# Can We Speculate Running Application With Server Power Consumption Trace?

Yuanlong Li, *Member, IEEE*, Han Hu, *Member, IEEE*, Yonggang Wen, *Senior Member, IEEE*, and Jun Zhang *Fellow, IEEE*

***Abstract*—In this paper, we propose to detect the running applications in a server by classifying the observed power consumption series for the purpose of data center energy consumption monitoring and analysis. Time series classification problem has been extensively studied with various distance measurements developed; also recently the deep learning-based sequence models have been proved to be promising. In this paper, we propose a novel distance measurement and build a time series classification algorithm hybridizing nearest neighbor and long short term memory (LSTM) neural network. More specifically, first we propose a new distance measurement termed as local time warping (LTW), which utilizes a user-specified index set for local warping, and is designed to be noncommutative and nondynamic programming. Second, we hybridize the 1-nearest neighbor (1NN)-LTW and LSTM together. In particular, we combine the prediction probability vector of 1NN-LTW and LSTM to determine the label of the test cases. Finally, using the power consumption data from a real data center, we show that the proposed LTW can improve the classification accuracy of dynamic time warping (DTW) from about 84% to 90%. Our experimental results prove that the proposed LTW is competitive on our data set compared with existed DTW variants and its noncommutative feature is indeed beneficial. We also test a linear version of LTW and find out that it can perform similar to state-of-the-art DTW-based method while it runs as fast as the linear runtime lower bound methods like LB_Keogh for our problem. With the hybrid algorithm, for the power series classification task we achieve an accuracy up to about 93%. Our research can inspire more studies on time series distance measurement and the hybrid of the deep learning models with other traditional models.**

***Index Terms*—Long short term memory (LSTM), recurrent neural network (RNN), time series classification, time warping.**

## I. Introduction

NOWADAYS, a growing number of data centers have been built to support complicated computation and massive storage required by various blooming applications [1]. Each data center is typically equipped with hundreds of thousands servers and requires many mega-watts electricity to power its hosted servers and the auxiliary facilities [2]. An essential problem is to monitor such a large amount of servers for energy saving and maintaining the business continuity, which is critical to build green data center [3] and improve energy efficiency [4].

Monitoring technologies [5] can be divided into two categories: 1) intrusive and 2) nonintrusive. Intrusive technologies require the install of certain monitoring software which requires the administration role of the system. Compared to the intrusive methods, nonintrusive methods are more flexible, which only require limited data for the monitoring and analysis.

In this paper, for the purpose of energy consumption monitoring, we propose to detect the running program in a server by analyzing the observed power consumption series. The power data can be measured without the administration right of the server, which can be useful in collecting the power related information of the servers for the purpose of energy consumption analysis. The proposed classification analysis can only gain the type of the running program, avoiding any possibility in accessing the privacy-related contents in the server.

The proposed program detecting problem falls into the field of time series classification. As a time series classification problem, the power data classification problem can be challenging as the power series collected in detection may be only a small piece of the whole power series of a program, with incomplete and limited information. For this problem, the key is to design an accurate and fast classification algorithm.

Currently, there are a few similar works on classifying signals (like the power consumption signals studied here) such as [6]–[8]. However, the technologies applied in these literature are based on common spectral or statistical features with classifiers such as nearest neighbor or neural network. In a more general aspect, the time series classification problem has been extensively studied [9], among which the most popular method is 1-nearest neighbor (1NN) with dynamic time warping (DTW). The major research line in time series classification has been the developing of various DTW-based distance measurements (variants such as [10] and enhancers [11]); yet we find that even though these measurements can be better than the original DTW by certain degree for certain cases, these variants all have been designed to incorporate the dynamic programming idea of DTW (except some lower

Manuscript received September 23, 2016; revised January 23, 2017 and March 22, 2017; accepted April 29, 2017. Date of publication May 24, 2017; date of current version April 13, 2018. This work was supported in part by the Singapore EMA under Grant EIRP02, in part by the Singapore IMDA under Grant GDCR01, and in part by the National Natural Science Foundation of China under Grant 61332002. This paper was recommended by Associate Editor H. Wang. *(Corresponding author: Han Hu.)*

Y. Li, H. Hu, and Y. Wen are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: liyuanl@ntu.edu.sg; hhu@ntu.edu.sg; ygwen@ntu.edu.sg).

J. Zhang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: junzhang@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2017.2703941

bound methods like LB_Keogh [12]) and all designed to be commutative. Another line of research has also become notable recently, i.e., the long short term memory (LSTM) neural network [13], [14], which shows great modeling ability for sequential data. In this paper, we propose a novel classifier with much higher accuracy and based on the great efforts in the current literature.

In this paper, first, we propose a local time warping (LTW) time series distance measurement, which is a light weight DTW variant that does not need the dynamic programming procedure and is designed to be noncommutative. LTW can be set to a linear runtime algorithm which can perform almost as good as the DTW on our data set. Second, instead of further enhancing the distance measurement which can be much more complicated and time consuming, we look into a less expensive solution, which is to develop a hybrid algorithm of the 1NN with LTW (1NN-LTW) and the recent deep learning model for time sequential modeling. To do so, we first utilize the state-of-the-art sequential data modeling neural network LSTM to classify the power series. Then we propose a new hybrid algorithm of the proposed 1NN-LTW and the LSTM. This paper shows that both 1NN-LTW and LSTM can outperform the 1NN-DTW with similar accuracy; however, these two algorithms have their unique different natures and the accurately classified samples of these two algorithms have significant differences. The hybrid algorithm of the two classifiers, termed as LSTM/LTW, improves the classification accuracy further easily.

The main contributions of this paper are summarized as follows.

1) We propose a new distance measurement LTW. LTW has two unique features which are different from the existing DTW variants: a) LTW is based on simple "local warping", no dynamic programming procedure is needed and b) LTW is noncommutative and is flexible for the nearest neighbor classifier for the time series classification problem. Our experimental results show that for our problem, the proposed LTW can perform better than DTW and its several different variants. Also our experiment shows that the noncommutative feature of LTW is beneficial. Furthermore, the linear version of LTW can perform almost as good as DTW on our data set. These results show that for certain cases, a light weight local warping distance measurement (such as the LTW) may be good enough for the classification task; however, this does not mean that the proposed LTW can work for all kinds of time series data sets.
2) For the first time, we develop a hybrid algorithm of 1NN-DTW and LSTM termed as LSTM/LTW. The hybrid algorithm is based on a well trained LSTM neural network. Although the training procedure of the LSTM can be time consuming, the classification process can be fast in testing with the LTW distance.
3) Numerical experiments show that for the power data classification problem, with the LTW distance measurement, the accuracy of the 1NN-LTW classifier can be improved from about 84% to about 90% compared to the 1NN-DTW. With the hybrid algorithm LSTM/LTW, we achieve the power consumption series classification accuracy up to about 93%, which proves that using the power consumption series to detect the type of the running programs in a server can be very accurate.

The remainder of this paper is organized as follows. In Section II, we briefly introduce the state-of-the-art time series classification algorithms. In Section III, we introduce the experimental data collection design and some preliminary analysis on the data. In Section IV, we introduce the new proposed algorithm and in Section V we show the numerical evaluation results and the analysis. In Section VI, we conclude the whole paper and introduce the future works.

## II. Related Works

The power data classification problem studied in this paper can be taken as a time series classification problem, which has been studied extensively for the past decades. For this problem, common classifiers like support vector machine (SVM), $k$-nearest neighbor (KNN) with Euclidean distance have been proved to be noncompetitive to the DTW distance measurement-based method like 1NN-DTW [15]. Recently there have been a lot of new methods which have been proved to be as competitive as 1NN-DTW. On one hand, there are many non-neural network-based methods like shapelet-based method, dictionary-based methods, interval-based methods, and ensembles of these methods. We will brief these methods below. On the other hand, recently with the fast development of deep learning [16] and its applications [17]–[19], LSTM neural network has also been proved to hold high modeling ability for sequential data. In the following we will briefly introduce LSTM.

### A. Non-Neural Network Approaches

The most popular non-neural network time series classifiers are nearest neighbor-based method with various different distance measurements. The most popular method is the 1NN-DTW, which is a special KNN classifier with $k = 1$ and a special DTW distance measurement. For the 1NN classifier, the common standard procedure to label of a test sample given a set of training samples is as follows. First, the distances of the test sample to all the training samples are computed; then the training sample that has the smallest distance to the test sample is chosen and its label is assigned to the test sample as the classification result. In the above procedure, the key is to utilize a proper distance measurement. For 1NN-DTW, the DTW distance is used, which has superior performance for time series data.

The DTW calculates the distance of two sequences $\mathbf{x}$ and $\mathbf{y}$ in a manner of finding the best match between them, as shown in Fig. 1. The idea is that sequential data often contain similar fluctuation patterns, however, a same pattern, when existed in different sequences as subsequences, may be stretched, shrank or delayed in the time axis. In this case, the DTW distance measurement aims to warp the time axis nonlinearly and finds the best match between the two samples such that when a same pattern exists in both sequences, the distance is smaller.

