracy of the BOSS VS model fell into CI = 0.9872 ± 0.0073 resulting in the best performance

result explains good scalability, which indicates that the BOSS VS model is robust to changes in data size.

Conclusion

While most literature on the subject was published using the results using well-preprocessed public data, in this work, we implemented noisy real-world data into the classification models.

A primary goal of this study is to build an excellent cyber-physical model (CPM) of an IoT device with significant classification accuracy in fog computing. To this end, some critical issues must be resolved: one is of time series classification (TSC) algorithms and domain-specific knowledge that can identify the state of an IoT device, and the other is about how to process streaming sensor data in real-time properly. A data-driven modeling approach could alienate quite the burden of such complicated theoretical domain-specific knowledge by using a vast amount of data to take advantage of machine learning and statistical inference. With a large amount of data transmitted in real-time from a networked IoT device, it is significant to correctly classify the device's state, a core of smart manufacturing. Recently, there has been a growing tendency to solve this issue in fog computing close to IoT devices because of the heavy loading on cloud computing.

We studied a three-blade fan with an accelerometer installed for an IoT device to create CPMs that can classify the state of the fan. Using state-of-the-art TSC algorithms, upon the classification performance of five CPMs with real-world data, we achieved an accuracy of about 98% at a 95% confidence level with the BOSS VS algorithm, which resulted in excellent classification and significant size reduction in streaming data.

In this work, we produced a lightweight CPM that works well in fog computing, which can determine the state of the fan, but, in the field, since there are a large number of fans installed, creating models for each fan will

be costly. In addition, the status model for fans will not be the same because the type and environment of installed fans are different. Therefore, further study on techniques that can economically create models for various fans should be a priority. As the state of the cooling fan can be intelligently determined at the fog computing level, it will ease the load that cloud computing has to bear significantly. In addition, we believe that the development of computer hardware is evolving very swiftly and that later research on intelligent models that work on-device levels can also be essential. Thus, further studies should be conducted for efficient models and algorithms against ever-increasing sensors in smart manufacturing. Therefore, it is necessary to expand this study to explore techniques applicable even in larger manufacturing environments.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13677-022-00337-y.

Additional file 1: Supplementary Materials.

Authors' contributions

Conceptualization, Experiment Setup, Experimentation, Data Collection, Analysis, Programming, Visualization, Writing, and Editing has been done by Deok-Kee Choi, Ph.D. The author has read and agreed to the published version of the manuscript.

Authors' information

The author is a professor of the department of mechanical engineering at Dankook University, Korea. His area of research is the application of machine learning into smart manufacturing and application in various computing environments such as cloud computing, fog computing, and on-device computing.

Funding

Not applicable.

Availability of data and materials

Experimental data file is attached.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 1 April 2021 Accepted: 28 September 2022
Published online: 05 October 2022

References

1. Oks SJ, Jalowski M, Fritzsche A, Möslein KM (2019) Cyber-physical modeling and simulation: A reference architecture for designing demonstrators for industrial cyber-physical systems. Procedia CIRP 84:257–264
2. Schroeder B, Gibson GA (2007) Understanding disk failure rates: What does an mttf of 1,000,000 hours mean to you? ACM Transactions on Storage (TOS) 3(3):8–es
3. Diaz CP, Postol M, Simon R (2019) Time-series data analysis for classification of noisy and incomplete internet-of-things datasets. Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States)
4. Shifaz A, Pelletier C, Petitjean F, Webb GI (2020) Ts-chief: a scalable and accurate forest algorithm for time series classification. Data Min Knowl Discov 34(3):742–775
5. Schäfer P (2015) The boss is concerned with time series classification in the presence of noise. Data Min Knowl Discov 29(6):1505–1530
6. Georgoulas G, Karvelis P, Loutas T, Stylios CD (2015) Rolling element bearings diagnostics using the symbolic aggregate approximation. Mech Syst Signal Process 60:229–242
7. Shin I, Lee J, Lee JY, Jung K, Kwon D, Youn BD, Jang HS, Choi JH (2018) A framework for prognostics and health management applications toward smart manufacturing systems. Int J Precis Eng Manuf-Green Technol 5(4):535–554
8. Lee CM, Park J, Park S, Kim CH (2020) Fall-detection algorithm using plantar pressure and acceleration data. Int J Precis Eng Manuf 21(4):725-737
9. Koo B, Kim J, Kim T, Jung H, Nam Y, Kim Y (2020) Post-fall detection using ann based on ranking algorithms. Int J Precis Eng Manuf 21(10):1985-1995
10. Patel P, Ali MI, Sheth A (2017) On using the intelligent edge for iot analytics. IEEE Intell Syst 32(5):64–69
11. Qi Q, Tao F (2019) A smart manufacturing service system based on edge computing, fog computing, and cloud computing. IEEE Access 7:86769–86777
12. Xu Z, Zhang Y, Li H, Yang W, Qi Q (2020) Dynamic resource provisioning for cyber-physical systems in cloud-fog-edge computing. J Cloud Comput 9(1):1–16
13. Hu Y, Wang H, Ma W (2020) Intelligent cloud workflow management and scheduling method for big data applications. J Cloud Comput 9(1):1–13
14. Chen S, Zhang T, Shi W (2017) Fog computing. IEEE Int Comput 21(2):4–6
15. Peralta G, Iglesias-Urkia M, Barcelo M, Gomez R, Moran A, Bilbao J (2017) Fog computing based efficient iot scheme for the industry 4.0. In: 2017 IEEE international workshop of electronics, control, measurement, signals and their application to mechatronics (ECMSM). New York: Institute of Electrical and Electronics Engineers; p 1–6
16. Qi Q, Zhao D, Liao TW, Tao F (2018) Modeling of cyber-physical systems and digital twin based on edge computing, fog computing and cloud computing towards smart manufacturing. In: ASME 2018 13th International Manufacturing Science and Engineering Conference, American Society of Mechanical Engineers, New York NY 10016-5990
17. Esling P, Agon C (2012) Time-series data mining. ACM Comput Surv (CSUR) 45(1):1–34
18. Schäfer P (2016) Scalable time series classification. Data Min Knowl Discov 30(5):1273–1298
19. Salvador S, Chan P (2007) Toward accurate dynamic time warping in linear time and space. Intell Data Anal 11(5):561–580
20. Wagner N, Antoine V, Koko J, Lardy R (2020) Fuzzy k-nn based classifiers for time series with soft labels. In: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. New York: Springer; p 578–589
21. Wang Z, Yan W, Oates T (2017) Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International joint conference on neural networks (IJCNN). New York: Institute of Electrical and Electronics Engineers; p 1578–1585
22. Karim F, Majumdar S, Darabi H, Chen S (2017) Lstm fully convolutional networks for time series classification. IEEE Access 6:1662–1669
23. Montero Quispe KG, Sousa Lima W, Macêdo Batista D, Souto E (2018) Mboss: A symbolic representation of human activity recognition using mobile sensors. Sensors 18(12):4354
24. Hatami N, Gavet Y, Debayle J (2019) Bag of recurrence patterns representation for time-series classification. Patt Anal Appl 22(3):877–887
25. Senin P, Malinchik S (2013) Sax-vsm: Interpretable time series classification using sax and vector space model. In: 2013 IEEE 13th international conference on data mining. New York: Institute of Electrical and Electronics Engineers; p 1175–1180
26. Baldini G, Giuliani R, Steri G, Sanchez I, Gentile C (2017) The application of the symbolic aggregate approximation algorithm (sax) to radio frequency fingerprinting of iot devices. In: 2017 IEEE Symposium on Communications and Vehicular Technology (SCVT). New York: Institute of Electrical and Electronics Engineers; pp 1–6

