Time Series Classification with Multivariate Convolutional Neural Network

Chien-Liang Liu, *IEEE Member*, Wen-Hoar Hsaio, and Yao-Chung Tu

***Abstract*—Time series classification is an important research topic in machine learning and data mining communities, since time series data exists in many application domains. Recent studies have shown that machine learning algorithms could benefit from good feature representation, explaining why deep learning has achieved breakthrough performance in many tasks. In deep learning, the convolutional neural network (CNN) is one of the most well-known approaches, since it incorporates feature learning and classification task in a unified network architecture. Although CNN has been successfully applied to image and text domains, it is still a challenge to apply CNN to time series data. This work proposes a tensor scheme along with a novel deep learning architecture called multivariate convolutional neural network (MVCNN) for multivariate time series classification, in which the proposed architecture considers multivariate and lag-feature characteristics. We evaluate our proposed method with prognostics and health management (PHM) 2015 challenge data, and compare with several algorithms. The experimental results indicate that the proposed method outperforms the other alternatives using the prediction score, which is the evaluation metric used by the PHM Society 2015 Data Challenge. Besides performance evaluation, we provide detailed analysis about the proposed method.**

***Index Terms*—Time Series Classification, Prognostics and Health Management, Convolutional Neural Networks**

I. Introduction

In recent years, Industry 4.0 and smart manufacturing have attracted extensive attention from both manufacturing and service companies [1]. Prognostics and health management (PHM), which applies predictive algorithms to forecast the future performances of the equipment based on condition-based maintenance (CBM), is a critical research problem in Industry 4.0. It has emerged as a key enabler for industry in predictive maintenance such that company can adopt maintenance actions ahead of failure and avoid from downtime. How to develop prognostic approaches laying on successful diagnosis is of key importance for a successful PHM system deployment [2].

Prognostics could be classified into four types based on applying techniques, including reliability-based approach, physics-based approach, data-driven approach, and hybrid approach, respectively. The physics-based approach builds a physical model for a specified equipment to model its degradation process, in which the model always involves mathematical formulas. This approach is very efficient and descriptive but time-consuming, since a complete and thorough understanding for the equipment is required. In contrast, data-driven approach relies on machine learning algorithms along with collected process parameters without the intervention of domain experts, explaining why data-driven approach has received great attention in recent years. Detailed introduction for these prognostics approaches could refer to [3].

Sensors have been used in the manufacturing industry for the last few decades and they could monitor process parameters, such as temperature, gas pressure, and measurement. These sensors provide not only monitoring functionality, but also the historical logs. These historical logs make it possible to devise data-driven or learning-based prognostic methods. Machine learning algorithms could use available historical logs to train a predictive model, which could predict or forecast equipment condition. More importantly, the developed model could enhance existing equipment, so this is considered as a cost-effective strategy in contrast to other prognostics approaches.

Time series data is characterized by a series of data points indexed in time order, and it is present in various application domains, including, but not limited to, manufacturing, stock price prediction and weather forecasting. In the industry, the sensor data is a typical time series format, since each sensor record is associated with time information. Additionally, it is expected that the sensor records from different sensors at any time t are correlated to a certain extent. Consequently, multivariate time series analysis is more appropriate in sensor data analysis, since the multivariate model considers the interactions and co-movements among a group of time series variables.

The last decade has witnessed the great success of machine learning, since enormous data samples are available at hand, and the advancement of the computing power makes it possible to realize complicated methods. In machine learning, the data samples should be represented as feature vectors, which become the input of the subsequent machine learning algorithms. Traditional machine learning algorithms rely on

Manuscript received Month xx, 2xxx; revised Month xx, xxxx; accepted Month x, xxxx. This work was supported in part by Ministry of Science and Technology, Taiwan, under Grant no. MOST 106-2221-E-009-100 and MOST 106-2218-E-009-031.

Chien-Liang Liu is with the Industrial Engineering and Management Department, University of National Chiao Tung University, Hsinchu, Taiwan (phone: +886-3-5712121 ext. 57309 ; fax: +886-3-5729101; email: clliu@mail.nctu.edu.tw).

Wen-Hoar Hsaio is with the Information Management Center, National Chung-Shan Institute of Science and Technology, Taoyuan, Taiwan.

Yao-Chung Tu is with the Computer Science Department, University of National Chiao Tung University, Hsinchu, Taiwan.

the practitioners to create hand-crafted features. It is apparent that the feature extraction process is a time-consuming and labor-intensive task, and normally does not generalize well. As a result, feature learning or representation learning has become an attractive research topic, since it allows a system to automatically discover the representations needed for classification from raw data.

Deep learning is part of a broader family of machine learning methods, and its goal is to jointly learn data representations and model parameters from a collection of data. The realization of deep learning relies on deep neural networks, which involve a cascade of many layers of nonlinear processing units for feature extraction and transformation. In the architecture, the output of a specific layer becomes the input of the next layer, and the layer architecture makes it possible to learn a hierarchical representation. Among these deep learning architectures, the convolutional neural network (CNN) has achieved a breakthrough improvement on image classification, and the key idea behind CNN is to perform feature extraction using convolutional layers.

The success of CNN inspires us to apply CNN to time series data. However, time series data possesses different characteristics with image data, explaining why only a few research studies dedicating to using CNN to construct models for time series data. First, the image data generally involves RGB three channels, while time series data does not have channel property. To encode image data, deep learning architectures consider tensor as a basic data structure, so we propose a tensor scheme to transform the time series data into 3D tensors, making it possible to exploit local interactions among variables with convolution operation. Moreover, the proposed tensor scheme considers natural temporal ordering of time series data using the information within a sliding window to organize the information with a tensor representation, so that the relationship between nearby data points could be considered as well. Second, image data does not have time information and the classification results are irrelevant to the ordering of the images, while the ordering of the data is an important property for time series data. Finally, multivariate analysis should consider the interaction and correlation between the variables, while image data does not have multivariate property.