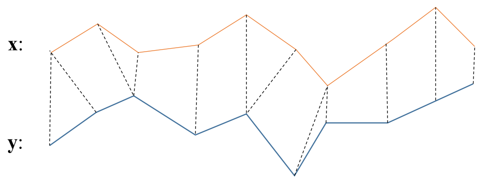

Fig. 1. Illustration of the DTW distance measurement. In computing the DTW distance between samples **x** and **y**, the DTW algorithm finds the best match (shown by the dashed lines) between the two series at different time steps.

Mathematically, the DTW distance is computed by the following dynamic programming process. Denote $D(i,j)$ as the DTW distance between subsequences $x[1:j]$ and $y[1:j]$, then the DTW distance between **x** and **y** can be computed by the dynamic programming process with the following iterative equation:

$$D(i,j) = \min\{D(i-1,j-1), D(i-1,j), D(i,j-1)\} + \left|x_i - y_j\right|. \quad (1)$$

The time complexity to compute the DTW distance is $O(nm)$, where $n$ and $m$ are the length of **x** and **y**, respectively. The DTW distance measurement actually realign the time step index pairs in the computing of the distance. In practice, usually a threshold $w$ is used to restrict the index offset in the alignment, which can be critical to the classification results [20]. Also there are many study [21] working on accelerating the computing speed of DTW, which results in the fast DTW that can be computed in linear time of the length of the sequences. In this paper, we follows the idea of DTW but propose a new distance measurement, which can be computed with a local warping index set without a dynamic programming process and has a special noncommunicative nature that can be helpful.

There are many DTW variants proposed. We name only a few here for the space constraint; one can refer to [9] for a more complete review and comparison of the existing methods. A popular line of research is the derivative DTW (DDTW) proposed by Keogh and Pazzani [22], which utilizes derivative of the raw series in computing the distance. Batista *et al.* [11] proposed the complexity invariant distance (CID), which is a weight modifier that can be used to enhance any kind of distance measurement, and is proved to be very useful when using with DTW. In CID, the first order derivative of the raw series is used in the computation of the modifier. The idea of using the derivative series is further studied in [23], in which the distance is computed based on the combination of the raw series and the derivative series. There are also other DTW variants, such as the move-split-merge [10] method, which introduces move and split operation to dynamic warping in DTW.

Besides DTW-based method, there have been many new different methods for time series classification which look into the pattern of the time series. For example, Ye and Keogh *et al.* [24] proposed the Shapelet-based method, which utilizes the subsequences that can differentiate different classes to do the classification; recently similar work is done in [25]. Lin *et al.* [26] proposed the dictionary-based method, which transforms the series into discrete words in a dictionary and then do the classification. Deng *et al.* [27] proposed interval-based classifiers to extract the feature from intervals in each time series for the classification. In this paper, we focus on DTW-based methods and will not compare with these methods, as proved by the through comparison experiments done in [9], these methods perform similar with DTW-based methods.

Beside the above listed works, ensemble method for time series classification is another popular research direction [28], [29]. Ensemble method combines multiple different classifiers to create a new classifier that can be better than any single classifier. Recently, different ensemble methods based on the above listed classifiers have been proposed and have shown promising performance [29]. However, we have not seen any work on ensemble method of the above methods and deep neural network like LSTM. In this paper, we propose a simple hybrid algorithm of a nearest neighbor classifier and LSTM. The LSTM neural network used in this paper is introduced below.

### B. LSTM

LSTM is first proposed by Mikolov *et al.* [30] as an upgrade of the recurrent neural network (RNN). RNN is used to handle sequential data with a special calculation process following the time step increment, while traditional neural network simply treats the sequence as a plain vector. With such nature, RNN is suitable for modeling sequential data. However, it suffers from a problem called diminishing gradient, which is caused by the iterative process on the time axis and makes the gradient used in the training process extremely small and causes training failure. To solve the problem, the LSTM is proposed and it utilizes a memory core to avoid the diminishing gradient. The details of the LSTM neural network will be introduced in Section IV.

LSTM has shown great modeling power for sequential data and has been successfully applied in various machine learning fields like image recognition [31], [32] (a simpler efficient sequence modeling neural network than LSTM is proposed in this paper), natural language process [33], video analysis [34], etc. It is also noted that LSTM can be both discriminative and generative. By discriminative, LSTM can be used for classification tasks while by generative, LSTM can be used to generate similar sequences like the training samples [35]. In this paper, we utilize the discriminative ability of LSTM for our power data classification task.

## III. Power Series Data Collection and Preliminary Analysis

In this section, we present the power series data we collected followed by some preliminary analysis on the data. We will detail the simulation design rules for the data collection and the data samples collected with some pretreatment. The proposed preliminary analysis includes data visualization with different dimension reduction methods, classification results with some canonical classifiers, and feature study.

TABLE I
COLLECTED POWER SEQUENCES OF DIFFERENT PROGRAMS. MAPREDUCE AND WEB SERVER ARE THE TWO MAJOR CLASSES, WHILE IN MAPREDUCE THERE ARE MANY SUBCLASSES. IN TOTAL THERE ARE 13 CLASSES

| Program | | | | Number of sequences | Class label |
|---|---|---|---|---|---|
| MapReduce | Spark | Word Count | | 100 | 0 |
| | | Sorting | | 100 | 1 |
| | | PI | | 100 | 2 |
| | | MLlib | CrossValidator | 40 | 3 |
| | | | Kmean | 40 | 4 |
| | | | LR | 40 | 5 |
| | | | SVM | 40 | 6 |
| | | | Cosine similarity | 40 | 7 |
| | | | PCA | 40 | 8 |
| | Hadoop | Word Count | | 100 | 9 |
| | | Sorting | | 100 | 10 |
| | | PI | | 100 | 11 |
| Web server data | | | | 40 | 12 |

## A. Power Series Data Collection

We first introduce the designing rules of the simulation for data set collection. As a data-driven study on the power series classification methodology, we need to collect a set of sample power data. The data collection should be designed carefully to make sure that the classification problem is neither trivial nor impossible to accomplish. In this sense, our guiding line for power data collection is to collect "different" and "similar" power series. By different, the power series must be generated by different programs. By similar, the different programs can have some similar features so that the classification algorithms need to be really discriminative.

Follow the above guideline, we collected in total 13 classes of power data as shown in Table I (for convenience they are labeled as $0, 1, \ldots, 12$, respectively). These data fall into two major categories.

1) *Web Server Power Data:* Usually fluctuate in a continuous pattern.
2) *Spark/Hadoop MapReduce Programs:* Usually show stage-pattern, e.g., the map stage and the reduce stage.

For the Hadoop/Spark category, we test different programs on these platforms, some are the same for both platforms, such as the "word count" program; some only exist on one platform, for example, the "MLlib" programs on the Spark platform. With such design, we can achieve the proposed different and similar design goal.

Note that the collected data series are of different lengths as the running duration can vary among different programs. Although classification methods like 1NN-DTW can deal with power series of different lengths, to apply other canonical methods, in the following we cut the collected series into fixed length sequences. It is also reasonable to label subsequences instead of the complete power sequences of the programs as in a blind test, we have no information about the start/end point of a program. The detailed cutting method we utilize here is as shown below.

The goal is to cut the power sequences into length 200 samples. To do so, first we discard sequences with length smaller than 200 time slots (time unit: 3 s). The left number of sequences for each class is: [77, 31, 30, 35, 28, 7, 40, 14, 5, 100, 58, 36, 40]. Although some power data are discarded, the total duration of the left sequences is about 199 h and with the time unit being 3 s, the amount of the data are still adequate for the study. Then these sequences are further cut into length 200 subsequences in the following way. For each sequence $\mathbf{q}$ of length $n$, we cut it into multiple sequences $q[0 : 200], q[50 : 250], \ldots, q[(n - 200) : n]$. We obtain 3200 test sequences in this cutting procedure, which are used as the power data in our classification study. Note that these sequences are overlapped, as indicated by the cutting method.

Furthermore, for the purpose of multifold tests, we divide these samples into fivefolds $F_0 - F_4$. Note that to avoid the overlapping of the training data and the test data, the fold partition is done before the sequence cutting. For each fold of test, we use $F_i, F_{(i+1)\%5}$ as the test data, and the left folds as the training data.

## B. Preliminary Analysis

We do some preliminary analysis on the pretreated data. The following analysis are meant to provide a basic understanding of the power data in view of classification.

*1) Basic Characteristic Analysis Based on Visualization:* We use various dimension reduction methods to visualize the data, which can help to identify if the power series can be successfully classified to a certain degree. We utilize eight different dimension reduction methods with scikit-learn [36] and project the original fixed length power sequences into a 2-D space. These dimension reduction methods are PCA, LDA, LLE, modified LLE, isomap, MDS, spectral embedding, and t-SNE, which are widely adopted dimension reduction methods. The 2-D codes of the power data generated by these methods are shown in Fig. 2. We use different colors to show samples from different classes.