27. Schäfer P, Leser U (2017) Fast and accurate time series classification with weasel. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. New York: Association for Computing Machinery; p 637–646
28. Large J, Bagnall A, Malinowski S, Tavenard RR (2019) On time series classification with dictionary-based classifiers. Intelligent Data Analysis 23(5):1073-1089
29. Schäfer P, Högqvist M (2012) Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In: Proceedings of the 15th international conference on extending database technology. New York: Association for Computing Machinery; p 516–527
30. Karimi-Bidhendi S, Munshi F, Munshi A (2018) Scalable classification of univariate and multivariate time series. In: 2018 IEEE International Conference on Big Data (Big Data). New York: Institute of Electrical and Electronics Engineers; pp 1598–1605
31. Sun Y, Li J, Liu J, Sun B, Chow C (2014) An improvement of symbolic aggregate approximation distance measure for time series. Neurocomputing 138:189–198
32. Fawaz HI, Lucas B, Forestier G, Pelletier C, Schmidt DF, Weber J, Webb GI, Idoumghar L, Muller PA, Petitjean F (2020) Inceptiontime: Finding alexnet for time series classification. Data Min Knowl Discov 34(6):1936–1962
33. Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015) The ucr time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. Accessed 29 Sept 2022.
34. Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min Knowl Discov 31(3):606–660
35. Reister H, Ross F (1997) Numerical simulation of an axial cooling fan. Tech. rep, SAE Technical Paper
36. Coggiola E, Dessale B, Moreau S, Broberg R, Bakir F (1998) Cfd based design for automotive engine cooling fan systems. Tech. rep, SAE Technical Paper
37. Sanjose M, Moreau S (2012) Numerical simulations of a low-speed radial fan. Int J Eng Syst Model Simul 4(1–2):47–58
38. Jian-Hui Z, Chun-Xin Y (2008) Design and simulation of the cpu fan and heat sinks. IEEE Trans Components Packag Technol 31(4):890–903
39. Oh H, Shibutani T, Pecht M (2012) Precursor monitoring approach for reliability assessment of cooling fans. J Intell Manuf 23(2):173–178
40. Jin X, Chow TW (2013) Anomaly detection of cooling fan and fault classification of induction motor using mahalanobis-taguchi system. Exp Syst Appl 40(15):5787–5795
41. Chang H, Hari A, Mukherjee S, Lakshman T (2014) Bringing the cloud to the edge. In: 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). New York: Institute of Electrical and Electronics Engineers; pp 346–351
42. Aazam M, Zeadally S, Harras KA (2018) Deploying fog computing in industrial internet of things and industry 4.0. IEEE Trans Ind Inf 14(10):4674–4682
43. Chekired DA, Khoukhi L, Mouftah HT (2018) Industrial iot data scheduling based on hierarchical fog computing: A key for enabling smart factory. IEEE Trans Ind Inf 14(10):4590–4602
44. Yin L, Luo J, Luo H (2018) Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. IEEE Trans Ind Inf 14(10):4712–4721
45. Almutairi J, Aldossary M (2021) A novel approach for iot tasks offloading in edge-cloud environments. J Cloud Comput 10(1):1–19
46. Gelman A, Shalizi CR (2013) Philosophy and the practice of bayesian statistics. Br J Math Stat Psychol 66(1):8–38

Publisher's Note