The contributions of this work are two-fold. First, we devise a novel CNN architecture called multivariate convolutional neural network (MVCNN) which dedicates to deal with multivariate time series data, in which the model considers the interactions and co-movements among a group of time series variables. The proposed model gives it a way to deal with multivariate time series based on the characteristics of multivariate time series data rather than images. Second, we conduct experiments on PHM 2015 challenge data set and occupancy detection data set [4] to assess the proposed method and compare with other alternatives. Meanwhile, we also provide detailed analysis in this work.

The rest of this paper is organized as follows. Section II presents related surveys and techniques. Section III introduces the proposed algorithm. Section IV shows the experimental results and Section V presents detailed discussion. The conclusions and future work are presented in Section VI.

II. RELATED WORK

This work proposes a tensor scheme along with a novel deep learning architecture for time series classification, in which the model considers multivariate and lag-feature characteristics of time series data simultaneously. Afterwards, this section introduces PHM 2015 data challenge and the works that are related to our proposed architecture.

A. PHM 2015 Data Challenge

As more and more sensors are available in the manufacturing industry, the monitoring information obtained from sensors makes it possible to apply predictive maintenance to equipment, so that maintenance actions could be taken ahead of failure to avoid unplanned downtime. Prognostics focuses on predicting the time at which a system or a component will no longer perform its intended function, and it is crucial to predictive maintenance. Thus, PHM society held a data challenge PHM 2015 [1], which was a competition open to all potential conference attendees and focused on fault detection and prognostics for industrial plant monitoring. The given data spans a period of approximately three to four years, and includes time series of sensor measurements and control reference signals for each plant component, times series data for cumulative energy consumed and instantaneous power from each zones within a given plant, and plant fault events which are characterized by a start time, an end time, and a failure type. The participants attending to this challenge should be scored on their ability to predict future failure events from the given time series data and to precisely localize faults in time.

In the data challenge PHM 2015, Xiao [5] won the first place, and he proposed an effective strategy to predict industrial plant faults based on the ensemble of classification methods, in which the ensemble involves penalized logistic regression, random forest [6] and gradient boosted decision trees [7]. He formulated the original problem as a classification problem, in which the start time and end time for a fault are predicted sequentially. Kim et al. [8] used a different approach in the competition. First, they applied a fault log recovery method to the data set. Then, they used Fisher discriminant analysis (FDA) to classify normal and faulty data. Finally, they merged the adjacent faulty logs into final results. Their approach got the second prize in this competition. The final scores [2] for the first place and the second place winners of PHM 2015 as mentioned above are $21,016$ and $20,640$ with the scoring formula specified by that competition, respectively.

B. Machine Learning for Fault Detection

Many machine learning algorithms for time series data have been proposed over the last decade [9]–[13]. For example, dynamic time warping (DTW), which is used to measure similarity between time series data, allows similar shapes

[1] PHM 2015 Data Challenge: https://www.phmsociety.org/events/conference/phm/15/data-challenge

[2] Final Scores for PHM Data Challenge 2015: http://phmdatachallenge.freeforums.net/thread/22/final-scores-data-challenge-2015

to match even if they are out of phase in the time axis, explaining why it has been widely used in science, medicine, industry and finance domains [9]. Lin et al. [10] formulated a new symbolic representation of time series, which improves performances of data mining algorithms. Baydogan et al. [14] proposed a framework to classify time series data based on a bag-of-features representation. The data in the plants always possesses high-dimensional characteristics, so using dimensionality reduction techniques to prevent from the curse of dimensionality is a popular method. Among these methods, principal component analysis (PCA) and Fisher's discriminant analysis (FDA) have been successfully used in pre-processing step when applying to detect faults in chemical processes [15]–[17].

Data-driven approaches to fault detection have attracted significant attention in recent years, so many research studies have applied machine learning algorithms to fault detection. For example, k-nearest neighbor and fuzzy-logic have been applied to fault diagnosis in semiconductor manufacturing processes [18]. Nonlinear classification approaches are good at discovering complicated patterns from data, so support vector machine (SVM) [19] and artificial neural networks [20] have been used to detect gearbox failure [21] and chemical process fault [22].

Ensemble learning combines various classifiers into the model, and can generally improve performance. Random forest [6] and gradient boosting trees [7] are two typical ensemble learning algorithms that use decision trees as the base classifiers, in which the former uses bagging technique and the latter uses boosting technique to combine enormous decision trees. It is worth mentioning that ensemble learning technique has been widely used in data science competitions, since the model performance could generally benefit from the combination of various algorithms.

C. Convolutional Neural Network

The convolutional neural network (CNN) [23] is a class of deep, feed-forward neural networks that have been successfully applied to many application domains. The CNN is characterized by shared-weights and translation invariance. As compared with traditional ANN, the neurons in CNN respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. The local-connectivity characteristics of the neurons with parameter sharing scheme could dramatically reduce the number of parameters, giving a base to prevent CNN from overfitting.

The CNN has been widely used in image classification problem. For example, Szegedy et al. [24] proposed a deep convolutional neural architecture called inception, which achieved great performance in visual classification and detection. One important property of inception module is that it allows for extending the depth and width of the network, while keeping the computational difficulties constant. The inception operates in a multi-scale process such that the network could process the input data with various scales and perform max pooling simultaneously, and then concatenate all the filter results. As a result, inception model is characterized by the ability to extract multi-level and multi-scale features from each input.

Although CNN has been successfully applied to images, texts and signals, applying CNN to time series data is not a straightforward task, since time series data possesses different characteristics with these data formats. Cui et al. [25] proposed an end-to-end neural network model called multi-scale convolutional neural network (MCNN), which incorporates feature extraction and classification in a single framework. MCNN applies different transformations to time series data, including different scales, frequencies and sampling rates. Once the transformations are completed, MCNN extracts high-level features by using convolutional layer. Then, MCNN concatenates all the extracted features, and makes the classification prediction.