From Fig. 2, we can observe that the power series data are not easy to distinguish after the dimension reduction. This may be due to the short length (2 here) of the embedding code; however, it still shows that the power series classification task cannot be easily done.

*2) Tackling the Classification Problem With the Canonical Classifiers:* We test some canonical classifiers to tackle the power series classification problem. The canonical classifiers tested here are listed as follows: nearest neighbors, linear SVM, RBF SVM, decision tree, random forest, AdaBoost, naive Bayes, LDA, and QDA [36]. Parameter settings for these classifiers are tuned manually. The classification results of these methods are shown in Table II.

From the results, we can observe that for a 13-class classification problem, the highest accuracy achieved by these methods are about 60% (by random forest). The classification accuracy is not promising (when compared to the 1NN-DTW shown below), which actually proves that our power series labeling problem is a typical time series classification problem, as stated in [15], for such problem, canonical Euclidean distance metric-based classifiers cannot achieve good results usually.

*3) Feature-Based Classification Study:* In general, as a signal classification problem, the power series labeling problem can be solved by first extracting certain features from the raw power series and then carry out the classification with these

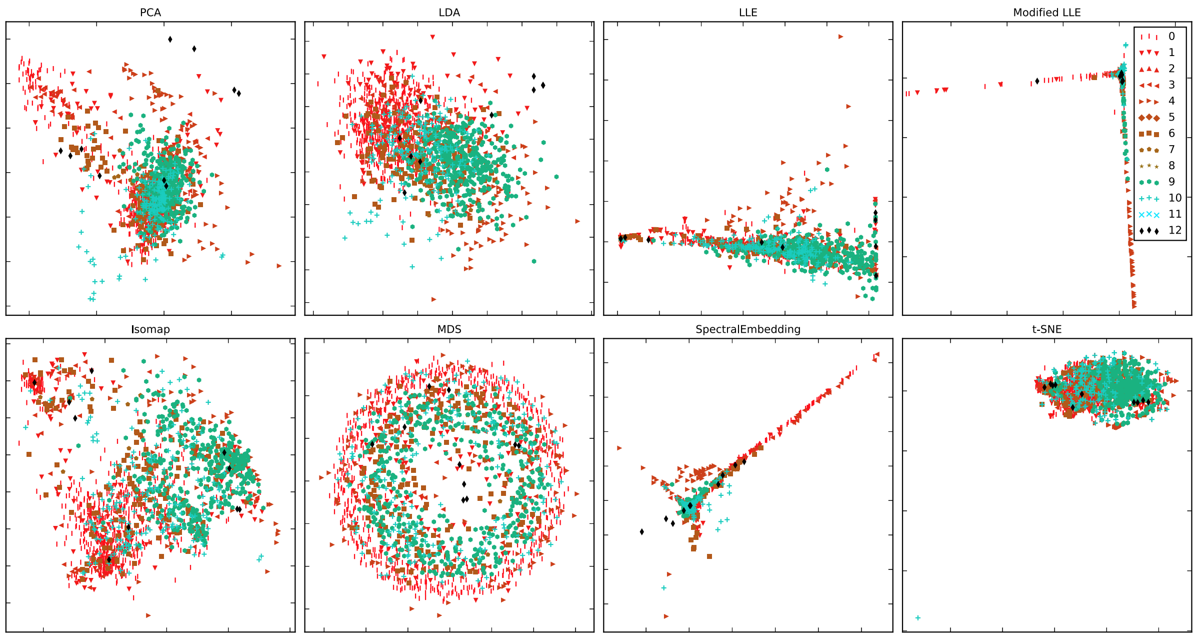

Fig. 2. Projection results with different manifold learning methods. The power series samples (200-dimensional data) are projected into 2-D space for visualization. The results show that the power series are difficult to discriminate.

TABLE II
CLASSIFICATION RESULTS OF THE CANONICAL CLASSIFIERS. THESE CLASSIFIERS CAN ACHIEVE ACCURACY UP TO ABOUT 60%

| Test case | Nearest Neighbours | Linear SVM | RBF SVM | Decision Tree | Random Forest | AdaBoost | Naive Bayes | LDA | QDA |
|---|---|---|---|---|---|---|---|---|---|
| Fold 0 | 0.4346 | 0.5187 | 0.3175 | 0.5747 | 0.5891 | 0.4593 | 0.4406 | 0.4380 | 0.4092 |
| Fold 1 | 0.4427 | 0.5105 | 0.3063 | 0.5607 | 0.6050 | 0.4192 | 0.4360 | 0.4377 | 0.4276 |
| Fold 2 | 0.4420 | 0.5071 | 0.3283 | 0.5901 | 0.6133 | 0.4121 | 0.4016 | 0.4256 | 0.4248 |
| Fold 3 | 0.4475 | 0.5004 | 0.3094 | 0.5835 | 0.6378 | 0.3813 | 0.4123 | 0.3707 | 0.4574 |
| Fold 4 | 0.4673 | 0.5383 | 0.3301 | 0.5610 | 0.6174 | 0.3834 | 0.4278 | 0.4407 | 0.4786 |

TABLE III
CLASSIFICATION RESULTS OF THE 1NN-DTW WITH THE ORIGINAL SERIES AND WITH THE DFT FEATURE

| Test case | 1NN-DTW with original series | 1NN-DTW with DFT feature |
|---|---|---|
| Fold 0 | 0.8548 | 0.7122 |
| Fold 1 | 0.8393 | 0.7029 |
| Fold 2 | 0.8369 | 0.6761 |
| Fold 3 | 0.8329 | 0.6998 |
| Fold 4 | 0.8514 | 0.6885 |

features. In this section, we study such possibility and test power series classification with the DFT [37] feature of the original power sequences. With DFT, each power sequence can be transformed into the spectrum space resulting a new representation. The spectrum representation can be aligned as a vector as the input to the classifiers. We test the classification result of 1NN-DTW with the raw data compared to with the DFT feature. The classification results are shown in Table III. Note that for the 1NN-DTW, the maximum offset $r$ is set to 15% of the sample length, which is manually tuned in the experiment.

From Table III, we can observe that the DFT features are not helping. The reason is that classification with the original data can maximize the information used in classification, while the DFT feature is less informative.

To summary, we find that the power series classification problem is not easy to tackle especially with the canonical classifiers and with some common used features. In the following, we will propose a new distance measurement inspired from DTW and combine it with the state-of-the-art sequence modeling neural network LSTM.

## IV. PROPOSED POWER SERIES CLASSIFICATION ALGORITHM

In this section, we present the proposed new power series classification algorithm which hybridizes a nearest neighbor classifier with a novel distance measurement and an LSTM classifier. In the following, we first introduce the two components, respectively, and then present the hybrid algorithm.

### A. Nearest Neighbor With the Local Time Warping

We propose a new classifier which utilize a novel distance measurement to compute the distance between two sequences which we termed as LTW as its warping computation for each time step is done in a local window without a dynamic programming procedure like DTW. The LTW is developed to replace the DTW distance measurement in the 1NN-DTW classifier.

**Algorithm 1** $\text{LTW}_G(\mathbf{x}, \mathbf{y})$

```
1:  $n = length(\mathbf{x})$
2:  $r = 0$
3:  $\mathbf{d}_0 = abs(\mathbf{x} - \mathbf{y})$
4:  $\mathbf{d}_1 = abs(\mathbf{x}[1, .., n-1] - \mathbf{y}[2, \ldots, n])$
5:  for $k \in G$ do
6:      $\mathbf{d}_k = abs(\mathbf{x}[1, .., n-k] - \mathbf{y}[k+1, \ldots, n])$
7:      $d_k = \sum_{i=k+1}^{n-k} min(\mathbf{d}_0[i], \mathbf{d}_1[i], \mathbf{d}_k[i])$
8:      $r = r + d_k$
9:  end for
10: Return $r$.
```

The idea behind LTW is as follows. Comparing the algorithms of DTW and the Euclidean distance, the major difference in between is that there are a lot of "min" operators in DTW. Such min operator actually is the key to the "warping" map between the two time series. Despite the warping operation, DTW utilizes a dynamic programming procedure to optimize the mapping. Note that dynamic programming is slow and it is not directly optimizing the classification accuracy. In this case, what if we do not use the dynamic programming procedure? We may try some low cost warping operations; is it possible that such a distance measurement can be as good as DTW? Here we propose the LTW to answer this question. Also, note that the DTW is computed by a beautiful symmetric formula which makes it commutative for the two time series in computing the distance. What if we do not need the distance measurement to be commutative? Can it be better with the noncommutative feature? Our proposed LTW will also answer this problem. The detailed design is shown below.

The LTW distance measurement is computed in the following way. Suppose we have two sequences $\mathbf{x}$ and $\mathbf{y}$, both of length $n$. We define the LTW distance between $\mathbf{x}$ and $\mathbf{y}$ as

$$d_k^{\text{LTW}}(\mathbf{x}, \mathbf{y}) = \sum_{i=k+1}^{n-k} \min(|x_i - y_i|, |x_i - y_{i+1}|, |x_i - y_{i+k}|) \quad (2)$$

$$\text{LTW}_G(\mathbf{x}, \mathbf{y}) = \sum_{k \in G} d_k^{\text{LTW}}(x, y). \quad (3)$$

As shown in (3), LTW works in the following manner. In computing the distance between $\mathbf{x}$ and $\mathbf{y}$ (when we want to find a nearest neighbor of $\mathbf{x}$), we set $\mathbf{x}$ as the base sequence and test the similarity of $\mathbf{y}$ to $\mathbf{x}$ in the following way: with a warping index set $G$, for $k \in G$, for time step $i$ in $\mathbf{x}$, we compute the minimum absolute distance between $x_i$ and one of $y_i, y_{i+1}, y_{i+k}$; then we add these distance measures for $i = k+1, \ldots, n-k$ and for $k \in G$ up, which is the LTW distance from $\mathbf{y}$ to $\mathbf{x}$ with warping index $G$. Note that (2) is a linear algorithm (only three items to compare no matter how large $k$ is). Detailed pseudo code to compute $\text{LTW}_G(\mathbf{x}, \mathbf{y})$ is shown in Algorithm 1.

Note that the $\text{LTW}_G(\mathbf{x}, \mathbf{y})$ distance is noncommutative, which means that $\text{LTW}_G(\mathbf{x}, \mathbf{y}) \neq \text{LTW}_G(\mathbf{y}, \mathbf{x})$ can be true. We use $\text{LTW}_G(\mathbf{x}, \mathbf{y})$ to compute the nearest neighbor of sequence $\mathbf{x}$, in a sense that to find the best match of $\mathbf{x}$ among the other samples such as $\mathbf{y}$. For comparison, the DTW distance is obviously commutative. The noncommutative feature of LTW can be beneficial, as our target is to find the nearest neighbor for each $\mathbf{x}$. A noncommutative distance measurement is enough to serve the purpose and can provide more flexibility by enforcing less constraints to the distance measurement.

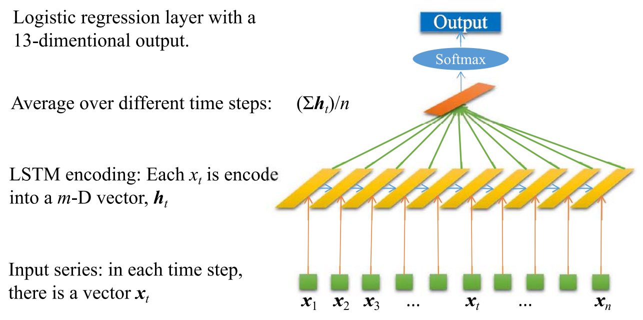
Logistic regression layer with a 13-dimentional output.
Output
Softmax
Average over different time steps: $(\Sigma \boldsymbol{h}_t)/n$
LSTM encoding: Each $x_t$ is encode into a $m$-D vector, $\boldsymbol{h}_t$
Input series: in each time step, there is a vector $\boldsymbol{x}_t$
$\boldsymbol{x}_1$ $\boldsymbol{x}_2$ $\boldsymbol{x}_3$ ... $\boldsymbol{x}_t$ ... $\boldsymbol{x}_n$

Fig. 3. Illustration of the LSTM neural network. It contains three layers: the input layer, the LSTM layer, and the logistic regression layer. The output is a 13-dimensional vector which denotes the probability of the sample belonging to each class.

## *B. Long Short Term Memory Neural Network*

We utilize the LSTM classifier following [38] for our power series classification problem. The LSTM neural network consists of an input layer, an LSTM layer and a logistic regression layer as depicted in Fig. 3. The three layers function in the following way, respectively.

1) *Input Layer:* The input data sample, which is a length $n$ vector $\mathbf{x}$, is first discretized into range $[0, S]$. Such an operation is a smoothing operation to the original power series, which can affect the performance of the LSTM. Then each time step $x_t, t = 1, \ldots, n$ is enriched into a $m$-dimensional vector $\mathbf{x}_t$ which can ease the following computation, i.e., $\mathbf{x}_t = x_t \cdot \mathbf{e}$, where $\mathbf{e}$ is a $m$-dimensional vector with all entries equal to 1. After the above process, the new sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ is used as the input to the LSTM layer.
2) *LSTM Layer:* The LSTM layer contains $n$ LSTM node, where each LSTM node $t$ can output an $m$ dimensional code $\mathbf{h}_t$. The operation inside the LSTM node is shown below. First, for each time step $\mathbf{x}_t$, the LSTM node needs to compute a new state denoted by $\mathbf{C}_t$. To compute $\mathbf{C}_t$, a candidate state $\mathbf{C}'_t$ is first computed as

$$\mathbf{C}'_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c). \quad (4)$$

Then two gates, an input gate $i_t$ and a forget gate $f_t$ are computed to update the new state

$$i_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) \quad (5)$$

$$f_t = \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f\right). \quad (6)$$

Then the new state of the LSTM node is computed as

$$\mathbf{C}_t = i_t \cdot \mathbf{C}_t + f_t \cdot \mathbf{C}'_t. \quad (7)$$

With the node state, to further compute the output, an output gate is first computed as

$$o_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o). \quad (8)$$

**Algorithm 2** LSTM Training Procedure

```
1: Divide the training data into a training set and a validation set. Partition the training data set into batches.
2: Set $Epoch = 0$, best validation error $V_{best} = \infty$; denoting $\theta$ as the set of all control parameters of the LSTM; randomly initialize $\theta$; set best LSTM parameter setting $\theta_{best} = \emptyset$; initialize $MaxEpoch$.
3: //Training process:
4: while $Epoch < MaxEpoch$ do
5:     for Each batch of the training data do
6:         Compute the loss function according to (13).
7:         Update $\theta$ to reduce the loss.
8:     end for
9:     Compute the loss on the validation set, termed as $V_c$.
10:    if $V_c < V_{best}$ then
11:        $V_{best} \leftarrow V_c$.
12:        $\theta_{best} \leftarrow \theta$.
13:    end if
14:    $Epoch = Epoch + 1$.
15: end while
16: Output: The LSTM network with the best parameter setting $\theta_{best}$.
```

Finally, the output of an LSTM node $t$ is computed as

$$\mathbf{h}_t = o_t \cdot \tanh(\mathbf{C}_t). \tag{9}$$

The output of all LSTM nodes are then added together as the output of the LSTM layer

$$\mathbf{h} = \sum_{t=1}^{n} \mathbf{h}_t. \tag{10}$$

3) *Logistic Regression Layer:* In this layer the output of the LSTM layer is used to compute the label of the test sample in the following way. First, we use the *softmax* [39] function to compute the probability vector $\mathbf{P}$ with its each entry representing the probability of the test sample belonging to a class

$$\mathbf{P} = \mathrm{softmax}(Wh + b). \tag{11}$$

Then the prediction $y_{\mathrm{pred}}$ is the class which achieves the largest probability

$$y_{\mathrm{pred}} = \mathrm{argmax}_i(\mathbf{P}). \tag{12}$$

To train the LSTM classifier, the loss function is defined as the negative log-likelihood function with the label of the training data $\mathbf{y}$

$$-L(\theta, D) = -\sum_{i=0}^{|D|-1} \log\Big(P_{y^{(i)}} \big| \mathbf{x}^{(i)}, \theta\Big) \tag{13}$$

where $\theta$ is the set of all the weight and bias parameters in the LSTM neural network (which are adjusted in the training process) and $D$ is a batch of training samples. Size of $D$ can be important for the performance of the classifier. The detailed training algorithm is shown in Algorithm 2.

**Algorithm 3** Hybridization of 1NN-LTW and LSTM

```
1: Train $C_{LSTM}$ with the training data according to Algorithm 2.
2: For test time series $\mathbf{x}$, compute the probability vector $\mathbf{p}_{LTW}(\mathbf{x})$ according to Algorithm 1 and $\mathbf{p}_{LSTM}(\mathbf{x})$ with the well trained $C_{LSTM}$.
3: Classify the test time series $\mathbf{x}$ according to (15) and get the label $l_{hybrid}(\mathbf{x})$.
4: Output: $l_{hybrid}(\mathbf{x})$, as the predicted label for sample $\mathbf{x}$.
```

### C. Hybridization of LSTM and 1NN-LTW

In this section, we propose to combine the 1NN-LTW classifier and the LSTM classifier. The underlying rational is that both classifier can achieve high classification accuracy for our problem but in very different manners: the 1NN-LTW is a nearest neighbor classifier, which is a data-based classifier without a training process; while LSTM is a training-based classifier in which the training data are first used to build a model and then the model is used to classify the test data. In our experiments, both classifiers can perform promisingly individually; however, our numerical simulation shows that the accurately classified samples by the two classifiers have significant differences. In such sense, we propose to combine the two algorithms to construct a even stronger classifier.