Yi et al. [26] proposed grouped convolutional neural networks, which is appropriate for high-dimensional multivariate time series data. The grouped convolutional neural networks comprise two algorithms to group features. First, the variables are partitioned into different groups by spectral clustering [27] [28]. Second, the architecture incorporates a clustering coefficient into the architecture, and the coefficient could be adjusted by the backpropagation algorithm.

The estimation of remaining useful live (RUL) plays an important role in PHM. Babu et al. [29] proposed to use CNN to predict RUL, in which the model receives a two-dimensional matrix as the input. They used a typical CNN architecture to build the regression model, which comprises two pairs of convolution layers and pooling layers, and one fully-connected layer. Long short-term memory (LSTM) [30] is a widely used architecture in sequence-to-sequence learning, so Zheng et al. [31] proposed to use LSTM to predict RUL. Although LSTM is designed for sequence data, the training of LSTM requires more GPU resource than CNN, and the model training is also more difficult than CNN. Recent research studies indicate that certain convolutional architectures could reach state-of-the-art accuracy in language modeling, audio synthesis, and machine translation [32]–[34]. Zhang et al. [35] proposed to used deep belief networks (DBN) ensemble to estimate RUL, since DBN provides a way to learn hierarchical feature representations and model performance could always benefit from the combination of various algorithms. Note that DBN typically involves layer-wise pre-training, followed by the fine-tuning of the entire network. It is apparent that the proposed work is different from LSTM and DBN from the perspective of architecture. As for [29], several differences exist between their work and the proposed MVCNN, even though both are based on CNN. First, the prediction of RUL is a regression problem, while this work focuses on classification task. Second, the inputs of the two models are different. Third, the architectures are also different. The proposed model considers the characteristics of multivariate time series data to design model, while [29] is based on a typical CNN architecture to deal with time series data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2864702, IEEE Transactions on Industrial Electronics

III. Multivariate Convolution Neural Network

CNN is originally designed for image applications, so many properties of CNN are specific for image representations. For example, an image comprises RGB channels, so the number of channels for input tensor is three. Meanwhile, the local-connectivity is also useful for images, since the relationship between nearby pixels is higher than those that are far away. However, this work focuses on PHM problem, in which the data is presented in a time-series manner. Thus, this work proposes a deep learning architecture to deal with multivariate time series data, in which the proposed model considers to exploit local interactions among variables with convolution operation.

A. Data Notation

The time series data is the data in sequence with time-stamps, and this work denotes time-stamps by $t_1, t_2, \ldots, t_N$ to indicate that N time-stamps are used in time series data. Meanwhile, the data is collected from K sensors, and we introduce the data $\mathbf{x}_i = \{x_{i1}, \ldots, x_{iK}\}$ of size K, in which x_{ik} $(1 \leq k \leq K)$ represents the sensor information from a specific sensor at time t_i. Additionally, this work proposes to make prediction at each time-stamp, and the prediction outcomes involve normal and abnormal classes. Thus, the problem could be transformed into a binary classification problem. The training data is represented as $\mathbf{D} = \{(t_1, \mathbf{x}_1, y_1), \ldots, (t_i, \mathbf{x}_i, y_i), \ldots, (t_N, \mathbf{x}_N, y_N)\}$, in which $\mathbf{x}_i$ and y_i represent the data and the classification label at time t_i. Additionally, this work uses $h \times w \times d$ to represent the dimension of a 3D tensor, in which d is the number of channels or depth, h and w are the height and width, respectively.

B. Multivariate Convolutional Neural Network (MVCNN)

This work proposes an input tensor transformation scheme along with an architecture called MVCNN for the multivariate time series problem. The proposed input tensor scheme transforms the multivariate time series data into appropriate tensor representation by wrapping the sliced time series data, so that the MVCNN could deal with multivariate data, in which we use filters to exploit local interactions among variables and apply sliding windows to extract the lag-feature characteristics for time series data. The PHM 2015 challenge requires to predict start time and end time for each fault, explaining why we use two different models to predict start time and end time sequentially. Figure 1 presents the framework of MVCNN, which comprises four stages, including input tensor transformation, univariate convolution, multivariate convolution, and fully connected stages. The introductions for the four stages are listed below.

1) Input tensor transformation stage: It is apparent that the formats of image data and time series data are different. Each RGB image comprises three channels, each of which is composed by a 2D array. In other words, one can represent an image as a 3D tensor, namely $h \times w \times 3$. Consequently, the convolutional layer uses 3D filters to scan the image. Given the great success of the CNN on image classification, we propose an image-like tensor scheme to encode multivariate time series data, so that the MVCNN could learn interactions among variables and lag-feature characteristics.

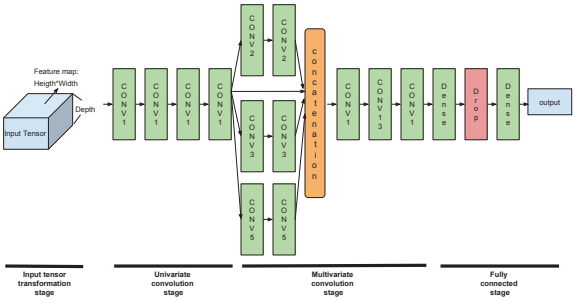

Fig. 1. The framework of MVCNN

One important property of time series data is that it has a natural temporal ordering, since it is a series of data points indexed in time order, so the relationship between nearby data points should be considered as well. As a result, we apply sliding windows or sub-sequences [36] to time series data, so that we could process data and perform classification in batches.

Given the signals within a window, we apply the proposed input tensor transformation scheme to transform time series data into tensors, in which the depth of the tensors is the length of the sliding window. For example, Figure 2 presents four sensor signals (a, b, c and d), and a sliding window of length four. For each sensor signal within the sliding window, four values in time series are available for us to construct a 3D tensor of size $1 \times 1 \times 4$. Then, we concatenate the four $1 \times 1 \times 4$ tensors in x axis and y axis to form a tensor of size $2 \times 2 \times 4$, where the depth is the same as the interval of the sliding window.