The hybrid algorithm is designed in the following way. Considering that in practice, the training of LSTM and the computing of the distance matrix for 1NN-LTW can be both time consuming, the hybrid algorithm is designed to be as simple as possible. We first obtain the two individual classifiers $C_{\mathrm{LTW}}$ (the nearest neighbor classifier with CID enhanced LTW) and $C_{\mathrm{LSTM}}$ (the trained LSTM classifier). For each classifier, we obtain the probability vector when classifying some test time series $x$: $\mathbf{p}_{\mathrm{LTW}}(\mathbf{x})$ and $\mathbf{p}_{\mathrm{LSTM}}(\mathbf{x})$, where $\mathbf{p}_{\mathrm{LTW}}(\mathbf{x})$ is defined as

$$\mathbf{p}_{\mathrm{LTW}}(\mathbf{x}) = \frac{\sum_{i=1}^{m}(m-i)\mathbf{v}_i}{\Delta} \tag{14}$$

where $\mathbf{v}_i$ is a all zero vector except its value at the index of the class of the $i$th neighbor obtained from $C_{\mathrm{LTW}}$ is to be 1. $\Delta$ is a normalization vector which is used to make sure that the summation of the obtained probability vector equal to 1. $\mathbf{p}_{\mathrm{LSTM}}(x)$ is obtained in (11).

With this two probability vector, we simply add them up and the test series will be classified to be the class index with the maximum probability, that is

$$C_{\mathrm{LSTM/LTW}}(\mathbf{x}) = \mathrm{argmax}(\mathbf{p}_{\mathrm{LTW}}(\mathbf{x}) + \mathbf{p}_{\mathrm{LSTM}}(\mathbf{x})). \tag{15}$$

The detailed algorithm is shown in Algorithm 3.

## V. Numerical Evaluation and Analysis

In this section, we present the experimental results of the above proposed algorithms and the analysis. We first conduct experiments to investigate the proposed LTW and compared it with various variants of DTW. Then we compare the classification accuracy of the proposed 1NN-LTW, LSTM, and their hybrid algorithm LSTM/LTW with the baseline algorithm 1NN-DTW. Test data and codes for the LTW tests are available at https://www.dropbox.com/sh/9iu6xg5eskq90g8/AABz8kaFOUWNtBa5XRPg6D9ua?dl=0. In presenting the classification results, for convenience, we will simply use test fold $i$ to denote the test with test samples in $F_i$ and $F_{(i+1)\%5}$.

## *A. Experimental Study on LTW*

In this section, we conduct experiments to prove that the proposed LTW is indeed a different distance measurement from the existed DTW variants, and prove that it works better or nearly the same(for the linear version) to DTW and its various state-of-the-art variants. We will also prove that the noncommunicative feature is indeed beneficial. Note that we will not try to use massive experimental data to prove that the proposed LTW is superior than other DTW variants, which is indeed not true not only because of the No Free Launch theory, but also because these distance measurements are mostly designed in a way without a training objective to directly optimize the classification accuracy: for example, in DTW, the dynamic programming process can optimize the match; however, such optimization goal is different from the classification accuracy. In such sense, all these distance measurements can suffer from model bias and when they are applied to different data sets, their performance will definitely vary. As proved in [9], only ensemble-based methods can significantly outperform 1NN-DTW by more than 3% on different data sets. In this section, we will only conduct experiments on our data set to compare LTW with the DTW with Manhattan distance (DTWm), DTW with Euclidean distance (we will use this as the default DTW in this paper, as it has better performance), and DTW variants MSM, LB_Keogh, and the enhancer CID as these methods performs good in general as shown in [9]. In the following experiments, we conduct pairwise Wilcoxon signed rank test [40] to test whether a target algorithm performs significantly different to the other algorithms or not. We show the computed $T$ value, where $T = 0$ means the compared algorithms are significantly different to the target algorithm at significance level 0.05 in our fivefold test, otherwise not. Details are shown below.

First, we present the comparison of 1NN-LTW ($G = \{1, 2, \ldots, 10\}$) with 1NN-DTWm (warping window $w = 20$), 1NN-DTW (warping window $w = 20$), and MSM (with its threshold parameter $c = 1.0$). These parameter settings are manually tuned to achieve best performance in a threefold internal cross validation of the training data. We will show analysis on the affects of the warp set $G$ in Section V-D. The results are shown in Table IV. Clearly the performance of 1NN-LTW is better.

Second, we compare the fast linear LTW with $G = \{10\}$ with the linear runtime lower bound method LB_Keogh. We test LB_Keogh with window size $w = 5$ and $w = 10$, respectively. The reason we set the window size to 5 is that in this case the warping set has a size of 10, which is of the same size of 1NN-LTW ($G = \{1, 2, \ldots, 10\}$), such that we can show whether 1NN-LTW ($G = \{1, 2, \ldots, 10\}$) is different to LB_Keogh or not. For window size 10, we want to test LB_Keogh with the same largest warp index offset as LTW with $G = \{10\}$. The results are shown in Table V and LTW with $G = \{10\}$ outperforms LB_Keogh significantly on our data set. This proves that the fast linear version of LTW can still perform quite good and is different from the lower bound method LB_Keogh.

TABLE IV
CLASSIFICATION ACCURACY OF 1NN-LTW ($G = \{1, 2, \ldots, 10\}$) WITH 1NN-DTWM ($w = 20$), 1NN-DTW ($w = 20$), AND MSM ($c = 1.0$)

| Test case | 1NN-LTW({1,...,10}) | 1NN-DTWm | 1NN-DTW | MSM |
|---|---|---|---|---|
| Fold 0 | 0.8862 | 0.7988 | 0.8548 | 0.8294 |
| Fold 1 | 0.8854 | 0.7958 | 0.8393 | 0.8310 |
| Fold 2 | 0.8661 | 0.8220 | 0.8369 | 0.8699 |
| Fold 3 | 0.8809 | 0.8118 | 0.8329 | 0.8682 |
| Fold 4 | 0.8894 | 0.8168 | 0.8514 | 0.8515 |
| Wilcoxon T | Target | 0 | 0 | 0 |

TABLE V
COMPARISON OF 1NN-LTW ($G = \{10\}$) WITH 1NN-LB_KEOGH ($w = 5, 10$)

| | 1NN-LTW($G = \{10\}$) | LB_Keogh ($w = 5$) | LB_Keogh ($w = 10$) |
|---|---|---|---|
| Fold 0 | 0.8829 | 0.5866 | 0.4542 |
| Fold 1 | 0.8762 | 0.5791 | 0.4753 |
| Fold 2 | 0.8444 | 0.6081 | 0.4592 |
| Fold 3 | 0.8696 | 0.6018 | 0.4616 |
| Fold 4 | 0.8838 | 0.6053 | 0.4326 |
| Wilcoxon T | Target | 0 | 0 |

TABLE VI
CLASSIFICATION RESULTS WITH CID ENHANCED DISTANCE MEASUREMENT

| | 1NN-CID(DTW) | 1NN-CID(LTW{1,...,10}) | 1NN-CID(LTW{10}) |
|---|---|---|---|
| Fold 0 | 0.8829 | 0.9041 | 0.8837 |
| Fold 1 | 0.8854 | 0.8904 | 0.8912 |
| Fold 2 | 0.8833 | 0.8684 | 0.8579 |
| Fold 3 | 0.8950 | 0.9013 | 0.8887 |
| Fold 4 | 0.8999 | 0.9031 | 0.8846 |
| Wilcoxon T | 4 | Target | 1 |

Third, we present the classification results with the CID enhanced distance measurement. We show the results of CID(DTW) and CID(LTW) with $G = \{1, 2, \ldots, 10\}$ and $G = \{10\}$ in Table VI. Clearly, the CID can improve the performance of both DTW and LTW for our data set. With CID modifier, the LTW is still slightly better than DTW; however, the advantage of CID(LTW) over CID(DTW) becomes smaller, we believe that it is reasonable as the accuracy is upper bounded and it will be much more difficult to further improve the accuracy when the algorithm is already very accurate.

At last, we present the experimental results to show that the noncommutative feature of LTW is indeed beneficial. To do so, we implement a simple commutative version of LTW defined as

$$\mathrm{LTW}^{\mathrm{com}}(\mathbf{x}, \mathbf{y}) = \mathrm{LTW}(\mathbf{x}, \mathbf{y}) + \mathrm{LTW}(\mathbf{y}, \mathbf{x}). \qquad (16)$$

The experimental results compare the LTW with the $\mathrm{LTW}^{\mathrm{com}}$ is shown in Table VII. Clearly, LTW outperforms its commutative version significantly. This proves that the noncommutative feature of LTW is indeed beneficial.