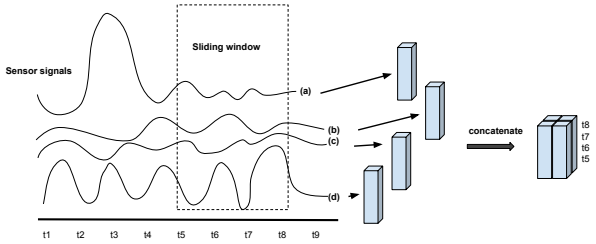

Fig. 2. An example of input tensor transformation scheme

2) Univariate convolution stage: Once the input tensor transformation process is completed, we apply univariate convolution to the transformed tensors. The goal of the univariate convolution stage is to separately extract the features from each sensor. The univariate convolution stage uses filter whose plane size is 1×1 to scan the tensor, and the filter's depth is determined by the depth of input tensor. The 1×1 convolution was first introduced in the network in network architecture [37], and then used by GoogLeNet [38] to make network deeper. This work uses 1×1 convolution to add non-linearity to enhance the abstraction ability of the local model.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2864702, IEEE Transactions on Industrial Electronics

Figure 3 presents an example of univariate convolution, in which the input tensors are the results of Figure 2. Then, the proposed method applies 1×1 filter and performs convolution operation on the input tensors, and then results are stacked on the feature maps of output. In this example, we assume the number of filters is five, explaining why the output tensor is the size of $2 \times 2 \times 5$ after the univariate convolution stage. Once the 1×1 convolution operations are completed, we apply rectified linear unit (ReLU) [39] activation function to the outputs to obtain the non-linear results. Note that the width and height of the output tensor are the same as those of the input tensor in univariate convolution stage.

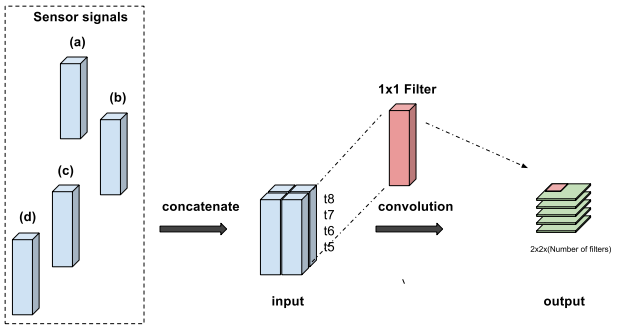

Fig. 3. An example of univariate convolutional layer

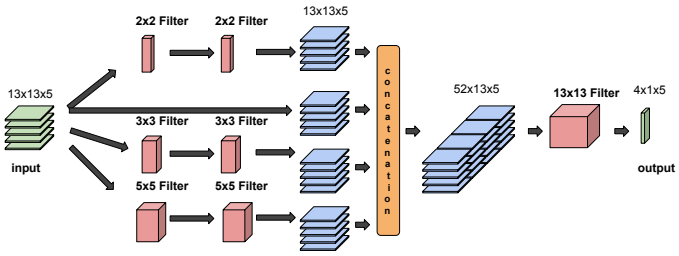

Fig. 4. An example of multivariate convolutional layer

3) Multivariate convolution stage: Inspired by Inception architecture [24], we design a four-way convolution to find the optimal local construction as presented in Figure 4. For image inputs, different scales of filters extract different local features. Large-scale filter may find the symmetric property, while small-scale filter could find particular characteristics. For time series data, different scales of filters give a way to extract features of the interaction between sensors. In the proposed architecture, we use three scales of filters, including 2×2, 3×3 and 5×5. Each way of convolutional layers comprises two layers. The first convolutional layer is to group the local sensors together, and it can be considered to find the local correlation of sensors. The second convolutional layer is to take the insightful information between groups. Note that we use the same padding on each convolutional layer, so the size of tensor plane is the same as output tensor of univariate convolution stage. Moreover, we apply ReLU activation function right after each convolutional layer. Besides the three scales of filters, we also directly use the output of the univariate convolution stage as another way of convolution.

Once the above process is completed, we concatenate all the tensors in the direction of x axis, which is different from the inception [24] to stack the tensors in the z axis. The reason we concatenate in x axis is to keep the unique features from each group of convolutional layers. Next, we apply a filter to the output of concatenation of the inception units, in which the filter size is the same size as the input tensor. As a result, the filter condenses each tensor from different ways into a single vector. As shown in Figure 4, the output tensor after this stage is of the size $4 \times 1 \times 5$.

4) Fully connected stage: Finally, we apply fully connected layers to the output tensors of the multivariate convolution stage. Besides, we use dropout layer before the final fully connected layer to prevent the model from over-fitting. The final output is the softmax function, which transfers the prediction into probabilities. Therefore, the output of the MVCNN is the probability of an abnormal event.

We use cross-entropy function as the loss function, and Equation (1) shows the cross-entropy function, where $p(i)$ is the true distribution, and $q(i)$ is estimated distribution. The proposed MVCNN is trained to minimize the loss function, and we apply adaptive moment estimation (Adam) optimization algorithm [40] with initial learning rate 10^{-4} to train our model, which combines elements of Adagrad and momentum together.

$$H(p,q) = -\sum_{i} p(i) \log q(i) \tag{1}$$

The proposed model could be applied to time series data, and we conduct experiments on two datasets. One of the dataset is the PHM 2015 challenge, which involves the prediction of start time, end time, and fault type. Meanwhile, the PHM 2015 challenge data set comprises 33 plants, each of which is supposed to be in different environment. In the implementation for PHM 2015 challenge, we use different models to predict start time and end time for each fault in a specific plant sequentially. Moreover, this work considers to use binary classification to model the problem, so each fault is predicted by a model. Thus, the number of models is at most 330. Some plants do not have all five fault types, and the number of implemented models for PHM 2015 challenge is 270.