## *B. Classification Accuracy Rate Comparison*

The fivefold classification accuracy results for the hybrid algorithm LSTM/LTW are shown in Table VIII, compared with the nonhybrid classifiers 1NN-DTW, 1NN-CID(LTW), LSTM, and a recently proposed ensemble classifier ensemble

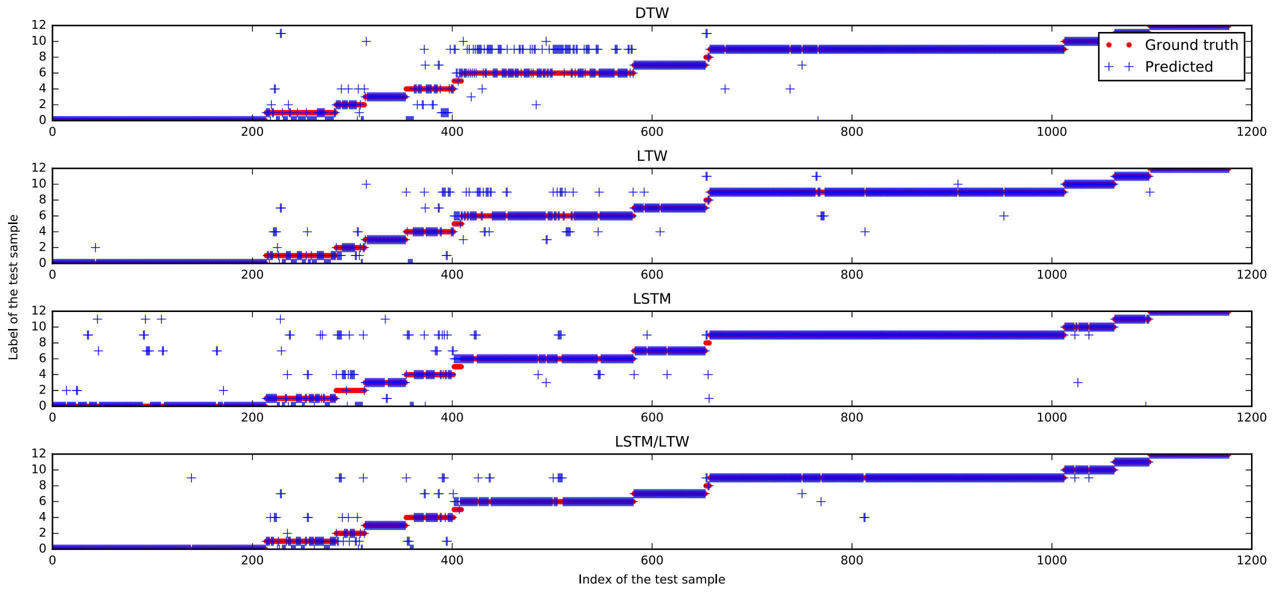

Fig. 4. Classification response for different algorithms on fold 0. The ground truth/predicted labels of the test samples are plotted against the index of the sample. Clearly the hybrid algorithm gains the advantage of both LSTM and 1NN-LTW.

TABLE VII
COMPARISON OF LTW AND THE LTW$^{com}$

| | 1NN-LTW | 1NN-$LTW^{com}$ |
|---|---|---|
| Fold 0 | 0.8862 | 0.7385 |
| Fold 1 | 0.8853 | 0.7029 |
| Fold 2 | 0.8661 | 0.7218 |
| Fold 3 | 0.8809 | 0.7294 |
| Fold 4 | 0.8894 | 0.7692 |
| Wilcoxon T | Target | 0 |

TABLE VIII
CLASSIFICATION ACCURACY OF 1NN-LTW, LSTM, AND THE HYBRID ALGORITHM LSTM/LTW COMPARED TO THE BASELINE ALGORITHM 1NN-DTW AND ENSEMBLE CLASSIFIER EETSC

| Test case | 1NN-DTW | 1NN-CID(LTW) | EETSC | LSTM | LSTM/LTW |
|---|---|---|---|---|---|
| Fold 0 | 0.8548 | 0.9040 | 0.8472 | 0.8791±0.0142 | 0.9265±0.0049 |
| Fold 1 | 0.8393 | 0.8903 | 0.8377 | 0.8834±0.0093 | 0.9130±0.0039 |
| Fold 2 | 0.8369 | 0.8683 | 0.8355 | 0.8621±0.0142 | 0.8909±0.0063 |
| Fold 3 | 0.8329 | 0.9013 | 0.8379 | 0.8729±0.0170 | 0.9151±0.0045 |
| Fold 4 | 0.8514 | 0.9031 | 0.8450 | 0.8762±0.0142 | 0.9239±0.0047 |
| Wilcoxon T | 0 | 0 | 0 | 0 | Target |

of elastic distance measurements for time series classification (EETSC) [28]. For the LSTM neural network, we set the maximum number of epoch (*MaxEpoch*) to 50. For some key parameters which can affect the performance of LSTM, we give a detailed discussion in the following parameter settings study. Results of LSTM and LSTM/LTW are based on 30 independent runs as the LSTM is trained by a stochastic algorithm. For EETSC, as it has four different ensemble methods, in the revised manuscript we only show the results of the method which achieves the best performance, which is the "Prop" method. Note that the code of EETSC is downloaded from the website provided by the authors.

From Table VIII, we can observe the following.

1) The proposed LSTM classifier shows similar accuracy compared to 1NN-LTW and it also outperforms 1NN-DTW on our data set.
2) The hybrid algorithm LSTM/LTW can achieve higher accuracy compared to 1NN-LTW and LSTM by an increment of about 3%, which proves that the hybrid algorithm can indeed improve the classification accuracy.
3) The performance of EETSC is not as good as the proposed hybrid LSTM/LTW, and it is also weaker than the CID(LTW). We noticed that the CID modifier is not used in EETSC, which may be the reason that it performs similarly to the 1NN-DTW and MSM.

For the first observation, we can see that LSTM, as a neural network, can significantly outperform the other canonical classifiers like SVM, which proves its strong modeling ability for sequential data. Note that a common neural network like multilayer perceptron cannot perform as good as LSTM. The performance of LSTM can be seriously affected by the training settings, which we will discuss below.

For the second observation, we can see that the improvement is small, which is reasonable as the baseline algorithms already achieve a high accuracy individually, making it difficult to achieve large improvement for the hybrid algorithm. The improvement caused by the hybrid algorithm will be shown clearly in the following detailed analysis.

## C. Analysis on the Accurately Classified Samples

In this section, we analyze the accurately classified samples of the power series and study the difference between different classifiers. In doing so we will be able to identify why and how the hybrid algorithm works.

Fig. 4 shows the predicted labels for the test samples in fold 0 of the 1NN-DTW, 1NN-LTW, LSTM and the hybrid algorithm LSTM/LTW. Fig. 5 shows the accurately classified samples for each class and for each algorithm. From Figs. 4 and 5, we can observe the following.

1) The proposed 1NN-LTW method performs similarly to 1NN-DTW, although 1NN-LTW can accurately predict

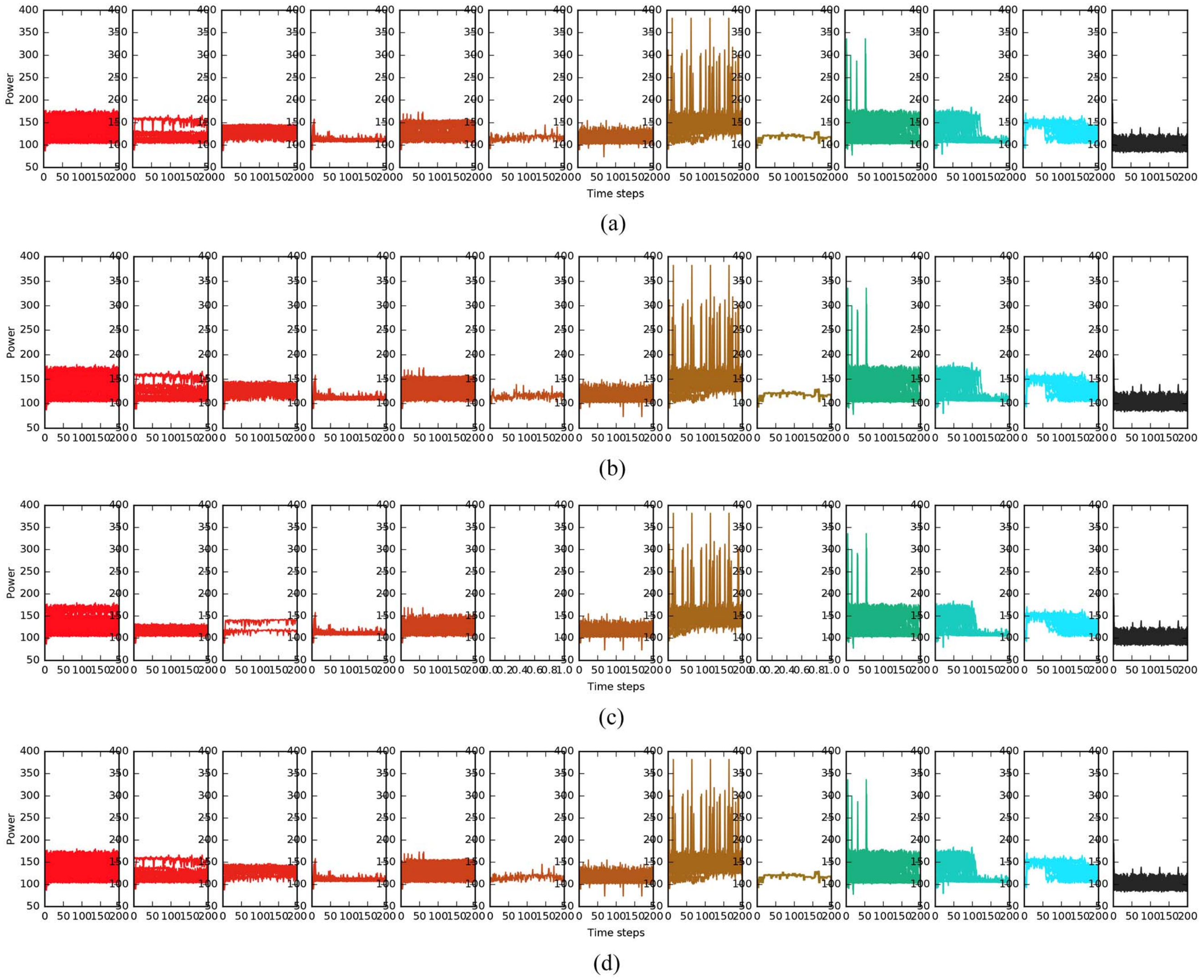

Fig. 5. Accurately classified samples for the different algorithms on fold 0. Samples from the same class are drawn in the same subfigure. (a) 1NN-DTW. (b) 1NN-LTW. (c) LSTM. (d) LSTM/LTW. LSTM cannot classify some classes while the hybrid algorithm gains the advantage of both LSTM and 1NN-LTW.

more test samples. This is reasonable as the two classifiers are both nearest-neighbor classifiers and they have similar measurement definition.

2) The proposed LSTM classifier shows certain degree of difference compared to the other two nearest neighbor classifiers. One example, the LSTM classifier cannot predict any test samples from the Spark-MLlib-LR (class label 5) and the Spark-MLlib-PCA (class label 8) classes, while both 1NN-DTW and 1NN-LTW can; however, LSTM performs better than the other algorithms on classes Spark-MLlib-SVM (class label 6) and Hadoop-WordCount (class label 9).
3) The proposed LSTM/LTW classifier can successfully combine the advantages of LSTM and 1NN-LTW. Such as for the Spark-MLlib-LR class and Hadoop-WordCount classes, the hybrid algorithm achieve similar performance to the better one of LSTM and 1NN-LTW.
4) All the classifiers can successfully classify the test samples of the Web server class, which is reasonable as the Web server program is of a completely different kind from the other MapReduce programs.

TABLE IX
UNION-ACCURACY OF THE 1NN-LTW AND THE LSTM CLASSIFIERS IN THE FIVEFOLD TESTS

| Test case | $acc_{union}$ |
|---|---|
| Fold 0 | 0.9482 |
| Fold 1 | 0.9489 |
| Fold 2 | 0.9468 |
| Fold 3 | 0.9541 |
| Fold 4 | 0.9612 |

The above results show the difference of the 1NN-LTW and the LSTM classifier which makes the hybrid algorithm work. Although 1NN-LTW and the LSTM can achieve similar accuracy, their accurately classified samples have significant differences. To make this more clearly, we compute the union-accuracy $\text{acc}_{\text{union}}$ of the two classifiers as follows:

$$\text{acc}_{\text{union}} = \frac{|A_{\text{LSTM}} \cup A_{1\text{NN}-\text{LTW}}|}{N} \tag{17}$$

where $A_{\text{LSTM}}$ and $A_{1\text{NN}-\text{LTW}}$ are the sets of the accurately classified samples by LSTM and 1NN-LTW, respectively and $N$ is

TABLE X
TEST RESULTS WITH DIFFERENT SETTINGS FOR THE WARPING INDEX SET $G$ IN 1NN-LTW

| | $G = \{1\}$ | $G = \{1, .., 4\}$ | $G = \{1, ..., 8\}$ | $G = \{1, ..., 12\}$ |
|---|---|---|---|---|
| Fold 0 | 0.8166 | 0.8591 | 0.8727 | 0.8846 |
| Fold 1 | 0.8075 | 0.8552 | 0.8728 | 0.8904 |
| Fold 2 | 0.8138 | 0.8601 | 0.8684 | 0.8661 |
| Fold 3 | 0.8132 | 0.8668 | 0.8760 | 0.8802 |
| Fold 4 | 0.8103 | 0.8571 | 0.8878 | 0.8902 |

the total number of test samples in this test fold. The union-accuracy of the fivefold tests are shown in Table IX. It can be seen that the union-accuracy is between 94% and 96%. It shows the potentiality of a hybrid algorithm of the two classifiers. Note that the hybrid algorithm can only achieve accuracy smaller than the union-accuracy, as the union-accuracy is computed in an ideal manner.

## D. Discussion on the Parameter Settings

In this section, we discuss the parameters settings in the above algorithms. First we study the parameter used in the LTW measurement, the warping index set $G$. The test results with different $G$ settings are shown in Table X. It can be seen that a proper $G$ setting is needed as a set $G$ too small can deteriorate the performance. In our experiments we find that with a larger set $G$, the performance of LTW is more stable. Note that increasing the size of $G$ can cause higher computing cost.

Second, we discuss the parameter settings for the LSTM classifier. Tuning of the hyper-parameters of the LSTM network is critical. In our experiments, we find that an improper setting can result a bad performance with accuracy lower than 50% for the LSTM. We find the following key settings in the LSTM classifier, which we have tested and find the proper setting, although detailed experimental results are omitted here. The hyper-parameter settings: three parameters are specially tuned in our experiments, which are the batch size (we set to 60), the dimension of the LSTM node (we set to 90), and the discretized range parameter $S$ (we set to 100). We also tested two more different implementation variations of LSTM.

1) Adding a dropout layer, which is tested and not helpful in our case.
2) More than one LSTM layers, which has been tested and is also not helpful.

## E. Discussion on the Time Cost

In this section, we analyze the time cost of the proposed algorithm. First, we compare the time cost of the 1NN-LTW($G = \{1, \dots, 10\}$), 1NN-LTW($G = \{10\}$), 1NN-DTW ($w = 20$), LB_Keogh, and LSTM (decomposed into the training time cost and the test time cost). These algorithms are all implemented in Python and tested on a Ubuntu server 14.04 with Intel Xeon CPU E5-2620 v2 @ 2.10 GHz. The codes are implemented as single thread programs which can only utilize one core. For the LSTM, we show both its training time cost and its classification (test) time cost. The time cost data for different algorithms on fold 0 are shown in Table XI.

TABLE XI
RUN TIME COST (SECONDS) OF DIFFERENT ALGORITHMS

| | 1NN-LTW ($G = \{1, ..., 10\}$) | 1NN-LTW ($G = \{10\}$) | 1NN-DTW | LB_Keogh | LSTM Train | LSTM Test |
|---|---|---|---|---|---|---|
| Time Cost | 4695±1.2 | 2019±1.0 | 55171±237.5 | 3390±0.6 | 11043±79.9 | 294±0.1 |

From Table XI, we can observe the following.

1) The proposed LTW distance measurement is much faster than DTW. The time cost of LTW($G = \{10\}$) is similar to LB_Keogh, which shows that the algorithm itself can be very useful when the time cost matters.
2) The training time needed of LSTM is smaller than the time cost of 1NN-DTW in our experiment.

However, it should to be noticed that the time cost of LSTM is determined by the training epochs. The classification time cost of LSTM is only 294 s, which is much faster when compared with the other classifiers; this shows that after the neural network is trained, it can be used in classification with high time efficiency. Also note that as GPU-based neural network training is widely used in practice, the time cost of LSTM can be greatly reduced if GPU is used. We also note that the time cost of EETSC is high (335 074 s in training and 25 335 s in testing with code written in Java and runs with all CPU cores), this is because in the training procedure, EETSC tests many different parameter settings through internal cross validation within the training data. The test time cost shows that as it is ensemble of nine different classifiers, the time cost is high as expected.