IV. EXPERIMENTS

In the experiments, this work compares with several machine learning methods on PHM 2015 challenge data set.

A. PHM 2015 data challenge

The goal of the 2015 PHM data challenge competition is to detect and diagnose failure using incomplete data. The available data comprises sensor measurements, control reference signals, and fault logs. The prediction of the predictive model includes start time, end time and fault type of failures.

The data instances are presented in time-series format, since they are collected from the sensors. Each sensor data is recorded at 15-minute interval. This work represents each

observation as a pair of feature vector and time-stamp information, namely $(\mathbf{x}, t)$, where $\mathbf{x} = [x_1, \ldots, x_k]^T$ is a k-dimensional feature vector and t represents the time when sample $\mathbf{x}$ is observed. Besides the data representation, the training data comprises label information, including fault type, and the beginning as well as end time of the fault. Thus, a fault event is represented as a pair of time-stamps and a fault type, namely (t^{start}, t^{end}, y), where t^{start} is the beginning time of the fault, t^{end} is the end time of the fault, and $y \in \{1, 2, 3, 4, 5\}$ is the corresponding fault label. A sequence s of observations in data set $\mathbf{D}$ corresponding to a fault event (t^{start}, t^{end}, y) is $s = \{(\mathbf{x}, t) | (\mathbf{x}, t) \in \mathbf{D}, t \geq t^{start}, t \leq t^{end}\}$.

To evaluate performance, PHM 2015 data challenge defines a scoring formula based on several metrics. True positive (TP) is the number of those predictions with correct fault type along with start time and end time identified within one hour before or after the real occurrence time. False positive (FP) is the number of those predictions with correct fault type but incorrect start time or end time, and false negative (FN) is the number of predictions that fail to identify faults. The misclassification (MS) is the number of those predictions that are correct in start time and end time, but incorrect in fault type.

With these metrics, the evaluation formula is listed in Eq. (2), and the scores in the experimental results are obtained using the following scoring formula.

$$\text{Score} = \text{TP} \times 10 - \text{FP} \times 0.1 - \text{FN} \times 0.1 - \text{MS} \times 0.01 \quad (2)$$

B. Data set

PHM 2015 challenge data set comprises three parts, including train (33 plants), test (15 plants), and validate (15 plants). For each plant, the data set provides time series logs, including sensor signals (S1-S4) and control references signals (R1-R4) from each plant component. Note that the number of components is different for each plant. Additionally, the time series logs for cumulative energy consumed (E1) and instantaneous power (E2) from each zone are also included in the data set.

The purpose of this competition is to predict fault events, including start time, end time, and fault type. Six fault types are given, but this competition only focuses on the prediction of fault 1-5. The training set comprises complete fault events, while the test set and validate set have only partial fault events, since some of them have been removed in the second half of them. The boundary between first and second half is given. The goal is to recover the removed fault events.

The deleted fault events for test set and validation set are unavailable, so this work simulates the settings used in [5] to randomly remove 50% fault events from the train set in the second half, so that we could evaluate performance with the removed ones. The boundary used to separate the original training set is the first day at the last year for each plant. Furthermore, we divide the first half of train set into two parts. The first part is called training set, which comprises the data samples with complete fault events and is used to train our prediction model. The other one is called validation set, in which 50% of the data samples with fault events are removed. The purpose of the validation set is to validate our model and tune model parameters. Afterward we briefly use the terms, training set and validation set, to represent those two parts of data mentioned above, respectively.

C. Experimental Settings

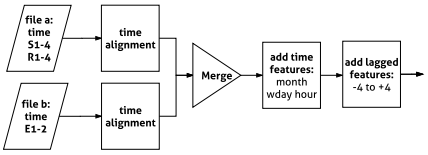

Fig. 5. The preprocessing flow

1) Experimental Flow: Figure 5 presents the pre-processing flow. Appropriate data pre-processing is required, since the sampling rate for the signal should be 15 minutes, but not all logs record the events every 15 minutes due to time delay or other problems. As a result, time alignment is applied to the data set to fix this problem. Besides data pre-processing, we also apply feature engineering to include more features into the model. Thus, additional features such as month, hour, weekday, and lagged features of all sensor signals are included in the prediction model.

As mentioned above, the PHM data belongs to time series format, and we follow the procedures proposed by Wei [5] to process data, so that the transformed feature vectors possess the characteristics of time-series data. For any signal $S(t)$ at time t, we consider the signal changes before and after time t by adding lagged features that are within $S(t-z)$ and $S(t+z)$, so that the signals within the $2z+1$ time steps are considered as a sliding window, in which the hyper-parameter z is 4 in the experiments.

Additionally, we transform the detection problem into a binary classification problem, indicating that the proposed method has to build a predictive model for each fault type. Note that the number of specific faults is insufficient to train a predictive model for some factories, so we discard those data samples where the number of specific faults in a factory is less than 199.

Next, we use different models to make predictions on start time and end time. For the prediction of start time, we suffer from imbalanced data set problem, since most data samples belong to normal class, and only a few data samples are abnormal. However, we are interested in the abnormal cases, so it is required to further pre-process the data. Tu et al. [41] empirically showed that the under-sampling algorithm could achieve better performance than other alternatives, so this work applies under-sampling technique to deal with imbalanced problems.

Furthermore, this work uses probabilistic learning model as the prediction algorithm, so the outcomes are probabilistic results. To transform the probabilistic outcomes into binary classification results, a threshold value is required when performing the transformation. This work proposes a simulating cut-point approach to determine the threshold value when determining whether a data sample is abnormal given its

probability. The criterion used by simulating cut-point is based on the scoring formula listed in Equation (2), which is the performance metric defined by PHM 2015 data challenge, rather than ROC curve. It adjusts the threshold value from 0.5 to 1 and increases by 0.01, and the adjusted threshold value that results in the highest score of the validation set is the simulating cut-point. Note that the above process is applied to validation set, so the proposed simulating cut-point is a cross-validation method.