## VI. CONCLUSION

In this paper, we study the server power consumption series classification problem used as a nonintrusive method for data center energy monitoring. First we propose a new time series distance measurement termed as LTW and build a hybrid algorithm of the 1NN with LTW and the LSTM neural network. The proposed LTW distance measurement is designed to be a light weight time series measurement with local warping operations within a predefined warping index set, and it is designed to be noncommutative. LTW can be taken as the simplified version of DTW with only the warping operation (a series of min operations). The LTW is proved to be better than DTW on our data set and its noncommutative feature is proved to be beneficial. Also a linear version of LTW can perform almost as good as the DTW on our data set. The proposed LTW shows that for a certain time series classification problem, it is possible to use some light weight time series distance measurement to achieve quite good classification accuracy.

Second, we apply the state-of-the-art sequential data modeling neural network LSTM to classify the power series. Our study show that LSTM can perform well compared to 1NN-LTW with similar accuracy; however, these two algorithms have their unique different natures and the accurately classified samples of these two algorithms have significant difference. In this sense, we propose a hybrid algorithm of the two classifiers termed as LSTM/LTW, which further improves the accuracy. The proposed hybrid algorithm can achieve classification accuracy as high as 93% in our experiments.

For the future work, one interesting problem is to study the case that the power series generated by multiple programs thus with multiple labels. The problem is especially interesting when we have the test data being the combination of different programs [such as a pair of programs ($A$, $B$)] where this special pair may not be seen in the training data, for example, the training data may only contain samples generated by program pairs like ($B$, $C$) and ($A$, $C$). In this case, the classifier should be able to recognize the new pair ($A$, $B$). Also, one can try more complicated ensemble algorithms with LSTM and other existed time series classification algorithms.

## REFERENCES

[1] "*America's Data Centers Consuming and Wasting Growing Amounts of Energy*. Accessed on Jul. 1, 2015. [Online]. Available: http://www.nrdc.org/energy/data-center-efficiency-assessment.asp

[2] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[3] W. Xia, Y. Wen, K.-C. Toh, and Y.-W. Wong, "Toward green data centers as an interruptible load for grid stabilization in singapore," *IEEE Commun. Mag.*, vol. 53, no. 11, pp. 192–198, Nov. 2015.

[4] W. Zhang, Y. Wen, Y. W. Wong, K. C. Toh, and C.-H. Chen, "Towards joint optimization over ICT and cooling systems in data centre: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1596–1616, 3rd Quart., 2016.

[5] J. Moore, J. Chase, K. Farkas, and P. Ranganathan, "Data center workload monitoring, analysis, and emulation," in *Proc. 8th Workshop Comput. Archit. Eval. Commercial Workloads*, 2005, pp. 1–8.

[6] A. Fehske, J. Gaeddert, and J. H. Reed, "A new approach to signal classification using spectral correlation and neural networks," in *Proc. New Front. Dyn. Spectr. Access Netw.*, 2005, pp. 144–150.

[7] S. S. Soliman and S.-Z. Hsue, "Signal classification using statistical moments," *IEEE Trans. Commun.*, vol. 40, no. 5, pp. 908–916, May 1992.

[8] M. B. I. Reaz, M. S. Hussain, and F. Mohd-Yasin, "Techniques of EMG signal analysis: Detection, processing, classification and applications," *Biol. Procedures Online*, vol. 8, no. 1, pp. 11–35, 2006.

[9] A. Bagnall, A. Bostrom, J. Large, and J. Lines, "The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version," *arXiv preprint arXiv:1602.01711*, 2016, accessed on May 16, 2017. [Online]. Available: https://arxiv.org/abs/1602.01711

[10] A. Stefan, V. Athitsos, and G. Das, "The move-split-merge metric for time series," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1425–1438, Jun. 2013.

[11] G. E. A. P. A. Batista, X. Wang, and E. J. Keogh, "A complexity-invariant distance measure for time series." in *Proc. SIAM SDM*, vol. 11. 2011, pp. 699–710.

[12] T. Rakthanmanon *et al.*, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Beijing, China, 2012, pp. 262–270.

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

[14] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000.

[15] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, USA, 2006, pp. 1033–1040.

[16] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[17] Y. Zhou and Y. Wei, "Learning hierarchical spectral–spatial features for hyperspectral image classification," *IEEE Trans. Cybern.*, vol. 46, no. 7, pp. 1667–1678, Jul. 2016.

[18] J. Xie, G. Dai, F. Zhu, L. Shao, and Y. Fang, "Deep nonlinear metric learning for 3-D shape retrieval," *IEEE Trans. Cybern.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/7801132/

[19] B. Du *et al.*, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1017–1027, Apr. 2017.

[20] C. A. Ratanamahatana and E. Keogh, "Everything you know about dynamic time warping is wrong," in *Proc. 3rd Workshop Min. Temporal Sequential Data, Conjunction 10th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Seattle, WA, USA, 2004.

[21] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2007.

[22] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proc. SIAM SDM*, vol. 1. 2001, pp. 5–7.

[23] T. Górecki and M. Łuczak, "Using derivatives in time series classification," *Data Min. Knowl. Disc.*, vol. 26, no. 2, pp. 310–331, 2013.

[24] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Paris, France, 2009, pp. 947–956.

[25] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *Proc. 13th SIAM Int. Conf. Data Min.*, Austin, TX, USA, 2013, pp. 668–676.

[26] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Min. Knowl. Disc.*, vol. 15, no. 2, pp. 107–144, 2007.

[27] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Inf. Sci.*, vol. 239, pp. 142–153, Aug. 2013.

[28] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Min. Knowl. Disc.*, vol. 29, no. 3, pp. 565–592, 2015.

[29] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: The collective of transformation-based ensembles," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2522–2535, Sep. 2015.

[30] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. INTERSPEECH*, vol. 2. 2010, p. 3.

[31] D. Tao, X. Lin, L. Jin, and X. Li, "Principal component 2-D long short-term memory for font recognition on single Chinese characters," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 756–765, Mar. 2016.

[32] H. Meng, N. Bianchi-Berthouze, Y. Deng, J. Cheng, and J. P. Cosmas, "Time-delay neural network for continuous emotional dimension prediction from facial expression sequences," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 916–929, Apr. 2016.

[33] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 2625–2634.

[34] J. Y.-H. Ng *et al.*, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 4694–4702.

[35] T.-H. Wen *et al.*, "Semantically conditioned LSTM-based natural language generation for spoken dialogue systems," *arXiv preprint arXiv:1508.01745*, 2015, accessed on May 16, 2017. [Online]. Available: https://arxiv.org/abs/1508.01745

[36] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[37] C. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms: Theory and Implementation*. New York, NY, USA: Wiley, 1991.

[38] *LSTM Networks for Sentiment Analysis*. Accessed on May 1, 2016. [Online]. Available: http://deeplearning.net/tutorial/lstm.html

[39] D. W. Hosmer, Jr., and S. Lemeshow, *Applied Logistic Regression*. New York, NY, USA: Wiley, 2004.

[40] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

**Yuanlong Li** (M'16) received the B.Sc. degree in mathematics and the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2009 and 2014, respectively.

He is currently a Research Fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His current research interests include black-box optimization, time series data analysis, and data driven solutions.

**Han Hu** (M'12) received the B.S. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2007 and 2012, respectively.

He is currently a Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. His current research interests include social media analysis and distribution, big data analytics, multimedia communication, and green network.

**Yonggang Wen** (S'99–M'08–SM'14) received the Ph.D. degree in electrical engineering and computer science (minor in western literature) from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2008.

He is an Associate Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He was with Cisco, San Jose, CA, USA, to lead product development in content delivery network, which had a revenue impact of 3 billion U.S. dollars globally. He has published over 140 papers in top journals and prestigious conferences. His current research interests include cloud computing, green data center, big data analytics, multimedia network, and mobile computing.

Dr. Wen was a recipient of the ASEAN ICT Award 2013 (Gold Medal) for his research in multiscreen cloud social TV has been featured by global media in over 1600 news articles from over 29 countries, and the Data Centre Dynamics Awards 2015 APAC for his research on Cloud3DView, as the only academia entry. He was a co-recipient of the 2015 IEEE Multimedia Best Paper Award, and the Best Paper Awards at the EAI/ICST Chinacom 2015, the IEEE WCSP 2014, the IEEE Globecom 2013, and the IEEE EUC 2012. He serves on Editorial Board of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the *IEEE Wireless Communication Magazine*, the IEEE COMMUNICATIONS SURVEY AND TUTORIALS, the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS, the IEEE ACCESS JOURNAL, and *Elsevier Ad Hoc Networks*. He was elected as the Chair of the IEEE ComSoc Multimedia Communication Technical Committee for the period 2014–2016.

**Jun Zhang** (M'02–SM'08–F'17) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

From 2004 to 2015, he was a Professor with Sun Yat-sen University, Guangzhou, China. Since 2016, he has been with the South China University of Technology, Guangzhou, where he is currently a Cheung Kong Chair Professor. His current research interests include computational intelligence, cloud computing, big data, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits. He has authored seven research books and book chapters, and over 100 technical papers in the above areas.

Prof. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS. He is the Founding and Current Chair of the IEEE Guangzhou Subsection, the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters, and the ACM Guangzhou Chapter.