Once the estimation of our proposed MVCNN model is completed, the abnormal probabilities of data samples are available. As shown in Figure 6, we consider six consecutive data samples once, and then check whether the maximum probability of those data samples exceeds the threshold value. If such a data sample is present in the sequence, the time-stamp for that data sample is the predicted start time for the specified fault. For example, assume the threshold is 0.7, and the two sequences presented in Figure 6 satisfy the requirement mentioned above, so time-stamps t_3 and t_7 are two predicted occurrences of the specified fault.

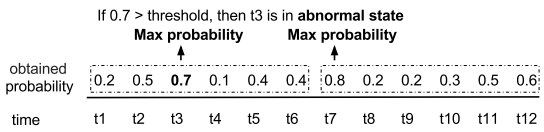

Fig. 6. An example of process

In a time series data, the start time must precede the end time, explaining why the prediction of end time should be based on the occurrence of start time. Once the prediction of start time is completed, we make a prediction on the end time. Intuitively, the number of corresponding end time is enormous, since all the time-stamps after the start time could be possible candidates. To reduce the number of candidates, we first analyze the intervals for all fault events to find a boundary that exceeds 95% fault intervals, in which the interval is defined as the difference between end time and start time. With the start time in training set, say t_{start_train}, and its corresponding boundary, say b, we collect the logs within the interval of t_{start_trian} to $t_{start\_train} + b$ as training set for end-time model. Then, with the start time obtained from the previous step, say t_{start_test}, the data sample with the highest probability in the interval of t_{start_test} to $t_{start\_test} + b$ is the occurrence of the end time. The pair of start time and end time is considered as a prediction. We have released the source code of our model on github [3], and many detailed parameters as well as settings are available in the source code.

2) Comparison Methods: In the experiments, we compare the proposed method with five comparison methods, including extreme gradient boosting (XGBoost) [42], random forest (RF) [6], logistic regression (LR), vanilla convolutional neural network and vanilla neural network [43], which is a feed-forward network with a single hidden layer. The XGBoost uses tree ensemble to combine the prediction of multiple trees, and the model is built in a stage-wise fashion. The optimization takes the Taylor expansion of the loss function up to the second order, and the model additionally uses a more regularized model formalization to control over-fitting, which gives it better performance.

Random forest is also an ensemble learning by creating an entire forest of random decision trees to make predictions. As compared with XGBoost, the training algorithm for random forests applies bagging technique to generate training samples, and fits trees to these samples. Moreover, the learning process uses a random subset of the features for node splitting. Once training step is completed, the model makes prediction by averaging the predictions from all the individual trees.

In statistics, logistic regression is used to explain the relationship between one dependent binary variable and one or more independent variables. The prediction is made in terms of a logistic function with the linear combination of the features as the input, and the prediction outcome is a real value between 0 and 1, explaining why the outcome could be interpreted as a probability.

As for vanilla NN and vanilla CNN, they are baseline methods. Vanilla NN comprises five layers of dense blocks with ReLU activation function followed by a fully-connected layer and a softmax layer. On the other hand, vanilla CNN comprises six layers of convolution blocks with ReLU activation function, one layer of dense blocks with ReLU activation function, and a fully-connected layer followed by a softmax layer. In particular, the vanilla CNN and vanilla NN have similar number of parameters with our proposed MVCNN. Table I shows the number of parameters for the three models. Besides NN-based methods, XGBoost and random forest are state-of-the-art algorithms, and they have been widely used in many data science competitions.

It is worth noting that Xiao [5], who is the winner of PHM 2015, used ensemble learning technique to combine gradient boosting trees, random forest and logistic regression in a model. It is difficult to re-produce the results without the details of implementation and model combination, so this work uses XGBoost, random forest and logistic regression to compare with the proposed model. Ensemble learning is a commonly used technique in data science competition, since model performance could always benefit from the combination of various algorithms. However, it is a time-consuming task to train several models, and it will become a tedious process when deploying many models to a production system. Furthermore, maintenance of several models is also a burden for the system administrator.

D. Experimental Results

In the experiments, we use five comparison methods to compare with the proposed MVCNN on PHM 2015 challenge data set. This work transforms the prediction problem into a binary classification problem, so 270 models are constructed to detect the abnormal state along with its start time and end time. Each predictive model corresponds to a plant and a fault type.

Table II presents the experimental results with our proposed simulating cut-point approach. The evaluation metric

[3]MVCNN: https://github.com/AlanHsiau/MVCNN

TABLE I
THE NUMBER OF PARAMETERS FOR NN-BASED ARCHITECTURES

	MVCNN	Vanilla CNN	Vanilla NN
params	8280304	8258456	8253952

TABLE II
THE NUMBER OF TPS, FPS AND FNS (SIMULATING CUT-POINT)

Algorithm	TP	FP	FN	MS	Score	Precision	Sensitivity
MVCNN	3996	100528	6179	3609	**29253.21**	**3.8230**	0.3927
Vanilla CNN	3637	108808	6538	4095	24794.45	3.2344	0.3574
Vanilla NN	3681	105758	6494	4034	25544.46	3.3635	0.3617
XGBOOST	4328	152895	5880	4474	27357.76	2.7527	0.4239
RF	4368	157107	5807	4830	27340.3	2.7050	**0.4292**
LR	2869	253613	7306	7804	2520.06	1.1185	0.2819

includes true positives, false positives, false negatives and mis-classification. Once these metrics are available, one can calculate precision, sensitivity and the score based on the formula listed in Equation (2).

E. Discussions

1) PHM 2015: As shown in Table II, the proposed multivariate CNN achieves the best performance on the score and precision metrics, since it can significantly decrease false positives. The number of false positives of MVCNN is less than other alternatives, and it reduces 34.25% false positives as compared with XGBOOST method, which has been widely used and recognized in many data science competitions. However, the experimental results also indicate that the decreasing of false positives also leads to the decreasing of true positives for the proposed method, explaining why MVCNN fails to perform well on sensitivity metric. Consequently, we believe that the MVCNN is more conservative than other algorithms. Besides, the score of the proposed method is higher than that of the winner of PHM 2015 challenge, so we believe that the proposed method is competitive.

Although CNN has achieved good performances on many application domains, the performance difference between vanilla CNN and vanilla NN is insignificant. In the implementation, we also apply the proposed tensor transformation to the vanilla CNN, but it seems like that vanilla CNN could not benefit from the proposed tensor transformation. One of the reasons is that although the proposed tensor transformation makes it possible to encode time series data into tensor representation, the subsequent deep learning architecture should consider the characteristics of multivariate time series data to get good performance. In contrast, the proposed MVCNN could achieve significant improvement on PHM 2015 data set. Table II indicates that MVCNN gets 14.51% improvement on score, 13.66% improvement on precision and 8.55% improvement on sensitivity comparing to vanilla NN.

2) Effectiveness of Simulating Cut Point: For probabilistic algorithms, the prediction outcomes are probabilities. To transform a probability value into a discrete result with two classes, one typical approach is to set 0.5 as the threshold value. If the probability is greater than 0.5, the classification outcome is positive. Otherwise, the classification outcome is negative. As mentioned above, the PHM data set is imbalanced, so 0.5 is not a good threshold value. Consequently, this work proposes a simulating cut-point to determine the cut-point with cross-validation technique.

To validate effectiveness of our proposed simulating cut-point, we also conduct experiments to determine the threshold by calculating the cut-point from the ROC curve [44] by cross-validation technique, which is a commonly used technique to determine the cut-point. The experimental results are presented in Table III. The experimental results listed in Table III and Table II indicate that most methods could benefit from the proposed simulating cut-point except random forest. It is concluded that the proposed simulating cut-point is based on the scoring formula defined by 2015 PHM data challenge. Although random forest could not benefit from the proposed simulating cut-point, it still works well with simulating cut-point.

3) Occupancy Detection Data: Besides PHM 2015 data set, we apply the proposed MVCNN to another time series data set to evaluate whether the proposed MVCNN could be applied to other time series data sets. The occupancy detection data set [4] involves the task of classifying the occupancy from light, temperature, humidity and CO2 measurements, and the sampling rate is 1 minute. It is a binary classification problem. The data set comprises 1 training set containing 8,144 events, and two testing sets containing 2,666 events and 9,753 events, respectively.

In this experiment, we use the random forest to compare with the proposed MVCNN, since random forest performs well on the previous experiments. We use several features in the model, including light, temperature, humidity and CO2 measurements. The architecture of the proposed MVCNN is the same as the model of predictions for start time on PHM 2015 data set. Additionally, For any signal $S(t)$, we add $S(t-z)$ as lagged features, in which the parameter z is 0 to 4. Table IV presents the experimental results, which indicate that the proposed method outperforms random forest on the two testing sets.

The last decade has witnessed the great success of deep

TABLE III
THE NUMBER OF TPS, FPS AND FNS (ROC CUT-POINT)

Algorithm	TP	FP	FN	MS	Score	Precision	Sensitivity
MVCNN	4161	141601	6014	4682	26801.68	2.8546	0.4089
Vanilla CNN	3870	156677	6305	5376	22348.04	2.4105	0.3803
Vanilla NN	3896	152787	6279	5346	22999.94	2.4865	0.3828
XGBOOST	4367	201960	5841	5367	22836.23	2.1165	**0.4278**
RF	4335	140219	5840	4350	**28700.6**	**2.9988**	0.4260
LR	334	92784	9841	2645	-6948.95	0.3586	0.0328

TABLE IV
THE ACCURACY OF OCCUPANCY DETECTION DATA

Algorithm	first testing set	second testing set
MVCNN	**0.9740**	**0.9772**
RF	0.9505	0.9716

learning, and this work proposes a input tensor scheme along with a novel deep learning architecture to tackle time series data with deep learning technique. The experimental results on the two data sets indicate that the proposed MVCNN is comparative in dealing with time series data.

V. CONCLUSION

Time series data exists in many application domains, and this work focuses on multivariate time series problem and proposes a deep learning architecture to deal with it. Central to the proposed MVCNN is to consider the characteristics of time series data in the model, and the experimental results on PHM 2015 data set and occupancy detection data set indicate that the proposed MVCNN outperforms other alternatives. Several future works are possible. For example, the investigation of the arrangement of the tensor in the proposed tensor scheme is an interesting research direction. Besides, this work deals with imbalance problem in the pre-processing stage, but considering imbalance problem in a deep learning architecture may result in a more effective architecture. Finally, the experimental results indicate that the proposed model works well on the two datasets, so it is expected that the proposed model could learn good feature representations from raw data. Besides the effectiveness of feature extraction, visualization of the feature extraction is an interesting and valuable direction to discover the insights of the feature learning.

REFERENCES

[1] Y. Liao, F. Deschamps, E. de Freitas Rocha Loures, and L. F. P. Ramos, "Past, present and future of industry 4.0 - a systematic literature review and research agenda proposal," *International Journal of Production Research*, vol. 55, DOI 10.1080/00207543.2017.1308576, no. 12, pp. 3609–3629, 2017. [Online]. Available: http://dx.doi.org/10.1080/00207543.2017.1308576

[2] Z. Zeng, F. D. Maio, E. Zio, and R. Kang, "A hierarchical decision-making framework for the assessment of the prediction capability of prognostic methods," *Journal of Risk and Reliability*, vol. 231, no. 1, pp. 36–52, 2017. [Online]. Available: http://EconPapers.repec.org/RePEc:sae:risrel:v:231:y:2017:i:1:p:36-52

[3] H. M. Elattar, H. K. Elminir, and A. M. Riad, "Prognostics: a literature review," *Complex & Intelligent Systems*, vol. 2, DOI 10.1007/s40747-016-0019-3, no. 2, pp. 125–154, Jun. 2016. [Online]. Available: https://doi.org/10.1007/s40747-016-0019-3

[4] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and co 2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, pp. 28–39, 2016.

[5] W. Xiao, "A probabilistic machine learning approach to detect industrial plant faults," *arXiv preprint arXiv:1603.05770*, 2016.

[6] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[8] H. Kim, J. M. Ha, J. Park, S. Kim, K. Kim, B. C. Jang, H. Oh, and B. D. Youn, "Fault log recovery using an incomplete-data-trained fda classifier for failure diagnosis of engineered systems," 2016.

[9] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[10] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[11] M. Majidpour, C. Qiu, C. P. Chu, R. Gadh, and H. R. Pota, "Fast prediction for sparse time series: Demand forecast of EV charging stations for cell phone applications," *IEEE Trans. Industrial Informatics*, vol. 11, DOI 10.1109/TII.2014.2374993, no. 1, pp. 242–250, 2015. [Online]. Available: https://doi.org/10.1109/TII.2014.2374993

[12] G. A. Susto, A. Schirru, S. Pampuri, and S. F. McLoone, "Supervised aggregative feature extraction for big data time series regression," *IEEE Trans. Industrial Informatics*, vol. 12, DOI 10.1109/TII.2015.2496231, no. 3, pp. 1243–1252, 2016. [Online]. Available: https://doi.org/10.1109/TII.2015.2496231

[13] R. D. Telford, S. J. Galloway, B. Stephen, and I. M. Elders, "Diagnosis of series DC arc faults - A machine learning approach," *IEEE Trans. Industrial Informatics*, vol. 13, DOI 10.1109/TII.2016.2633335, no. 4, pp. 1598–1609, 2017. [Online]. Available: https://doi.org/10.1109/TII.2016.2633335

[14] M. G. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2796–2802, 2013.

[15] L. H. Chiang, E. L. Russell, and R. D. Braatz, "Fault diagnosis in chemical processes using fisher discriminant analysis, discriminant partial least squares, and principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 50, no. 2, pp. 243–252, 2000.

[16] L. H. Chiang, M. E. Kotanchek, and A. K. Kordon, "Fault diagnosis based on fisher discriminant analysis and support vector machines," *Computers & chemical engineering*, vol. 28, no. 8, pp. 1389–1401, 2004.

[17] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process," *Journal of Process Control*, vol. 22, no. 9, pp. 1567–1581, 2012.

[18] Q. P. He and J. Wang, "Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 20, no. 4, pp. 345–354, 2007.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] S.-C. Wang, "Artificial neural network," in *Interdisciplinary computing in java programming*, pp. 81–100. Springer, 2003.

[21] B. Samanta, "Gear fault detection using artificial neural networks and support vector machines with genetic algorithms," *Mechanical Systems and Signal Processing*, vol. 18, no. 3, pp. 625–644, 2004.

[22] Z.-y. GUO, B. CHENG, M. YE *et al.*, "Advances in natural computation," 2006.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[25] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *CoRR*, vol. abs/1603.06995, 2016. [Online]. Available: http://arxiv.org/abs/1603.06995

[26] S. Yi, J. Ju, M.-K. Yoon, and J. Choi, "Grouped convolutional neural networks for multivariate time series," *arXiv preprint arXiv:1703.09938*, 2017.

[27] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[28] F. R. Bach and M. I. Jordan, "Learning spectral clustering," in *Advances in neural information processing systems*, pp. 305–312, 2004.

[29] G. S. Babu, P. Zhao, and X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *International conference on database systems for advanced applications*, pp. 214–228. Springer, 2016.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *Prognostics and Health Management (ICPHM), 2017 IEEE International Conference on*, pp. 88–95. IEEE, 2017.

[32] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *arXiv preprint arXiv:1705.03122*, 2017.

[33] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[34] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," *arXiv preprint arXiv:1612.08083*, 2016.

[35] C. Zhang, P. Lim, A. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2306–2318, 2017.

[36] B. Hu, Y. Chen, and E. J. Keogh, "Time series classification under more realistic assumptions," in *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA.*, DOI 10.1137/1.9781611972832.64, pp. 578–586, 2013. [Online]. Available: https://doi.org/10.1137/1.9781611972832.64

[37] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013. [Online]. Available: http://arxiv.org/abs/1312.4400

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[39] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[40] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] W.-H. H. Yao-Chung Tu, Ching-Hsien Lee and C.-L. Liu, "Data sampling to imbalanced data of industrial plant fault prediction," National Chiao Tung University, Tech. Rep., 8 2016, presented at 2016 International Symposium on Semiconductor Manufacturing Intelligence (ISMI2016).

[42] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, DOI 10.1145/2939672.2939785, pp. 785–794. New York, NY, USA: ACM, 2016. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[43] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[44] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

Chien-Liang Liu received the M.S. and Ph.D. degree in Department of Computer Science from National Chiao Tung University, Taiwan, in 2000 and 2005, respectively. He is currently an associate professor in Department of Industrial Engineering and Management at National Chiao Tung University, Taiwan. His research interests include machine learning, data mining, deep learning, and big data analytics.

Wen-Hoar Hsaio received the M.S. and Ph.D. degree in Department of Computer Science from National Chiao Tung University, Taiwan, in 1996 and 2015. He is currently an Engineer in National Chung-Shan Institute of Science and Technology, Taiwan. His research interests include information retrieval, data mining, and machine learning.

Yao-Chung Tu received the M.S. degree in Department of Computer Science from National Chiao Tung University, Taiwan in 2017. His research interests include machine learning and deep learning.